NATIONAL OPEN UNIVERSITY OF NIGERIA

SCHOOL OF SCIENCE AND TECHNOLOGY

COURSE CODE: CIT 465

COURSE CODE: Network Administration

COURSE GUIDE

Course Code	CIT 465
Course Title	Network Administration
Course Developer/Writer	Dr. O. Oyebanji National Open University of Nigeria
Programme Leader	Prof. Kehinde Obidairo
Course Coordinator	

Headquarters
14/16 Ahmadu Bello Way
Victoria Island
Lagos

Abuja Annex
245 Samuel Adesujo Ademulegun Street
Central Business District
Opposite Arewa Suites
Abuja

e-mail: centralinfo@nou.edu.ng
URL: www.nou.edu.ng

National Open University of Nigeria 2010

First Printed 2011

ISBN:

All Rights Reserved

Printed by ……………..
For
National Open University of Nigeria

Introduction

Network Administration is a two credit load course for students offering Bachelor's Degree in Computer Science and other allied courses.

Network and system administration is a branch of *engineering* that concerns the operational management of human–computer systems. It is unusual as an engineering discipline in that it addresses both the technology of computer systems and the users of the technology on an equal basis. It is about putting together a network of computers (workstations, PCs and supercomputers), getting them running and then *keeping* them running in spite of the activities of *users* who tend to cause the systems to fail.

What You will Learn in this Course

This course consists of fifteen units and a course guide. This guide gives you a brief insight into what the course is all about, the necessary course materials and how you can work with them. In addition it advocates some general guidelines for the amount of time you are likely to spend on each unit of the course in order to complete it successfully.

It gives you guidance in respect of your Tutor-Marked Assignments which will be made available in the assignment file. There will be regular tutorial classes that are related to the course. It is advisable for you to attend these tutorial sessions. The course will equip you with the wherewithal needed to conduct network administration.

Aim of the Course

The purpose of the course material is to provide a mixture of theory and practice for a course in network and system administration; to extract those principles and ideas of network and system administration which do not change on a day-to-day basis; and to present them in a defensible manner

Course Objectives

To achieve the aims set out, the course has a set of objectives. Each unit has specific objectives which are included at the beginning of each unit. You should read these objectives before you study the unit. It is also advisable that you refer to them as you progress in your study of the unit to ascertain your progress. Also you should look at the objectives after completion of each unit. By so doing, you would know whether you have followed the instructions in the unit.

Below are the comprehensive objectives of the course as a whole. By meeting these objectives, you should have achieved the aims of the course. In addition to the aims above, this course sets to achieve some objectives. Thus after going through the course, you should be able to:

- Design a network which is logical and efficient.

• Deploy large numbers of machines which can be easily upgraded later.
• Decide what services are needed.
• Plan and implementing adequate security.
• Provide a comfortable environment for users.
• Develop ways of fixing errors and problems which occur.
• Keep track of and understand how to use the enormous amount of knowledge which increases every year.

Working through this Course

To complete this course you are required to read each study unit, read the textbooks and read other materials which may be provided by the National Open University of Nigeria.

Each unit contains self-assessment exercises and at certain points in the course you will be required to submit assignments for assessment purposes. At the end of the course there is final examination. Below you will find listed all the components of the course, what you have to do and how you should allocate your time to each unit in order to complete the course on time and successfully.

This course demands that you spend a lot of time to study. My advice is that you optimise the opportunity provided by the tutorial sessions where you have the opportunity of comparing your knowledge with that of your colleagues.

The Course Materials

The man components of the course are:

1. The Course Guide
2. Study Units
3. References/Further Readings
4. Assignments
5. Presentation Schedule

Study Unit

The study units in this course are as follows:

Module 1: Introduction to Network Basics and Administration

Unit 1: Network Administration: Scope, Goals, Philosophy and Standards

Unit 2: Network Structures

Unit 3: Network Technology

Unit 4: Network Protocols

Module 2: System Components and Management

Unit 1: System components

Unit 2: Networked communities

Unit 3: Host management

Unit 4: User management

Module 3: Network management

Unit 1: Methods of network administration: snmp & rmon

Unit 2: Configuration and maintenance

Unit 3: Diagnostics, fault and change management

Unit 4: Monitoring and system performance tuning

Module 4: Network simulation, documentation and security

Unit 1: Network simulation and documentation

Unit 2: Network security

Unit 3: Outlook and the future of network administration

Note each unit consists of one or two weeks' work and includes introduction, objectives, reading materials, exercises, conclusion and summary, Tutor-Marked Assignment (TMAs), references and other resources. The unit directs you to work on these exercises related to required reading. In general these exercises test you on the materials thereby assisting you to evaluate your progress and to reinforce your comprehension of the material. Together with the TMAs these exercises will help you in achieving the stated learning objectives of the individual units and of the course as a whole.

Presentation Schedule

Your course materials have important dates for early and timely completion and submission of your TMAs and attending tutorials. You should remember that you are required to submit all your assignments by the stipulated time and date. You should guide against falling behind in your work.

Assessment

There are three aspects to the assessment of the course. First is made up of self-assessment exercises, second consists of the TMA and third is the written examination/end of course examination.

You are advised to do the exercises. In tackling the assignments, you are expected to apply information, knowledge and techniques you gathered during the course. The assignments must be submitted to your facilitator for formal assessment in accordance with the deadlines stated in the presentation schedule and the assignment file. The work you submit to your tutor for assessment accounts for 30% of your total course work. At the end of the course you will need to sit for a final examination or end of course examination of about three hour duration. This examination will count for 70% of the total course mark.

Tutor-Marked Assignment

This is the continuous assessment component of your course. It accounts for 30% of the total score. You will be given four TMAs (4) to answer. Three of these must be answered before you are allowed to sit for the end of the course examination. The assignment questions for the units in the course are contained in the assignment file. You will be able to complete them through your reading the information contained in the reading materials, references and the study units. You are advised to research deeper into topics so as have a broader view of the discussions.

Endeavour to get the assignments to the facilitator on or before the deadline. If for any reason you cannot complete the work on time, contact your facilitator before the assignment is due to discuss the possibility of an extension. Extension will not be granted after the due date has passed unless on exceptional circumstances.

Final Examination and Grading

The end of course examination for Network Administration will be for about 3 hours and it has a value of 70% of the total course work. The examination will reflect the type of self-testing, practice exercise and tutor-marked assignment problems you have previously encountered. All these areas of the course will be assessed.

Use the time between finishing the last unit and sitting for the examination to revise the whole course. It might be useful to review your self tests, TMAs and the comments on them before the course examination. The end of course examination covers information from all parts of the course.

Course Marking Scheme

Assignment	Marks
Assignments 1-4	Four assignments, best three marks of the four count at 10% each – 30% of course marks
End of course examination	70% of overall course marks
Total	100% of course materials

Facilitators/Tutors and Tutorials

There are 21 hours of tutorials provided in support of this course. You will be notified of the dates, times and venues of these tutorials as well as the name and phone numbers of the facilitator, as soon as you are allocated to a tutorial group.

Your facilitator will mark and comment on your assignments, keep a close watch on your progress and any difficulties you might face and provide assistance to you during the course. You are expected to mail TMA to your facilitator at least two working days before the schedule date. The TMAs will be marked by your tutor returned back to you as soon as possible.

Do not delay to contact your facilitator by telephone or email if you need assistance.

The following might lead to your needing your facilitator's assistance:

- You do not understand any part of the study or assigned reading
- You have difficulty with the self test
- You have a question or a problem with an assignment or with the grading of an assignment

Endeavour to attend tutorials. It affords you the opportunity of face to face contact with the facilitator and to ask questions which are answered instantly. You also raise problems encountered in the course of study.

Summary

Network Administration is a course designed to equip you with what it takes to manage networked communities. A key task of network and system administration is to build hardware configurations; another is to configure software systems. Both of these tasks are performed for users. Each of these tasks presents its own challenges, but neither can be viewed in isolation.

I wish you the best and believe you will find the material very interesting.

Course Title	Network Administration
Course Developer/Writer	Dr. O. Oyebanji National Open University of Nigeria Lagos
Programme Leader	Prof. Kehinde Obidairo National Open University of Nigeria Lagos
Course Coordinator	Dr. O. Oyebanji National Open University of Nigeria Lagos

NATIONAL OPEN UNIVERSITY OF NIGERIA
Headquarters
14/16 Ahmadu Bello Way
Victoria Island

Lagos

Abuja Annex
245 Samuel Adesujo Ademulegun Street
Central Business District
Opposite Arewa Suites
Abuja

e-mail: centralinfo@nou.edu.ng
URL: www.nou.edu.ng

National Open University of Nigeria 2010

First Printed 2011

ISBN:

All Rights Reserved

Printed by ……………..
For
National Open University of Nigeria

TABLE OF CONTENTS

MODULE 1 INTRODUCTION TO NETWORK BASICS AND ADMINISTRATION

UNIT 1: Network Administration: Scope, Goals, And Philosophy

1.0	Introduction	1
2.0	Objectives	1
3.0	Main content	1
3.1	What is network and system administration?	1
3.2	Scope of Network administration	2
3.3	The goal of Network administration	3
3.4	The challenges of system administration	3
3.5	The Meta principles of system administration	3
3.6	Advice to the students	4
4.0	Conclusion	5
5.0	Summary	5
6.0	Tutor-Marked Assignments	6
7.0	References/Further Reading	6

UNIT 2: Network Structure

1.0	Introduction	7
2.0	Objectives	7
3.0	Main Content	7
3.1	Physical Topology	7
3.4	Ring Topology	11
3.5	Hybrid Topology	12
4.0	Conclusion	13
5.0	Summary	13
6.0	Tutor-Marked Assignments	13
7.0	References/Further Reading	14

UNIT 3: Network Technology

1.0	Introduction	15
2.0	Objectives	15
3.0	Main Content	15
3.1	Categories of Networks / Network Technologies	15
4.0	Conclusion	20
5.0	Summary	20
6.0	Tutor-Marked Assignments	21
7.0	References/Further Reading	21

UNIT 4: PROTOCOLS, PACKETS AND STANDARDS

1.0	Introduction	22
2.0	Objectives	22
3.0	Main Content	22
3.1	Protocols, Packets and Standards	22
3.2	Protocols	24
3.3	The layered approach to computer communications	25
3.4	Standards	28
3.5	The OSI model	30
3.6	The TCP/IP protocol	34
4.0	Conclusion	40
5.0	Summary	40
6.0	Tutor-Marked Assignments	40
7.0	References/Further Reading	41

MODULE 2: SYSTEM COMPONENTS AND MANAGEMENT

UNIT 1: SYTSTEM COMPONENTS

1.0	Introduction	42
2.0	Objectives	42
3.0	Main Content	42
3.1	What Is ‘The System’?	42
3.2	Handling hardware	44
3.3	Operating systems	47
3.4	Filesystems	54
3.5	Networks	59
3.6	IPv4 networks	68
3.7	Address space in IPv4	72
3.8	IPv6 networks	74
4.0	Conclusion	78
5.0	Summary	78
6.0	Tutor Marked Assignment	79
7.0	References/Further Reading	79

UNIT 2: NETWORKED COMMUNITIES

1.0	Introduction	80
2.0	Objectives	80

3.0 Main Content 80
3.1 Communities and Enterprises 80
3.2 Policy Blueprints 81
3.3 System Uniformity 82
3.4 User behavior: socio-anthropology 83
3.5 Clients, servers and delegation 83
3.6 Host identities and name services 85
3.7 Common network sharing models 87
3.8 Local Network Orientation and Analysis 90
4.0 Conclusion 99
5.0 Summary 99
6.0 Tutor Marked Assignments 100
7.0 References/Further Reading 101

UNIT 3: HOST MANAGEMENT
1.0 Introduction 102
2.0 Objectives 102
3.0 Main Content 102
3.1 Computer startup and shutdown 102
3.4 Configuring and personalizing workstations 104
3.6 Installation of the operating system 112
4.0 Conclusion 114
5.0 Summary 114
6.0 Tutor-Marked Assignments 115
7.0 References/Further Reading 115

UNIT 4: USER MANAGEMENT

1.0 Introduction 116
2.0 Objectives 116
3.0 Main Content 116
3.1 Issues 116
3.2 User Registration 116
3.3 Account Policy 121
4.0 Conclusion 122
5.0 Summary 122
6.0 Tutor-Marked Assignments 123

7.0 References/Further Reading 123

MODULE 3: NETWORK MANAGEMENT

UNIT 1: METHODS OF NETWORK ADMINISTRATION: SNMP & RMON

1.0 Introduction 124
2.0 Objectives 124
3.0 Main Content 124
3.1 What Is Network Management? 124
3.2 SNMP Network Management 128
3.3 SNMP Tools 131
3.4 TCP/IP Remote Network Monitoring (RMON) 133
4.0 Conclusion 134
5.0 Summary 134
6.0 Tutor-Marked Assignments 135
7.0 References/Further Reading 135

UNIT 2: CONFIGURATION AND MAINTENANCE

1.0 Introduction 136
2.0 Objectives 136
3.0 Main Content 136
3.1 System Configuration Policy 136
3.2 Methods: controlling causes and symptoms 137
3.3 Change management 138
3.4 Automation of host configuration 139
4.0 Conclusion 142
5.0 Summary 143
6.0 Tutor-Marked Assignments 144
7.0 References/Further Reading 144

UNIT 3: DIAGNOSTICS, FAULT AND CHANGE MANAGEMENT

1.0 Introduction 145
2.0 Objectives 145
3.0 Main Content 145
3.1 Fault Tolerance and Propagation 145
3.2 Faults 147
4.0 Conclusion 156
5.0 Summary 156
6.0 Tutor-Marked Assignments 157
7.0 References/Further Reading 157

UNIT 4: MONITORING AND SYSTEM PERFORMANCE TUNING

1.0 Introduction 158
2.0 Objectives 158
3.0 Main Content 158
3.1 Monitoring 158
3.2 System performance tuning 159
4.0 Conclusion 165
5.0 Summary 165
6.0 Tutor-Marked Assignments 166
7.0 References/Further Reading 166

MODULE 4: NETWORK SIMULATION, DOCUMENTATION AND SECURITY

UNIT 1: NETWORK SIMULATION AND DOCUMENTATION

1.0 Introduction 167
2.0 Objectives 167
3.0 Main Point 167
3.1 What Is Network Simulation? 167
3.2 Network Simulator 168
3.3 Simulations 168

3.4 Documentation 169
4.0 Conclusion 170
5.0 Summary 171
6.0 Tutor-Marked Assignments 171
7.0 References/Further Reading 171

UNIT 2: NETWORK SECURITY

1.0 Introduction 172
2.0 Objectives 172
3.0 Main Content 172
3.1 Principles of Security 172
3.2 Security Implementation 179
4.0 Conclusion 196
5.0 Summary 196
6.0 Tutor-Marked Assignments 197
7.0 References/Further Reading 197

UNIT 3: OUTLOOK AND THE FUTURE OF NETWORK ADMINISTRATION

1.0 Introduction 199
2.0 Objectives 199
3.0 Main Content 199
3.1 Outlook 199
3.2 Pervasive computing 199
3.3 The future of system administration 200
4.0 Conclusion 201
5.0 Summary 201
6.0 Tutor-Marked Assignments 202
7.0 References/Further Reading 202

MODULE 1: INTRODUCTION TO NETWORK BASICS AND ADMINISTRATION

UNIT 1: NETWORK ADMINISTRATION: SCOPE, GOALS, AND PHILOSOPHY

UNIT 2: NETWORK STRUCTURES

UNIT 3: NETWORK TECHNOLOGY

UNIT 4: NETWORK PROTOCOLS

UNIT 1: NETWORK ADMINISTRATION: SCOPE, GOALS, AND PHILOSOPHY

1.0 INTRODUCTION

Network and distribution processing systems are of critical and growing importance in business, government and other organizations. Therefore, networks must be managed for effectiveness and efficiency. This unit discusses fundamental aspects of network administration.

2.0 OBJECTIVES

At the end of this unit, you should be also to:

- Define network administration
- Know the scope of network administration
- State the goals of system administration
- Understand the challenges of system administration
- State the Meta principles of system administration

3.0 MAIN CONTENT

3.1 What is network and system administration?

Network and system administration is a branch of *engineering* that concerns the operational management of human–computer systems. It is about putting together a network of computers (workstations, PCs and supercomputers), getting them running and then *keeping* them running in spite of the activities of *users* who tend to cause the systems to fail.

A system administrator works for users, so that they can use the system to produce work. However, a system administrator should not just cater for one or two selfish needs, but also work for the benefit of a whole community. Today, that community is a global community of machines and organizations, which spans every niche of human society and culture, thanks to the Internet. It is often a difficult balancing act to determine the best policy, which accounts for the different needs of everyone with a stake in a system. Once a computer is attached to the Internet, we have to consider the consequences of being directly connected to all the other computers in the world.

In the future, improvements in technology might render system administration a somewhat easier task – one of pure resource administration – but, today, system administration is not just an administrative job, it is an extremely demanding engineer's job. It's about hardware, software, user support, diagnosis, repair and prevention. System administrators need to know a bit of everything: the skills are technical, administrative and socio-psychological.

The terms *network administration* and *system administration* exist separately and are used both variously and inconsistently by industry and by academics.

System administration is the term used traditionally by mainframe and Unix engineers to describe the management of computers whether they are coupled by a network or not. To this community, network administration means the management of network infrastructure devices (routers and switches). The world of personal computers (PCs) has no tradition of managing individual computers and their subsystems, and thus does not speak of system administration, *per se*. To this community, network administration is the management of PCs in a network. In this material, we shall take the first view, since this is more precise.

Network and system administration are increasingly challenging. The complexity of computer systems is increasing all the time. Even a single PC today, running Windows NT, and attached to a network, approaches the level of complexity that mainframe computers had ten years ago. We are now forced to think *systems* not just computers.

3.2 Scope of Network administration

The management of a network, usually called network administration, consists of procedures and services that keep the network running properly. An important part of network management entails making sure that the network is available (or up and running as IT professionals say) when employees and managers need it. Other admin activities are:

- Monitoring the network capacity to ensure that all transmission requirements can be met.
- Adding capacity to the network by increasing band width interconnecting additional modes, or creating and interconnecting additional networks.
- Training people to use the network effectively
- Assisting IT professionals in organizational applications that will make good use of the network's capabilities.

- Backing up the network software and data regularly to protect against the failure of network or any of its components
- Putting security procedures in place to make certain that only authorized users have access to the network and ensuring that all security procedures are followed
- Making sure the network personnel can respond quickly and effectively in the event of a network operational or security failure.
- Diagnosing and troubleshooting problems on the network and determining the best course of action to take to solve them.

3.3 The goal of Network administration

The goal is to keep the network running properly and configuring and managing services that are provided over the network.

There are many services that we use regularly. There are some which work in the background enabling other services to run smoothly.

3.4 The challenges of system administration

System administration is not just about installing operating systems. It is about planning and designing an efficient *community* of computers so that real *users* will be able to get their jobs done. That means:

• Designing a network which is logical and efficient.

• Deploying large numbers of machines which can be easily upgraded later.

• Deciding what services are needed.

• Planning and implementing adequate security.

• Providing a comfortable environment for users.

• Developing ways of fixing errors and problems which occur.

• Keeping track of and understanding how to use the enormous amount of knowledge which increases every year.

Some system administrators are responsible for both the hardware of the network and the computers which it connects, i.e. the cables as well as the computers. Some are only responsible for the computers. Either way, an understanding of how data flow from machine to machine is essential as well as an understanding of how each machine affects every other.

3.5 The Meta principles of system administration

Many of the principles in this course material derive from a single overriding issue: they address the *predictability* of a system. The term system clearly implies an operation that is *systematic*, or predictable – but, unlike simple mechanical systems, like say a clock, computers interact with humans in a complex cycle of feedback, where uncertainty can enter at many levels. That makes human–computer systems difficult to predict, unless we somehow fix the boundaries of what is allowed, as a matter of policy.

Principle (Policy is the foundation). *System administration begins with a policy – a decision about what we want and what should be, in relation to what we can afford.*

Policy speaks of what we wish to accomplish with the system, and what we are willing to tolerate of behavior within it. It must refer to both the component parts and to the environment with which the system interacts. If we cannot secure predictability, then we cannot expect long-term conformance with a policy.

Principle (Predictability). *The highest level aim in system administration is to work towards a predictable system. Predictability has limits. It is the basis of reliability, hence trust and therefore security.*

Policy and predictability are intertwined. What makes system administration difficult is that it involves a kind of 'search' problem. It is the hunt for a stable region in the landscape of all policies, i.e. those policies that can lead to stable and predictable behavior. In choosing policy, one might easily promote a regime of cascading failure, of increasing unpredictability that degenerates into chaos. Avoiding these regimes is what makes system administration difficult.

As networks of computers and people grow, their interactions become increasingly complex and they become *non-deterministic*, i.e. not predictable in terms of any manageable number of variables. We therefore face another challenge that is posed by inevitable growth:

Principle (Scalability). *Scalable systems are those that grow in accordance with policy; i.e. they continue to function predictably, even as they increase in size.*

These meta-themes will recur throughout this material. The important point to understand about predictability is that it has limits. Human–computer systems are too complex and have too many interactions and dependencies to be deterministic. When we speak of predictability, it must always be within a margin of error. If this were not the case, system administration would not be difficult.

3.6 Advice to the students

To study this subject, we need to cultivate a way of thinking which embodies a basic scientific humility and some core principles:

• Independence or self-sufficiency in learning. We cannot always ask someone for the right answer to every question.

• Systematic and tidy work practices.

• An altruistic view of the system. Users come first: collectively and only then individually.

• Balancing a fatalistic view (the inevitability of errors) with a determination to gain firmer control of the system.

Some counter-productive practices could be avoided:

• The belief that there exists a right answer to every problem.

• Getting fraught and upset when things do not work the way we expect.

• Expecting that every problem has a beginning, middle and an end (some problems are chronic and cannot be solved without impractical restructuring).

We can begin with a checklist:

• Look for answers in manuals and newsgroups.

• Use controlled trial and error to locate problems.

• Consider all the information; listen to people who tell you that there is a problem. It might be true, even if you can't see it yourself.

• Write down experiences in an A–Z so that you learn how to solve the same problem again in the future.

• Take responsibility for your actions. Be prepared for accidents. They are going to happen and they will be your fault. You will have to fix them.

• Remember tedious jobs like vacuum cleaning the hardware once a year.

• After learning about something new, always pose the question: *how does this apply to me?*

SELF ASSESSMENT EXERCISES

1. Is system administration management or engineering?
2. Why does the physical environment play a role in system administration?

4.0 CONCLUSION

Network administration is concerned with establishing and administering overall goals, policies and procedures of network management. This requires a well rounded skills set and not just technical skills.

5.0 SUMMARY

Network and system administration is a branch of *engineering* that concerns the operational management of human–computer systems. System administration is not just about installing operating systems. It is about planning and designing an efficient *community* of computers so that real *users* will be able to get their jobs done. *System administration begins with a policy – a decision about what we want and what should be, in relation to what we can afford.* Policy speaks of what we wish to accomplish with the system, and what we are willing to tolerate of behavior within it. To study this subject, we need to cultivate a way of thinking which embodies a basic scientific humility and some core principles:

6.0 TUTOR-MARKED ASSIGNMENTS

1. State the top-most principles that guide network and system administrators
2. What kinds of issues does system administration cover?
3. State the meta principles of system administration
4. What are the challenges of system administration?

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

3. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley

4. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

UNIT 2: NETWORK STRUCTURE

1.0 INTRODUCTION

This unit provides a survey of the basic network structures or topologies. Topology can be considered as a virtual shape or structure of a network.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- To name the four basic network topologies
- To cite advantages and disadvantages of each type
- State the criteria necessary for an effective and efficient network

3.0 MAIN CONTENT

3.1 PHYSICAL TOPOLOGY

The term physical topology refers to the way in which a network is laid out physically. Two or more devices connect to a link; two or more links form a topology. The topology of a network is the geometric representation of the relationship of all the links and linking devices (usually called nodes) to one another. There are four basic topologies possible: mesh, star, bus and ring (see figure 1)

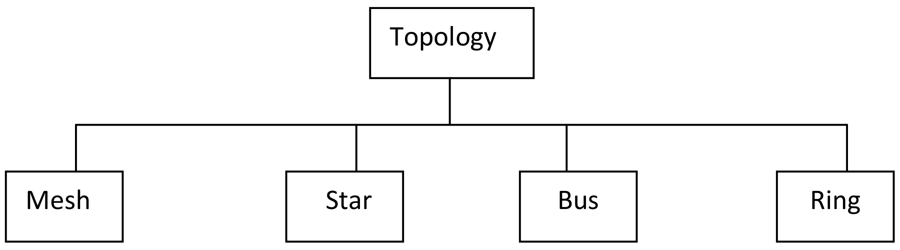


Figure 1 Categories of topology

3.1.1 Mesh

In a mesh topology, every device has a dedicated point to point link to every other device. The term dedicated means that the link carries traffic only between the two devices it connects. (see figure 2)

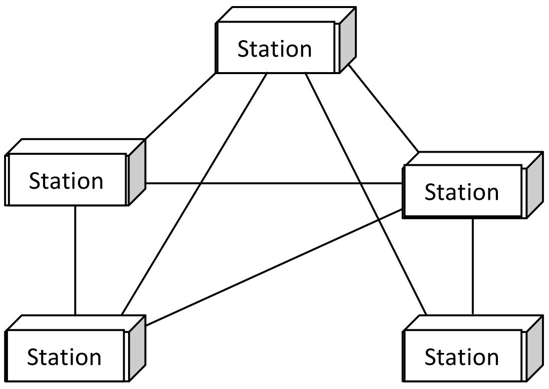


Figure 2 A fully connected mesh topology (five devices)

A mesh offers several advantages over other network topologies. First, the use of dedicated links guarantees that each connection can carry its own data load, thus eliminating the traffic problems that can occur when links must be shared by multiple devices. Second, a mesh topology is robust. If one link becomes unavailable it does not incapacitate the entire system. Third, there is the advantage of privacy or security. Whenever message travels along a dedicated line, only the intended recipient sees it. Finally, point to point links make fault identification and fault isolation easy. Traffic can be routed to avoid links with suspected problems. This enables the network manger to discover the precise location of the fault and aids in finding its cause and solution.

The main disadvantages of a mesh are related to the amount of cabling and that of I/O ports required. First, because every device must be connected to every other device, installation and reconnection are difficult. Second, the sheer bulk of the wiring can be greater than the available space (in walls, ceilings or floors) can accommodate. Finally, the hardware required to connect each link (I/O ports and cables) can be prohibitively expensive. For these reasons, a mesh topology is usually implemented in a limited fashion, for example, as a backbone connecting the main computers of a hybrid network that can include several other topologies.

One practical example of a mesh topology is the connection of a telephone regional office in which each regional office needs to be connected to every other regional office.

3.1.2 Star Topology

In a star topology, each device has a dedicated point-to-point link only to a central controller, usually called a hub. These devices are not directly linked to one another. Unlike a mesh topology, a star topology does not allow direct traffic between devices. The controller acts as an exchange. If one device wants to send data to another it sends the data to the controller, which then relay the data to the other connected devices (see figure 3)

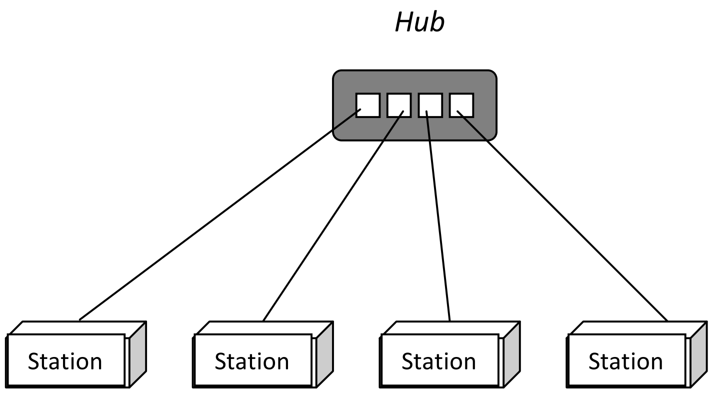


Figure 3 A star topology connecting four station

A Star topology is less expensive than a mesh topology. In a star, each device needs only one link and one I/O port to connect to any number of others. This factor also makes it easy to install and reconfigure. Far less calling needs to be housed, and additions, moves and deletions involve only one connection between that device and the hub.

Other advantages include robustness. If one link fails, only that link is affected. All other remain active. This factor also lends itself to easy fault identification and fault isolation. As long as the hub is working, it can be used to monitor link problems and bypass detective links.

One big disadvantage of a star topology is the dependency of the whole topology on one single point, the hub. If the hub goes down, the whole system is dead.

Although a star requires far less cable than a mesh, each node must be linked to a central hub. For this reason, often more cabling is registered in a star then in some other topologies (such as ring or bus)

3.1.3 Bus Topology

A bus topology is multipoint. One long cable acts as a backbone to link all the devices in the network (see figure 4)

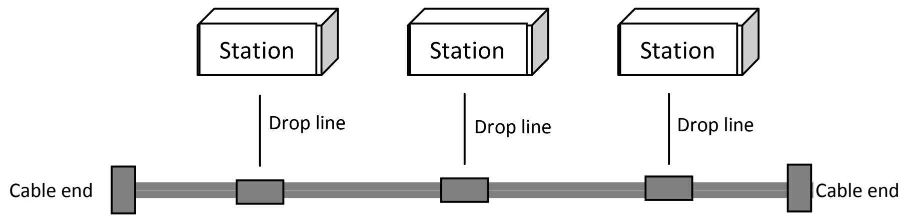


Figure 4 A bus topology connecting three stations

Nodes are connected to the bus cable by drop lines and taps. A drop line is a connection running between the device and the main cable. A tap is a connector that either splices into the main cable or punctures the sheathing of a cable to create a contact with the metallic core. As a signal travels along the backbone, some of its energy is transformed into heat. Therefore, it becomes weaker and weaker as it travels farther and farther. For this reason, there is a limit on the number of taps a bus can support and on the distance between those taps.

Advantages of a bus topology include ease of installation. Backbone cable can be laid along the most efficient path and then connected to the nodes by drop lines of various lengths. In this way, a bus uses less cabling than mesh or star topologies.

Disadvantages include difficult reconnection and fault isolation. A bus is usually designed to be optimally efficient at installation. It can therefore be difficult to add new devices. Signal reflection at the taps can cause degradation of quality.

In addition, a fault or break in the bus cable stops all transmission, even between devices on the same side of the problem. The damaged area reflects signals back in the direction of origin, creating noise in both directions.

Bus topology was the one of the first topologies used in the design of early local area networks

3.4 Ring Topology

In a ring topology, each device has a dedicated point to point connection with only the two devices on either side of it. A signal is passed along the ring in one direction from device to device, until it reaches its destination. Each device in the ring incorporates a repeater. When a device receives a signal intended for another device, its repeater regenerates the bits and passes them along (see figure 5).

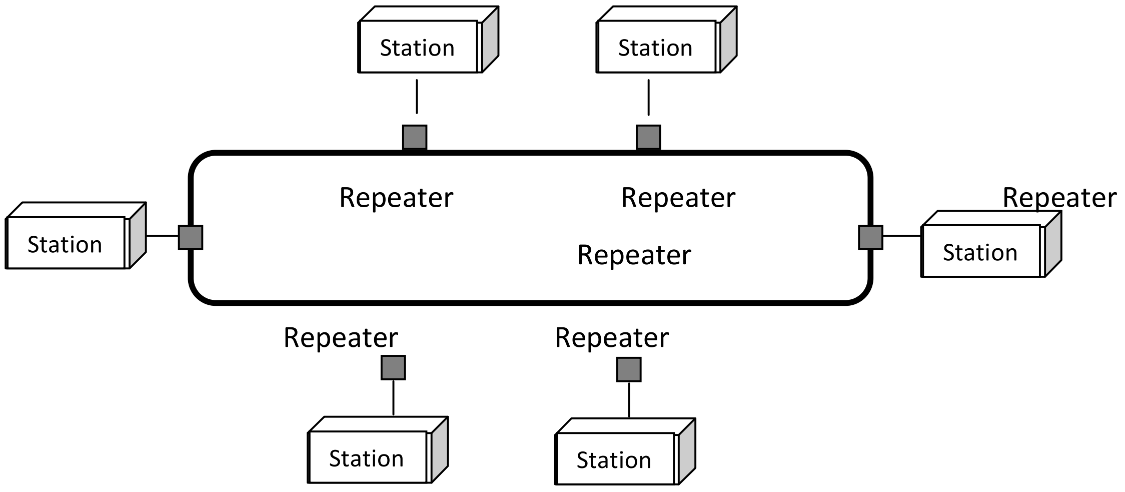


Figure 5 ***A ring topology connecting six stations***

A ring is relatively easy to install and reconfigure. Each device is linked to only its immediate neighbors (either physically or logically). To add or delete a device requires changing only two connections. The only constraints are media and traffic considerations (maximum ring length and number of devices). In addition, fault isolation is simplified. Generally in a ring, a signal is circulating at all times. If one device does not receive a signal within a specified period, it can issue an alarm. The alarm alerts the network operator to the problem and its location.

However, unidirectional traffic can be a disadvantage. In a simple ring, a break in the ring (such as a disabled station) can disable the entire network. This weakness can be solved by using a dual ring or a switch capable of closing off the break.

Today, the need for higher speed LANS has made this topology less popular

3.5 Hybrid Topology

A network can be hybrid. For example, we can have a main star topology with each branch connecting several stations in a bus topology as shown in Figure 6.

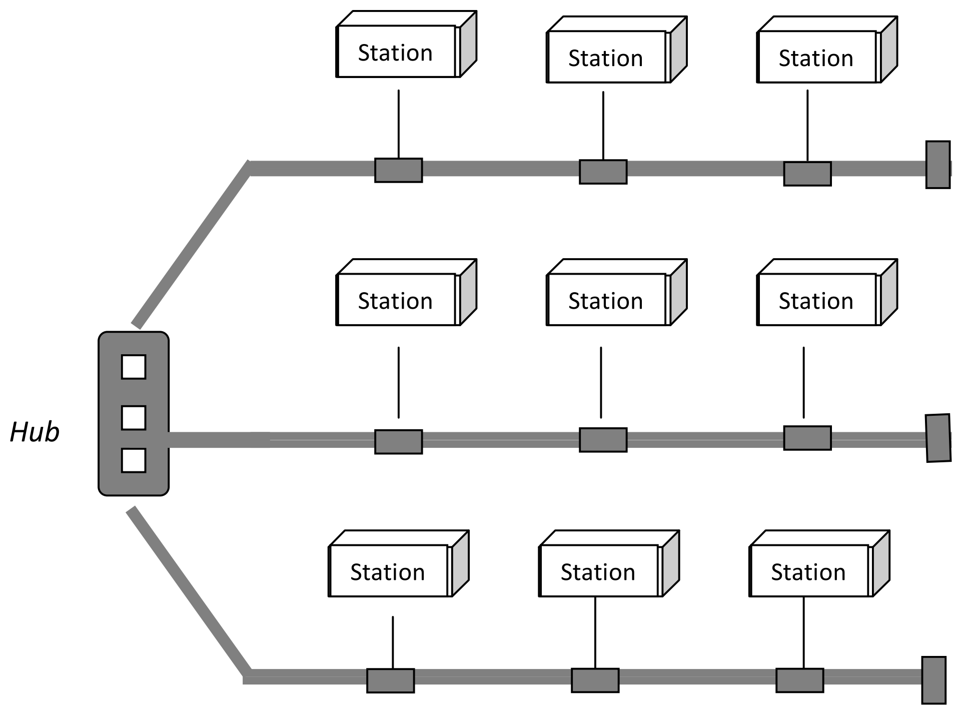


Figure 6 A hybrid topology: a star backbone with three bus networks

SELF ASSESSMENT EXERCISES

a) What are the three criteria necessary for an effective and efficient network?

b) What is network topology?

4.0 CONCLUSION

In the context of a communication network, the term topology refers to the very in which the end points, or stations attached to the network are interconnected. Topologies are the important part of the network design theory. A better network can be built if you have the knowledge of these topologies and if you know the difference between each topology.

5.0 SUMMARY

Topology refers to the physical or logical arrangement of a network. Devices may be arranged in a mesh, star, bus or ring topology.

A mesh offers several advantages over other network topologies. The main disadvantages of a mesh are the number of I/O ports required.

Star topology is less expensive than a mesh topology. One big disadvantage of a star topology is the dependency of the whole topology on one single point, the hub.

A bus topology is multipoint unlike mesh and star topologies that are point to point connections. An advantage of a bus topology is ease of installation. Disadvantages include difficult reconnection and fault isolation.

Ring topology is relatively easy to install and reconfigure. However, unidirectional traffic can be a disadvantage.

Hybrid topology is complex which can be built of two or more above networked topologies.

6.0 TUTOR-MARKED ASSIGNMENTS

1. For each of the following four networks, discuss the consequences if a connection fails.

a) Five devices arranged in a bus topology

b) Five devices arranged in a ring topology

2. For n devices in a network, what is the number of cable links required for a mesh, ring, bus and star topology?

3. Name the four basic networking topologies and cite on advantage of each type.

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

3. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley

4. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

5. Subramanian, M. (2000). Network Management: Principles and Practice, Addison-Wesley

UNIT 3: NETWORK TECHNOLOGY

1.0 INTRODUCTION

This unit looks at what constitutes a local area network (LAN), then a wide area network (WAN) and then discusses the differences between the two. We then discuss the technologies for implementing WAN.

2.0 OBJECTIVES

- Know about the categories of networks.
- State the distinctions between LAN and WAN.
- Understand the technologies used in implementing WAN.

3.0 MAIN CONTENT

3.1 Categories of Networks / Network Technologies

Today when we speak of networks, we are generally referring to two primary categories: local area networks (LANs) and wide area networks (WANs).

The category into which a network falls is determined by its size. A LAN normally covers an area less than 2 miles; a WAN can be worldwide. Networks of a size in between are normally referred to as metropolitan area networks (MANs) and span tens of miles.

3.1.1 Local Area Network (LAN)

One type of network that becomes ubiquitous is the local area network. Indeed, LAN is to be found in virtually all medium and large size office buildings. Depending on the needs of an organization and the type of technology used, a LAN can be as simple as two PCs and a printer in someone's home office; or it can extend throughout a company and include audio and video peripherals. Currently LAN size is limited to a few kilometers. LANs are designed to allow resources to be shared between personal computers or workstations. The resources to be shared can include hardware (e.g. a printer), software (e.g. an application program) or data

In addition to size, LANs are distinguished from other types of networks by transmission media and topology. In general, a given LAN will use only one type of transmission medium. The mos common LAN topologies are bus, ringed star.

Early LANS had data rates in the 4 to 16 megabits per seconds (mbps) ranges. LANs come in a parallel of different configurations. The most common is switched LANs and wireless LANs. The most switched LAN is a switched Ethernet LAN, which may consist of a single switch with a parallel of attached devices, or parallel of interconnected switches. Today, however, speeds are normally 100 or 1000 mbps. Wireless LANs are the newest evolution in LAN technology.

3.1.2 Wide Area Network (WAN)

A wide area network (WAN) provides long distance transmission of data, image, audio, video information over large geographic area that may comprise a country, a continent or even the whole world. WAN can be as complex as the backbones that connect the Internet or as simple as a dial-up line that a home computer to the Internet. We normally refer to the first as a switched WAN and to the second as a point to point WAN. The switched WAN connects the end systems which usually comprise a router (internet – working connecting devices) that connects together LAN or WAN. The point to point WAN is normally a line leased from a telephone or cable T.V provider that connects a home computer or a small LAN to an Internet service provider (ISP). This type of WAN is often used to provide Internet access. Wireless WANs are become more and more popular. Traditionally, WANs have been implemented using one of two technologies: Circuit switching and packet switching. More recently, frame relay and asynchronous transfer mode (ATM) networks have assumed major roles.

Circuit Switching

In a circuit- switching network, a dedicated communications path is established between two stations through the nodes of the network. That path is a connected sequence of physical links between nodes. On each link, a logical channel is dedicated to the connection. Data generated by the source station are transmitted along the dedicated path as rapidly as possible. At each mode, incoming data are routed or switched to the appropriate outgoing channel without delay. The most common example of circuit switching is the telephone network.

Packet Switching

A quite different approach is used in a packet switching network. In this case, it is not necessary to dedicate transmission capacity along a path through the network. Rather, data are sent out in a sequence of small chunks, called packets. Each packet is passed through the network from node to node along some path leading from source to destination. At each node, the entire packet is received, stored briefly, and then transmitted to the next node. Packet switching networks are commonly used for terminal to computer communications.

Frame Relay

Packet switching was developed at a time when digital long distance transmission facilities exhibited a relatively high error rate compared to today's facilities. As a result, there is a considerable amount of overhead built into packet switching schemes to compensate for errors. The overhead includes additional bits added to each packet to introduce redundancy and additional processing at the end stations and the intermediate switching nodes to detect and recover from errors.

With modern high-speed communication systems, this overhead is unnecessary and counterproductive. It is unnecessary because the rate of errors has been dramatically lowered and any remaining errors can easily be caught in the end systems by logic that operates above the level of the packet-switching logic. It is counterproductive because the overhead involved soaks up a significant fraction of the high capacity provided by the network.

Frame relay was developed to take advantage of these high data rates and low error rates whereas the original packet-switching networks were designed with a data rate to the end user of about 64 kbps. Frame relay networks are designed to operate efficiently at user data rate of up to 2mbps. The key to achieving these high data rates is to strip out most of the overhead involved with errors control.

Asynchronous Transfer Mode (ATM)

Sometimes referred to as cell relay is a culmination of developments in circuit switching and packet switching. ATM can be viewed as an evolution from frame relay. The most obvious difference between frame relay and ATM is that frame relay uses variable length packets called frames and ATM uses fixed length packets, called cells. As with frame relay, ATM provides little overhead for error control depending on the inherent reliability of the transmission system and on higher layers of logic in the end of systems to catch and correct errors. By wiring a fixed packet length, the processing overhead is reduced even further for ATM compared to frame relay. The result is that ATM is designed to work in the range of 10s and 100s of mbps and in the Gbps range. ATM can also be viewed as an evolution from circuit switching; only fixed-data-rate circuits are available to the end system. ATM allows the definition of multiple virtual channels with date rate that are dynamically defined at the time the virtual channel is created. By using small, fixed-size cells, ATM is so efficient that it can offer a contant-data rate channel even though it is using a packet-switching technique. Thus ATM extends circuit switching to allow multiple channels with the data rate on each channel dynamically set on demand.

3.1.3 Distinctions between LANs and WANs

There are several key distinctions between LANs and WANs. Among which are:

1. The scope of the LAN is small, typically a single building or a cluster of buildings. This difference in geographic scope leads to different technical solution.
2. It is usually the case that the LAN is owned by the same organization that owns the attached devices. For WANs, this is less often the case, or at least a significant fraction of the network assets is not owned. This has two implications. First, care must be taken in the choice of LAN, because there may be a substantial capital investment (compared to dial-up or leased charges of WANs) for both purchase and maintenance. Second, the network management responsibility for a LAN falls solely on the user.
3. The internal data rates of LANs are typically much greater than those of WANs.

3.1.4 Metropolitan Area Network (MAN)

A MAN is a network with a size between a LAN and WAN. It normally covers the area inside a town or a city. It is designed for customers who need a high-speed connectivity, normally to the Internet and have end points spread over a city or part of a city.

3.1.5 Interconnection of Networks: Internetwork

Today, it is very rare to see a LAN, a MAN or a WAN in isolation; they are connected to one another. When two or more networks are connected, they become an internetwork or internet.

SELF ASSESSMENT EXERCISES

What is an internet?

What is the Internet?

4.0 CONCLUSION

Whereas wide area networks may be public or private, LANs usually are owned by the organization that is using the network to interconnect equipment. LANs have much greater capacity than WANS to carry what is generally a greater internal communication load.

5.0 SUMMARY

In general terms, communications networks can be categorized as local area networks (LANs) and wide area networks (WANs).

A LAN consists of a shared transmission medium and a set of hardware and software for interfacing devices to the medium and regulating the orderly access of the medium. LAN size is limited. In addition to size, LANs are distinguished from other types of networks by their transmission media and topology. A WAN provides long-distance transmission over large geographic areas. WAN is often used to provide Internet access. Traditionally, WANs have been implemented using one of two technologies: circuit switching and packet switching. Wireless WANs are becoming more and more popular.

6.0 TUTOR-MARKED ASSIGNMENTS

1. What are the advantages of packed switching compared to circuit switching?

2. What are some of the factors that determine whether a communication system is LAN or WAN?

3. Outline the distinctions between LAN and WAN.

4. Discuss circuit-switching network.

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

3. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley

4. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

UNIT 4: PROTOCOLS, PACKETS AND STANDARDS

1.0 INTRODUCTION

This unit discusses packets and protocols which are the fundamental building blocks of data transmission over the network.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- State why are protocols and standards needed?
- State the function of a packet.
- State the principal function of protocols in a network.
- Understand the layered approach to computer communications.
- Discuss some of the different protocols and their uses.
- State how protocols work.
- State the advantages and disadvantages of standards.

3.0 MAIN CONTENT

3.1 PROTOCOLS, PACKETS AND STANDARDS

All data that is transmitted across the network is put into packets containing information about the source and destination of the data. These packets are created using standards or protocols. Since there are many different network configurations, there are many different protocols. By having a variety of protocols, you can choose the one that best fulfills the needs of your network.

3.1.1 Function of packets

The function of a packet is to carry data from one point to another. Protocols require that packet contain some basic information about their source and their destination and in many cases, protocols require that the packet include a *checksum*. A checksum is a number that can be used to verify that the packet has been transferred across the network without being corrupted.

3.1.2 Packet Structure

The structure of the packet is extremely important. Useless a packet is structured exactly as it is supposed to be; it is ignored by the receiving party and assumed to be corrupted. Basic packet structure requires that the packet include a *header* section, a *data* section, and in most cases, a *cyclic redundancy check* (CRC) section (also called a *trailer*). Not every protocol requires that a CRC be attached.

3.1.2.1 Header

The header section of a packet contains the routing information. This information includes the source and destination of the packet. The header also contains the number of the packet, which is generated when the packet is created. In addition, the header can contain a protocol version number, the length of the header, the type of service, the length of the entire packet, the flags, the time to live, and other information.

3.1.2.2 Data

The data is the actual information that is being transmitted over the network from one application to another. Each protocol has a predefined maximum data size. If the data is larger than this maximum data size, the data is broken into smaller pieces and transmitted in multiple packets.

3.1.2.3 CRC

A CRC (Cyclic redundancy check) is calculated prior to the data being sent and attached to the bottom of a packet. At the destination, a new CRC is computed and compared to the original to verify that the packet was not corrupted. A CRC is usually attached to the bottom of a packet, but some protocols include CRC within the header.

3.1.3 Creating packets

Before data is transmitted across the network, it is broken into smaller, more manageable pieces called *packets.* All packets are numbered so they can be put back together when they reach their destination. The header, which contains the source address, destination address, and packet number, along with other information, is attached to the beginning of the packet. A CRC is then calculated and added to the end of the packet.

3.1.4 Encapsulation

Encapsulation is the process of encoding data for transmitting it across the network. Once a packet is created as described previously, in order for the packet to be transmitted to its final destination, it may need to use a protocol in addition to the one that it is currently using. A header and CRC are then added to the newly created packet. This packet is an encapsulated packet. Figure 1 illustrates an encapsulated packet.

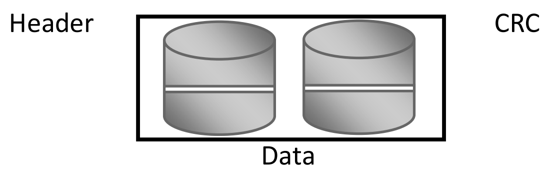


Figure 1 Encapsulated packet

3.2 Protocols

In computer networking, communication occurs between entities in different systems. An entity is anything capable of sending or receiving information. However, two entities cannot simply send bit streams to each other and expect to be understood. For communication to occur, the entities must agree on a protocol. A protocol is a set of nodes that govern data communications. A protocol defines what is communicated, how it is communicated, and when it is communicated. The key elements of protocol are syntax, semantics and timing.

Syntax: The term syntax refers to the structure or format of the data, meaning the order in which they are presented. For example, a simple protocol might expect the first 8 bits of data to be the address of the sender, the second 8 bits to be the address of the receiver, and the rest of the stream to be message itself.

Semantics:

The word semantics refers to the meaning of each section of bits. How is a particular pattern to be interpreted, and what action is to be taken based on that interpretation? For example, does the address identify the route to be taken or the final destination of the message?

Timing:

The term timing refers to two characteristics: When data should be sent and how fast they can be sent. For example, if a sender produces data of 100mbps and the receiver can process data at only 1 mbps, the transmission will overload the receiver and some data will be lost.

3.2.1 How Protocols Work

When an application first attempts to transfer data across the network, the data is usually tc large to be sent in a single transmission. To meet the need of networking, the protocol th governs the transmission of the data first breaks the data into packets. The protocol numbers each of the packets so can later be put back together when they arrive at their destination and transmits each of the packets across the network. In addition to this numbering, information on the source, destination, and the protocol used is added to the header of the packet.

Protocols are the first software layer to receive data that has been transmitted across the network. After all packets have been received, they are put back together using the numbers that were placed in the header at the origin of the packet. Once the data has all been put back together, it can be used by the application that the data was sent across the network to.

3.2.2 Functions of protocols.

The principal functions of protocols in a network are line access and collision avoidance. Line access concerns how the sending device gains access the network to send a message. Collision avoidance refers to managing message transmission so that the messages do not collide with each other on the network. Other functions of protocols are to identify each device in the communication path, to secure the attention of the other device, to verify correct receipt of the transmitted message, to verify that a message requires transmission because it cannot be correctly interpreted and to perform recovery when errors occur.

3.3 The layered approach to computer communications

In order to enable two or more computers to communicate in a meaningful manner, we must define with great care all aspects of the communication process (i.e. we must define a ‘communications protocol’). By way of a useful analogy, let us consider the situation in which the director of a company in the UK wishes to communicate with a person in another company located in China. The director may ask a secretary to put a call through and will provide sufficient information for the secretary to identify the person who is to be contacted. Here, the director will not give the actual phone number- it may be left to the secretary to obtain this information. From this point, the director has no further involvement until the phone connection is in place. The secretary will locate and dial the number and this will initiate various electronic/software activities. Neither the director nor the secretary has any interest in knowing how the electronic and software systems will route the call. It may be carried by electronic cables, fiber optic cables, or be routed via a satellite. Additionally, it may use communications systems that route the call across the Atlantic through the US and then across the Pacific Ocean, or it may be routed in an easterly direction. These low-level issues are of little interest to the secretary – a number is dialed and processes occur that result in a phone ringing in an office somewhere in China. Hopefully, the intended recipient is available and the secretary notifies the director. Both parties must now adopt/agree on a common language and must exercise a degree of hand-shaking (in this sense we mean that only one person should talk at any one time). Finally, at the end of the conversation, an acceptable convention is used to bring the call to a conclusion. All these issues form part of the ‘communications protocol’ that is needed to enable a useful dialogue and it is important to note that the elements that underpin the communication do not need to have any knowledge of the overall purpose that they will serve. For example:

25

- The secretary does not necessarily know why the call is to be placed – the information exchange may be confidential to the company director and the recipient of the phone call.
- The keypad via which the secretary enters the phone number converts the key presses into electrical signals. These signals are dispatched and initiate various routing actions. However, the keypad is not involved in these actions – it serves a single function.
- The director has no knowledge of the path taken by the 'voice signals' as they are routed to China. Perhaps they pass via trans-oceanic cables or are beamed to an orbiting satellite.
- Any cables used during the conversation have no 'knowledge' of the meaning that will be placed on the digital signals that they transmit.

The establishment of a communications protocol that enables computers (and other digital systems) to communicate is, in many ways, similar to the protocols used to support the sort of phone conversation referred to in the above analogy (although computer communications are perhaps more complex). To handle the design implementation and maintenance of such systems, a 'layered' approach is adopted. In figure 2, we indicate two computers that need to communicate. Perhaps, for example, an applications program running on Node A wishes to send a data file to a similar program running on Node B (just as in the same way the company director mentioned above wishes to talk to a person in a remote location). In order to transmit the data a number of tasks must be performed, and these are carried out by layers of software located on both nodes.

Each layer carries out a number of specific tasks and directly communicates with the immediately adjacent software layers. However, from a logical point of view each layer communicates with a corresponding layer on the remote computer – i.e. corresponding software layers located on the two nodes have similar/equivalent functionality. The lowest layer on either node is responsible for interfacing with the physical interconnect.

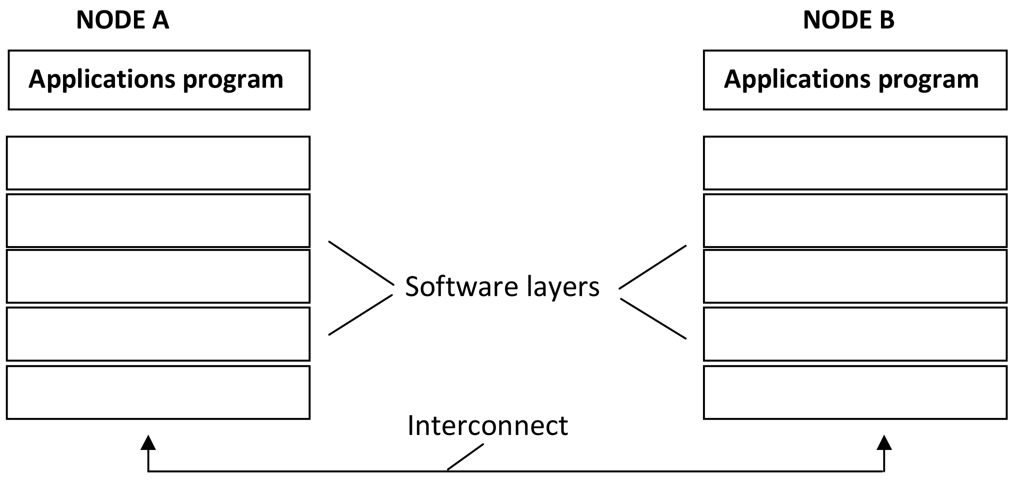


Figure 2: An applications program on Node A wishes to send data to an equivalent program on Node B

In order for Node A to transmit a data file to Node B, various events must take place.

For example:

- Node A must inform the communications network of the identity of the destination system (Node B)
- Node A must ensure that Node B is prepared to receive the data.
- The file transfer applications program on Node A must ensure that the file management program on the destination system is ready to accept and store the file.
- If the file formats used on the two systems are incompatible, one or other system must perform a format translation function.
- File transfer must be carried out in an orderly manner and in such a way that the two communicating machines do not block other network traffic. This will involve splitting the data file into packets (chunks) and appending various information to each packet.
- Node B provides acknowledgement of receipt
- Node B reassembles the packet in other to reconstruct the original data file
- Node B must attempt to detect any errors in data it has received. In some cases Node B may be able to correct errors.
- In the case that secure transmission is required, the data may be encrypted by Node A prior to transmission. Node B must then perform the reverse process.

To achieve this high degree of cooperation between computers, the tasks are broken into subtasks that are implemented individually using a layered approach. These layers form the data communication protocol architecture. Example of such layer architectures are: the Open System Interconnection (OSI) model, and the Transmission Control Protocol/ Internet Protocol (TCP/IP). Key advantages of a layered structure include:

- The complex communication protocol is divided into subtasks and these are implemented within a layered structure. Each layer has limited functionality and this 'divide and conquer' approach facilitates the design and the implementation of the system.
- Higher-level layers need have no knowledge of tasks performed by the lower layers. Thus, for example, a higher-level layer needs no knowledge of the type of interconnect that is in use. Again, this facilitates the design process.
- When changes are made to the communications protocol, only certain relevant layers need to be modified/replaced. This makes it easier to upgrade software and undertake software testing.

Structuring software using a layered approach tends to result in *larger programs* which run *more slowly* than if a non-layered approach were to be adopted. However, these two weaknesses are outweighed by the benefits that are associated with the layered approach - especially in terms of providing a structured framework within which the complex issues associated with computer communications may be resolved.

3.4 Standards

Are essential in creating and maintaining an open and competitive market for equipment manufacturers and in guaranteeing national and international interoperability of data and telecommunications technology and processes. Standards provide guidelines to manufacturers, vendors, government agencies and other service providers to ensure the kind of interconnectivity necessary in today's market place and in international communication.

Standards play an important role in our everyday lives and facilitate the operation of products produced by different manufacturers. For example:

- Countries adopt a standard type of mains plug and socket. Without such a standard, we would find that we had to continually rewire mains plugs or employ some form of adaptor. This provides an example of national standard.
- Car manufacturers adopt a standard for the relative placement of the clutch, brake and accelerator pedals. This provides an example of global standard.
- Computers are equipped with standard interface sockets (e.g. serial, parallel and USB) via which they are able to connect to peripheral devices. This provides an example of global standard.

Standards may come into being in various ways. For example:

- A standard may be established (imposed) by the company that plays the most dominant role in any particular area. For example, the serial and parallel ports employed by today's PC were implemented on the earliest PCs introduced by IBM. They soon became standard for desktop computing
- A standard may gradually evolve
- A standard may be developed/defined by a committee of experts.

Although standardization can facilitate our use of technologies and products, standards seldom reflect an optimal solution. For example, the VHS videotape format became a standard, while other superior and equally cost-effective formats fell by the wayside. Furthermore, in the case of standards developed by committees, these often reflect many technological compromises and take long periods to develop. Such standards are often out of date even before they are released!

From a computer user's perspective, standards are extremely important because they allow a combination of products from different manufacturers to be used together. Standards ensure greater compatibility and interoperability between various types of equipment and technologies.

In data communications, standards provide guidelines to manufacturers and service providers to ensure compatibility, connectivity, and interoperability of technologies – an essential requirement in today's global market. Key advantages of standards are:

- To ensure a large market for hardware or software products – thus encouraging mass production
- To allow products from different vendors to communicate, thus giving customers more flexibility in the selection and use of equipment.

On the other hand, standards do have limitations:

- They tend to slow down technological change. This is due to the fact that, in some cases, by the time a standard is developed, subjected to scrutiny, reviewed, compromised and endorsed by all concerned parties – and then disseminated, more efficient technologies could have developed.
- Many standards may exist for the same thing. It is often difficult to decide which standard will provide better compatibility and remain in place for the greatest amount of time.

Many official computer-related standards are defined by the following organizations:

- **ANSI** (America National Standards Institute)
- **ITU** (International Telecommunication Union)
- **IEEE** (Institute of Electrical and Electronic Engineers)
- **ISO** (International Organization for Standardization)
- **VESA** (Video Electronics Standards Association).

Car drivers generally use agreed signals when turning left or right. Aero plane pilots follow specific standardized rules for communicating throughout the world. Similarly, for any computer-based systems to communicate successfully, they need to use 'the same language'. This means that what is communicated, how it is communicated, and when it is communicated must conform to some mutually acceptable conventions agreed between the parties involved. These conventions are known as a 'protocol', which can be defined as a set of rules governing the exchange of data between two or more devices.

Typical tasks performed by protocols are as follows:

- To make sure that the source device activates the data communication line
- To inform the transmission system about the destination system.
- To make sure that the source device communicates with the destination device before sending data
- To make sure the destination device is ready to accept the data
- To make sure that the destination file management system is ready to store incoming files
- To ensure compatibility between source and destination, and to perform format translation.

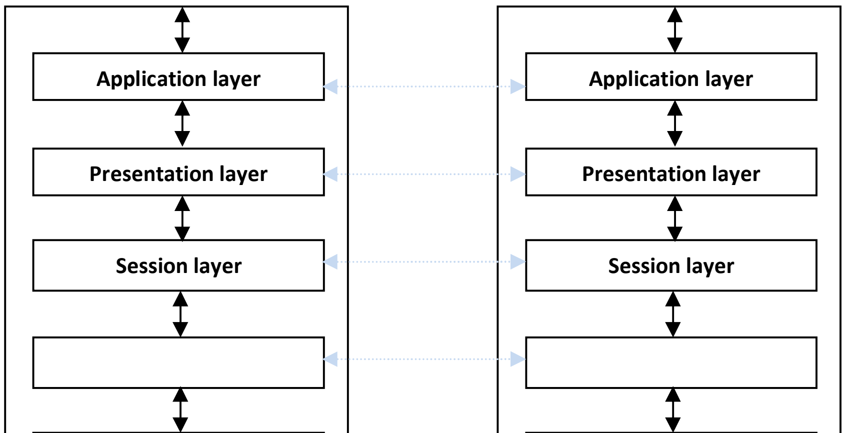
In the 1980s, many companies entered the desktop computing market and this led to a rich diversity of products. Unfortunately, these systems would often not operate together, nor could software developed for use on one particular type of machine necessarily be used on another. In short, although the lack of standards enabled product diversity, it hampered computer usage. Quickly, standards were developed (and/or evolved) and these impacted on many areas of computing. For example:

- **Compatibility improved.** By conformance to standards, hardware and software systems developed by different manufacturers could be used together (although there were often unforeseen problems)
- **The diversity of available products decreased**
- **Backwards compatibility became an important issue.** For example, a new model of computer, or a new release of an operating system should support the function of older products. This has greatly increased hardware and software complexity and retarded the development of radically new computer products.

3.5 The OSI model

The Open System Interconnection (OSI) reference model was developed by the International Standards Organization (ISO) and provides a framework for protocol development. By implementing a communication protocol that adheres to the OSI model, systems developed by different manufacturers are able to communicate. The tasks that must be performed to enable machines to communicate in an effective and efficient manner are incorporated within a seven-layer hierarchy, as indicated in figure 3. Although the protocols detailed within this reference

model are seldom used, the model provides us with an excellent conceptual framework for understanding the tasks performed by the various software layers. Below we briefly summarize aspects of the functionality of the various layers.



Node A Node B

Figure 3: The layers within the OSI reference model

3.5.1 Application layer

This should not be confused with the applications programs that may be running on a computer. The application layer provides network access to the user and to applications programs. This layer passes data to (and receives data from) the presentation layer, and logically communicates directly to the application layer on the remote computer. This is indicated in figure 3 where the horizontal lines indicate the logical communication of each layer with its remote counterpart. The application layer needs know nothing of the tasks carried out by the lower layers – it needs only interface with the user (and applications programs) and with the presentation layer.

3.5.2 Presentation layers

Different computers may employ different character set formats. A user is not interested in such differences and one of the tasks undertaken by the presentation layer is to translate between different formats that may be used to represent numbers, characters and other symbols. Additionally, the presentation layer is also involved in ensuring secure data transmission (consequently, when data is being transmitted the presentation layer undertakes encryption, and when data is being received it performs decryption).

3.5.3 Session layer

A user applications program may need to open a ‘session’ with a remote machine. For example, a user may wish to log on to a remote computer and carry out various tasks and this will involve the transmission and reception of data over a period of time. This necessitates synchronization whereby each node knows when it can transmit and when it is to receive data (i.e. when it must ‘listen’). The session layer deals with this synchronization and additionally is involved in error recovery. Consider the case that a file is being transmitted between two nodes, and during this process the network fails. Without the support of the session layer it would be necessary to start the transmission process again from the beginning. However, the session layer inserts checkpoints into the transmitted data stream and these are used to efficiently recover from such failures. Following a failure, transmission can be recommenced from the point at which the last checkpoint was successfully delivered to the destination node. The session layer carries out various other activities, such as bracketing a set of related and non-independent activities. For example, there may be a need to carry out a task on a remote machine, which involves the execution of series of commands. Perhaps if only some of these commands are executed (i.e. they are not carried out in their entirety) problems will ensue. If the individual commands are executed as each arrives at the remote machine then, in the case that the network connection fails, there is the likelihood of incomplete execution. One task performed by the session layer relates to the buffering of such commands – as each arrives it is temporarily stored and not passed to higher layers until all commands (and any associated data) have been received. The series of commands may then execute in full.

3.5.4 Transport layer

This acts as the intermediary between the lower layers (whose implementation is dependent on the underlying network architecture) and the three upper layers which provide user services and whose architecture is (at least in principle) independent of the detailed network characteristics.

The type of transport service that is provided to the session layer is determined by the transport layer. Suppose a node wished to send an extremely large file to a remote machine via a shared network (or set of interconnected networks). Without judicious design (in relation to the type of transport service used), there is the possibility that such a transmission could block the network(s) in such a way that whilst the transmission is in progress no other machines could communicate. The approach commonly used to prevent such a situation is to split the data into chunks (‘packets’) which are individually encapsulated within a frame containing all the necessary data needed to enable a packet delivery to the intended destination. The splitting of the data into smaller units is carried out by the transport layer. These packets may traverse a set of networks by different routes and so arrive at their destination out of order. The transport layer reorders packets and so enables them to be correctly reassembled.

3.5.5 Network Layer

This layer decides on routing issues, determining the path that should be followed by packets when they traverse networks. In fact, in such a situation the path taken is not defined solely by the source node but by all the nodes (network devices) through which packets pass on their way to the destination. Consider the situation illustrated in figure 4

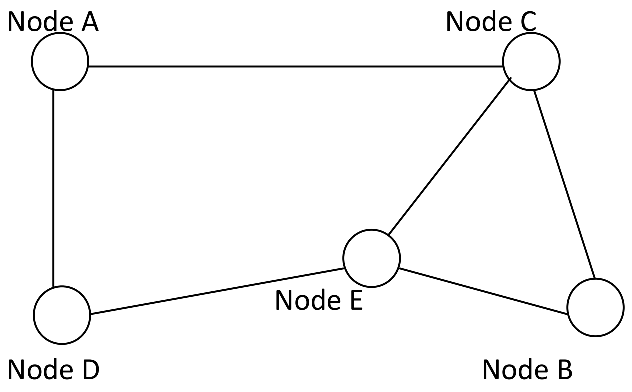


Figure 4: A simple network in which a packet may be sent from Node A to B via different routes. The circles represent nodes, and the lines network interconnects.

Suppose that a packet is to be sent from Node A to Node B. The packet will have to pass through at least one intermediate node (network device). These nodes may simply forward the packet, or may decide on the direction of the next step in its voyage. Thus, for example, Node D simply performs a forwarding function, whereas Nodes C and E are able to make routing decisions. The transport layer plays a critical role in determining the time it will take for packets to reach their destination and in this sense the actions of the transport layer impact on transmission latency.

3.5.6 The data link layer

This layer is responsible for various low-level network specific tasks and plays a crucial part in the detection and correction of errors that may occur during the transmission process.

Correction may be achieved by means of additional information inserted into messages prior to their transmission that can be used to modify bits corrupted during the transmission process.

Alternatively, correction may involve requesting re-transmission. Additionally, the data link layer plays a pivotal role in managing network access and ensuring that network ‘collisions’ (which occur when two or more nodes attempt to transmit onto the same LAN at the same time) are handled correctly. Devices connected together via networks do not necessarily demonstrate the same transmission/reception characteristics.

Thus a device able to transmit at high speed (i.e. that has high bit-rate) could readily swamp a slower recipient. Buffering techniques are used to circumvent this problem and this necessitates a protocol that ensures that the capacity of the buffer is not exceeded. This is referred to as flow control.

3.5.7 The physical layer

This layer deals with the transmission of the bit stream through the transmission medium, and the connection strategy used to enable the transfer of bits from one node to another. Thus the physical layer defines the signal levels, the type of transmission medium employed (e.g. twisted pair cable, coaxial cable, fiber optic cable), and also the techniques that will be used to permit the passage of data, such as circuit switching (in which a dedicated path is set up between two communicating nodes), packet switching, etc.

3.6 The TCP/IP protocol

In the late 1960s, the US Department of Defence's Advance Research Project Agency (ARPA) initiated a project that centered upon the interconnection of geographically dispersed computing systems. Gradually a large-scale network of university and government computing facilities evolved (this network was named ARPANET), which used packet switching techniques and initially employed leased phone lines. Early networking protocols were slow and unreliable and in 1974 a new set of protocols were proposed. These formed the basis for TCP/IP (Transmission Control Protocol/Internet Protocol (TCP/IP) which today underpins the operation of the Internet.

A protocol such as TCP/IP must support a number of essential requirements such as:

- **Reliability:** in terms of both data integrity and timely delivery
- **Fault tolerance:** the failure of a network segment should not seriously disrupt overall network operation; it must be possible to route packets along different paths so that the can still reach their destination
- **Transparent communications:** different computer systems and LANs should be able to communicate transparently.

It is convenient to employ a layered model in order to most readily conceptualize TCP/IP. We can therefore consider TCP/IP within a four-layer framework (a five–layer model is sometimes preferred). In figure 5 these layers are depicted, and are placed alongside the layers that comprise the OSI model. Below we briefly summarize aspects of their role

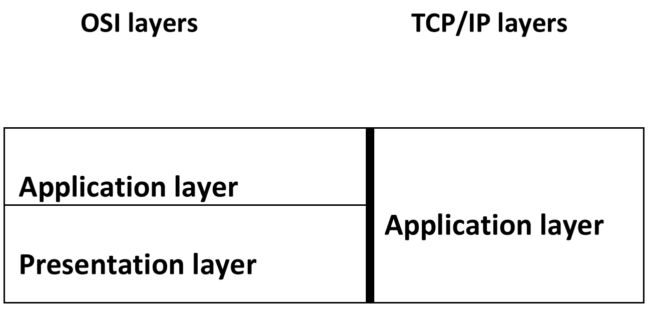
3.6.1 Application layer

This layer provides communication services to the user and to applications programs. It can be viewed as corresponding to the application, presentation and session layers found in the OSI model. The application layer contains all the high-level protocols (such as those that we commonly encounter when accessing the Internet – such as DNS (Domain Name System) and HTTP).

3.6.2 Transport layer

Two different protocols are defined in this layer (TCP and UDP (User Datagram Protocol)). These differ in a number of important respects. For example:

- **Reliability:** in the case of UDP, error correction is not implemented – the onus for this activity is placed on the applications program. This contrast with TCP in which error detection and correction form an integral part. Free from error correction overheads, UDP can (under some circumstances) demonstrate high performance
- **Flow control:** in the case of TCP, flow control is implemented and this prevents a faster machine from swamping a recipient that operates more slowly.



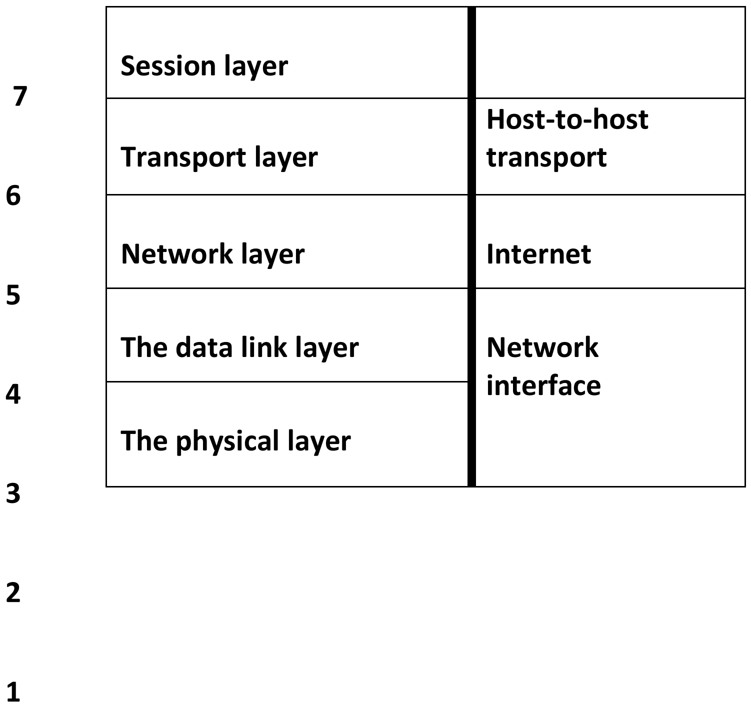
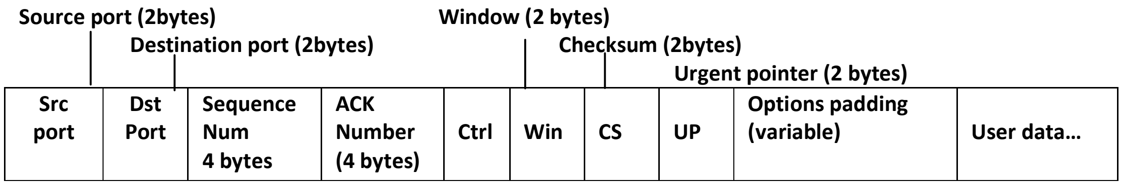
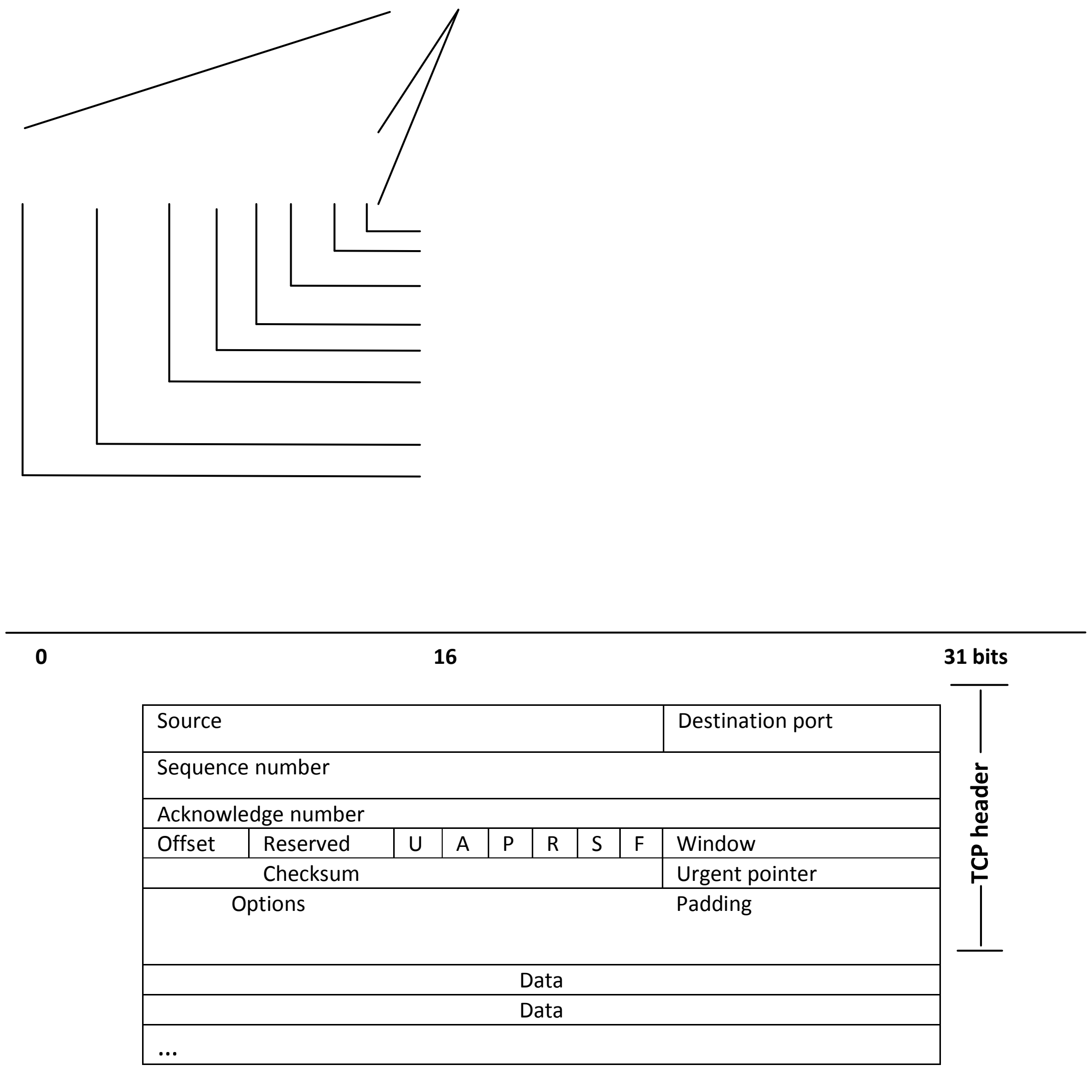


Figure 5: A conceptual model of TCP/IP set alongside the layers that comprise the OSI model

A stream of data that is to be transmitted is fragmented into chunks and the transport layer appends various information, before passing these to the internet layer. At the receiving node, the transport layer reassembles these data chunks. In the case of TCP, the transport layer encapsulates the data chunks into a TCP segment (in the case of UDP, the encapsulated data is usually referred to as a packet. There are differences between the information contained in the UDP and TCP headers.) Here the data is provided with a 'header' containing various important information; see Figure 6. It is instructive to consider the purpose of several pieces of information contained in the header:

Source and Destination ports: many well known (widely used) application protocols are designated by unique identification numbers provided by the 'Internet Assigned Numbers Authority'. For example, the File Transfer Protocol (FTP) is identified as "port21', and the Simple Mail Transfer Protocol (SMTP) as 'port 25'. TCP inserts this information into the header and thereby provides information on the source and destination applications protocol associated with the data to be transferred. The source port and destination port fields are each two bytes long, and values below 256 are used to reference 'well-known' ports.





Source								Destination port
Sequence number								
Acknowledge number								
Offset	Reserved	U	A	P	R	S	F	Window
	Checksum							Urgent pointer
	Options							Padding
				Data				
				Data				
...								

Figure 6: Information contained within a TCP segment

- **Sequence number:** TCP numbers each byte of data that is transmitted between two nodes during the transfer process. The sequence number references the first byte of data encapsulated within frame. This is most readily understood by means of an example. Suppose that a set of frames are transmitted between node A and node B, and that each contains 256 bytes of data. Then the sequence numbers contained in the first four frames transmitted by Node A could be 1,257,513, 769 (the process is slightly more complex since the sequence number of the first frame need not be 1). Node B these sequence numbers to reconstruct the data chunks and correct for frames being received out of their transmitted order

- **Header length:** this enables the receiving node to determine the point at which the header ends and the data starts. It is necessary to specify this length as not of fixed size.
- **Checksum:** this enables the transport layer to perform error detection
- **Options:** various options can be included. For example, one option enables the recipient to inform the source node about the maximum segment size that it is able to accept. This is indicated during the establishment of a communication and ensures that the recipient's buffer will not be swamped by a high-speed transfer.

Internet layer

At the sending node, the Internet layer takes packets or segments generated by the transport layer, and further encapsulate these to produce datagrams. The additional information appended by the Internet layer (the 'IP header') is intended to enable the datagrams to be injected onto any network and travel (via intermediate networks) to the intended destination. During their transit, intermediate network devices will use this information to determine the direction they should take. Since the routing of packets is fundamental to the Internet layer, it may be considered to be equivalent to the network layer used in the OSI model.

Network interface layer

In terms of its functionality, this layer is equivalent to the lowest two layers used in the OSI model. It further encapsulates a datagram received from the Internet layer producing a 'frame'. This layer makes the connection to the transmission medium and employs the appropriate protocol for launching and receiving frames.

The process of encapsulation referred to above is summarized in figure 7 and in Table 1 an overview of the functionality of the layers that have been conceptualized in connection with TCP/IP is presented.

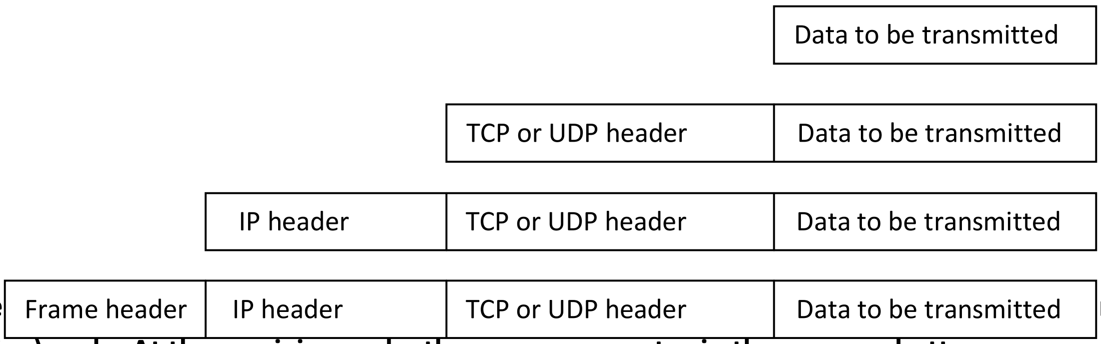


Figure **rce (sending) node. At the receiving node, the process operates in the reverse: bottom up.**

Application (4)

- Similar to OSI application layer
- Serves as communication interface by providing specific application services
- Examples include email, virtual terminal, file transfer, WWW

Transport (3)

- Defined by two protocols:

  User Datagram protocol (UDP)

  - a connectionless protocol
  - provides unreliable datagram service (no end -to-end error detection or correction)
  - does not retransmit any unreceived data
  - requires little overhead
  - application protocols include Trivial File Transfer Protocol (TFTP), Network File System (NFS), Simple Network Management Protocol (SNMP), Bootstrap Protocol (BOOTP), and Domain Name Service (DNS)

  Transmission Control Protocol (TCP)

  - (the TCP of TCP/IP)
  - connection–oriented protocol
  - provides reliable data transmission via end-to-end detection and correction
  - guarantees data is transferred across a network accurately and in correct order
  - retransmits any data not received by destination node
  - guarantees against data duplication between sending and receiving nodes
  - application protocols include Telnet, FTP, SMTP and POP

Internet (2)

- (The IP of TCP/IP)
- Transfers user messages from source host to destination host
- Connectionless datagram service
- Route selection is based on a metric
- Uses Internet or IUP addresses to locate a host within the Internet
- Relies on routers or switches
- Integral part is Internet Control Message Protocol (ICMP); this uses an IP datagram to carry messages about state of communications environment

Network Interface (1)

- Connects host to the local network hardware
- Makes a connection to the physical medium
- Uses a specific protocol for accessing the medium
- Places data into frames
- Effectively performs all functions of the first two layers of the OSI model

Table 1: A summary of some aspects of the functionally of the conceptualized four-layer TCP/IP model

4.0 CONCLUSION

You have been introduced to the basic concepts of packets, protocols and standards. You are now in a position to relate these concepts to how communication occurs between entities in different systems in your environment.

5.0 SUMMARY

Packets and protocols are the fundamental building blocks of data transmission over the network. A packet is a segment of data that has a header with destination and addressing information attached to it.

A protocol is a set of rules that govern data communication; the key elements of a protocol are syntax, semantics and timing.

Standards are necessary to ensure that products from different manufacturers can work together as expected. The ISO, ITU-T, ANSI, IEEE and EIA are some of the organizations involved in standards creation.

To enable two or more computers to communicate in a meaningful manner, a communication protocol must be defined.

We briefly summarize aspects of the functionality of the various layers of the OSI model and layers of the TCP/IP protocol.

TCP/IP today underpins the operation of the Internet.

6.0 TUTOR-MARKED ASSIGNMENTS

1 List the major disadvantages with the layered approach to protocols.

2a. Why are protocols needed?

b Why are standards needed?

3. How does the protocol travel through the OSI model?

4. What does OSI stand for and what do we use it for?

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

3. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley

4. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

MODULE 2: SYSTEM COMPONENTS AND MANAGEMENT

UNIT 1: SYTSTEM COMPONENTS

UNIT 2: NETWORKED COMMUNITIES

UNIT 3: HOST MANAGEMENT

UNIT 4: USER MANAGEMENT

UNIT 1: SYSTEM COMPONENTS

1.0 INTRODUCTION

In this unit we assemble the components of a human–computer community, so as to prepare the way for a discussion of their management.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- Understand what is 'the system'?
- Discuss the three main components of human-computer system
- Have a basic appreciation of the frailties and procedures surrounding hardware
- Acquire a basic understanding of the principles of file systems
- Know what operating systems are and their functions
- Compare Unix and Windows operating systems
- Acquire knowledge about classes of networks
- Learn more about protocols and encapsulation
- Know about IP addressing scheme

3.0 MAIN CONTENT

3.1 What is 'the system'?

In system administration, the word *system* is used to refer both to the operating system of a computer and often, collectively the set of all computers that cooperate in a network. If we look at computer systems analytically, we would speak more precisely about *human–computer systems*:

Definition (human–computer system). *An organized collaboration between humans and computers to solve a problem or provide a service. Although computers are deterministic, humans are non-deterministic, so human–computer systems are non-deterministic.*

For the machine part, one speaks of operating systems that govern the operation of computers The term *operating system* has no rigorously accepted definition. Today, it is often thought of as the collection of all programs bundled with a computer, combining both in a kernel of basic services and utilities for users; some prefer to use the term more restrictively.

3.1.1 Network infrastructure

There are three main components in a human–computer system (see figure 1):

• *Humans*: who use and run the fixed infrastructure, and cause most problems.

• *Host computers*: computer devices that run software. These might be in a fixed location or mobile devices.

• *Network hardware*: This covers a variety of specialized devices including the following key components:

– dedicated computing devices that direct traffic around the Internet.

Routers talk at the IP address level, or 'layer 3' of the OSI model described later, simplistically speaking.

– *Switches*: fixed hardware devices that direct traffic around local area networks. Switches talk at the level of Ethernet or 'layer 2' protocols, in common parlance.

– *Cables*: There are many types of cable that interconnect devices: fiber optic cables, twisted pair cables, null-modem cables etc.

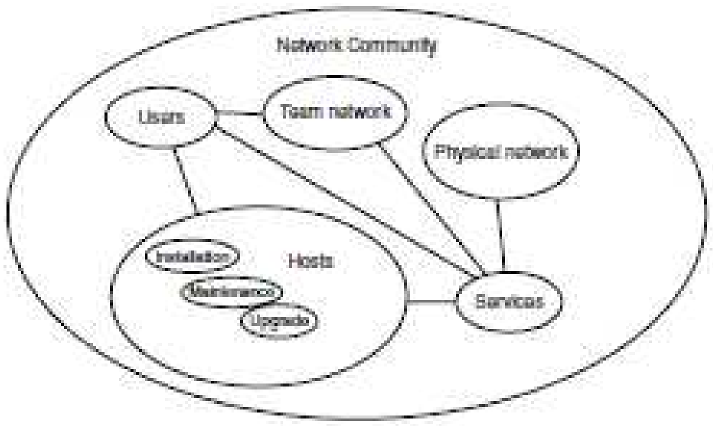


Figure 1: Some of the key dependencies in system administration. The sum of these elements forms a networked community, bound by human ties and cable ties. Services depend on a physical network, on hosts and users, both as consumers of the resources and as teams of administrators that maintain them.

3.1.2 Computers

All contemporary computers in common use are based on the Eckert–Mauchly–von Neumann architecture, sketched in figure 2. Each computer has a clock which drives a *central processor unit* (CPU), a *random access memory* (RAM) and an array of other devices, such as disk drives. In order to make these parts work together, the CPU is designed to run programs which can read and write to hardware devices. The most important program is the *operating system kernel*. On top of this are software layers that provide working abstractions for programmers and users. These consist of files, processes and services. Part of 'the system' refers to the network devices that carry messages from computer to computer, including the cables themselves. Finally, the system refers to all of these parts and levels working together.

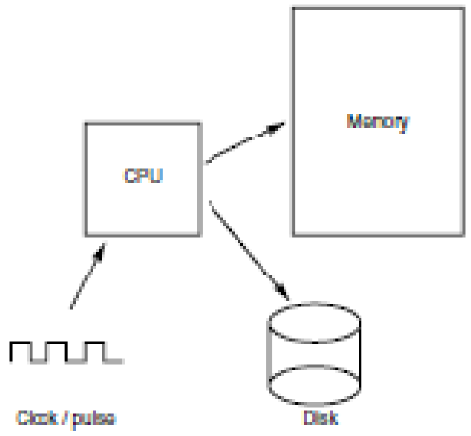


Figure 2: The basic elements of the von Neumann architecture.

3.2 Handling hardware

To be a system administrator it is important to have a basic appreciation of the frailties and procedures surrounding hardware. In our increasingly virtual world of films and computer simulations, basic common-sense facts about the laws of physics are becoming less and less familiar to us, and people treat fragile equipment with an almost casual disregard.

All electronic equipment should be treated as highly fragile and easily damaged, regardless of how sturdy it is. Today we are far too biased towards electronic equipment.

- Never insert or remove power cords from equipment without ensuring that it is switched off.
- Take care when inserting multi-pin connectors that the pins are oriented the right way up and that no pins are bent on insertion.

Moreover:

- *Read instructions:* When dealing with hardware, one should always look for and *read* instructions in a manual. It is foolish to make assumptions about expensive purchases. Instructions are there for a reason.

• *Interfaces and connectors:* Hardware is often connected to an interface by a cable or connector. Obtaining the correct cable is of vital importance.

Many manufacturers use cables which look similar, superficially, but which actually are different. An incorrect cable can result in damage to an interface.

Modem cables in particular can damage a computer or modem if they are incorrectly wired, since some computers supply power through these cables which can damage equipment that does not expect to find a power supply coming across the cable.

• *Handling components:* Modern day CMOS chips work at low voltages (typically 5 volts or lower). Standing on the floor with insulating shoes, you can pick up a static electric charge of several thousand volts. Such a charge can instantly destroy computer chips. Before touching any computer components, earth yourself by touching the metal casing of the computer. If you are installing equipment inside a computer, wear a conductive wrist strap. Avoid wearing rubber sandals or shoes that insulate you from Earth when dealing with open-case equipment, since these cause the body to build up charge that can discharge through that equipment; on the other hand it is a good idea to wear rubber soles when working around high voltage or current sources.

• *Disks:* Disk technology has been improving steadily for two decades. The most common disk types, in the workplace, fall into two families: ATA (formerly IDE) and SCSI. The original IDE (Integrated Drive Electronics) and SCSI (Small Computer Software Interface) had properties that have since evolved faster than the prejudices about them. ATA disks are now generally cheaper than SCSI disks (due to volume sales) and excel at sequential access, but SCSI disks have traditionally been more efficient at handling multiple accesses due to a multitasking bus design, and are therefore better in multitasking systems, where random access is important. However, filesystem design also plays an important role in determining the perceived performance of each; i.e. how operating systems utilize buses during updates is at least as important as bus performance itself. Interesting comparisons show that IDE technology has caught up with the head start that SCSI disks once had [322] for many purposes, but not all. SCSI comes in several varieties: SCSI 1, SCSI 2, wide SCSI, fast-wide etc. The difference has to do with the width of the data-bus and the number of disks which can be attached to each controller. There are presently three

SCSI standards: SCSI-1, SCSI-2 and SCSI-3. The SCSI-2 standard defines also wide, fast and fast/wide SCSI. Each SCSI disk has its own address (or number) which must be set by changing a setting on the disk-cabinet or by changing jumper settings inside the cabinet. Newer disks have programmable identities. Disk chain buses must be terminated with a proper terminating connector. Newer disks often contain automatic termination mechanisms integrated into the hardware. The devices on the SCSI bus talk to the computer through a controller. On modern PCs the SCSI controller is usually connected to the PCI bus either as an on-board solution on motherboards or as a separate card in a PCI slot. Other buses are also used as the carrier of the SCSI protocol, like FireWire (IEEE 1394) and USB. The SCSI standard also supports removable media devices (CD-ROM, CD-R, Zip drives), video frame grabbers, scanners and tape streamers (DAT, DLT).

• *Memory:* Memory chips are sold on small pluggable boards. They are sold in different sizes and with different speeds. A computer has a number of slots where they can be installed. When buying and installing RAM, remember

– The physical size of memory plug-in is important. Not all of them fit into all sockets.

– Memory is sold in units with different capacities and data rates. One must find out what size can be used in a system. In many cases one may not mix different types.

– There are various incompatible kinds of RAM that work in different ways.

Error correcting RAM, for instance, is tolerant to error from external noise sources like cosmic rays and other ultra short wave disturbances. It is recommended for important servers, where stability is paramount.

– On some computers one must fill up RAM slots in a particular order, otherwise the system will not be able to find them.

Another aspect of hardware is the extent to which weather and environment are important for operation.

• *Lightning:* strikes can destroy fragile equipment. No fuse will protect hardware from a lightning strike. Transistors and CMOS chips burn out much faster than any fuse. Electronic spike protectors can help here, but nothing will protect against a direct strike.

• *Power:* failure can cause disk damage and loss of data. A UPS (uninterruptible power supply) can help.

• *Heat:* Blazing summer heat or a poorly placed heater can cause systems to overheat and suddenly black out. One should not let the ambient temperature near a computer rise much above 25 degrees Centigrade. Clearly some equipment can tolerate heat better than other equipment. Bear in mind that metals expand significantly, so moving parts like disks will be worst affected by heat. Increased temperature also increases noise levels that can reduce network capacities by a fraction of a percent. While this might not sound like much, a fraction of a percent of a Giga-bit cable is a lot of capacity. Heat can cause RAM to operate unpredictably and disks to misread/miswrite.

Good ventilation is essential for computers and screens to avoid electrical faults.

• *Cold:* Sudden changes from hot to cold are just as bad. They can cause unpredictable changes in electrical properties of chips and cause systems to crash. In the long term, these changes could lead to cracks in the circuit boards and irreparable chip damage.

• *Humidity:* In times of very cold weather and very dry heat, the humidity falls to very low levels. At these times, the amount of static electricity builds up to quite high levels without dissipating. This can be a risk to electronic circuitry. Humans pick up charge just by walking around, which can destroy fragile circuitry. Paper sticks together causing paper crashes in laser printers. Too much humidity can lead to condensation and short circuits.

3.3 Operating systems

An operating system has a number of key elements: (i) a *technical layer of software* for driving the hardware of the computer, like disk drives, the keyboard and the screen; (ii) a *filesystem* which provides a way of organizing files logically, and (iii) a simple *user interface* which enables users to run their own programs and to manipulate their files in a simple way.

Of central importance to an operating system is a core software system or *kernel* which is responsible for allocating and sharing the resources of the system between several running programs or *processes*. It is supplemented by a number of supporting *services* (paging, RPC, FTP, WWW etc.) which either assist the kernel or extend its resource sharing to the network domain. The operating system can be responsible for sharing the resources of a single computer, but increasingly we are seeing *distributed operating systems* in which execution of programs and sharing of resources happens without regard for hardware boundaries; or *network operating systems* in which a central server adds functionality to relatively dumb workstations. Sometimes programs which do not affect the job of sharing resources are called *user programs*.

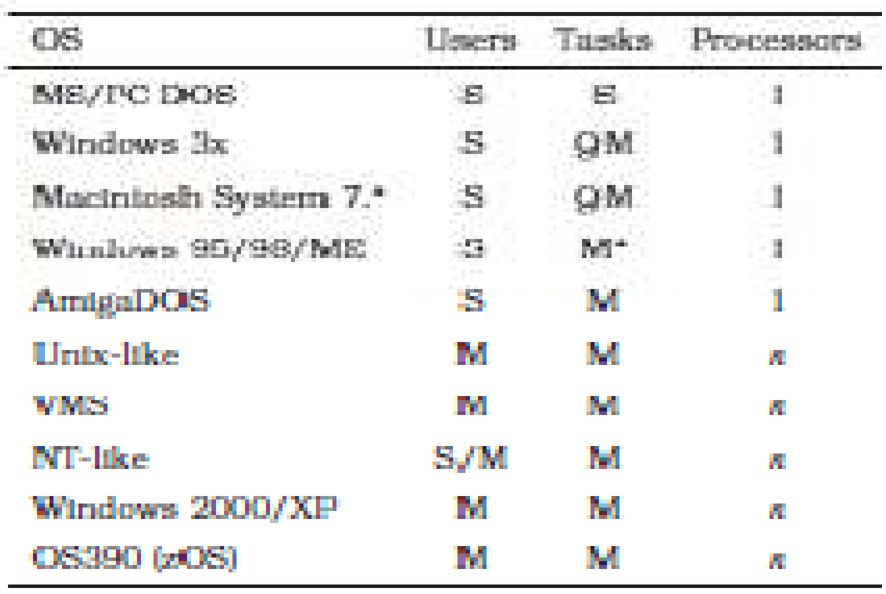
In short, a computer system is composed of many subsystems, some of which are *software systems* and some of which are *hardware systems*. The operating system runs interactive programs for humans, *services* for local and distributed users and support programs which work together to provide the infrastructure which enables machine resources to be shared between many processes. Some operating systems also provide text editors, compilers, debuggers and a variety of other tools. Since the operating system (OS) is in charge of a computer, all requests to use its resources and devices need to go through the OS kernel. An OS therefore provides *legal entry points* into its code for performing basic operations like writing to devices.

For an operating system to be managed consistently it has to be possible to prevent its destruction by restricting the privileges of its users. Different operating systems vary in their provisions for restricting privilege. In operating systems where any user can change any file, there is little or no possibility of gaining true control over the system. Any accident or whim on the part of a user can make uncontrollable changes.

Today it important to distinguish between a user interface and an operating system. A windowing system is a *graphical user interface* (GUI); an operating system shares resources and provides functionality. This issue has been confused by the arrival of the operating systems collectively called Windows, which include a graphical user interface. In principle, an operating system can have any number of different windowing interfaces, one for every taste.

Operating systems may be classified both by how many tasks they can perform ‘simultaneously’ and by how many users can be using the system ‘simultaneously’.

That is: *single-user* or *multi-user* and *single-tasking* or *multitasking*. A multi-user system must clearly be multitasking. The table below shows some examples.



OS	Users	Tasks	Processors
MS/PC DOS	S	S	1
Windows 3x	S	QM	1
Macintosh System 7.*	S	QM	1
Windows 95/98/ME	S	M*	1
AmigaDOS	S	M	1
Unix-like	M	M	*n*
VMS	M	M	*n*
NT-like	S/M	M	*n*
Windows 2000/XP	M	M	*n*
OS390 (zOS)	M	M	*n*

The first of these (MS/PC DOS/Windows 3x) are single-user, single-task systems which build on a ROM-based library of basic input–output functions called the BIOS. Windows also includes a windowing library. These are system calls which write to the screen or to disk etc. Although all the operating systems can service *interrupts*, and therefore simulate the appearance of multitasking in some situations, the DOS environment cannot be thought of as a multitasking system in any sense. Only a single user application can be open at any time. Note that

Windows 3x is not really a separate operating system from DOS; it is a user interface to DOS.

The Macintosh System 7 could be classified as single-user quasi-multitasking (QM). Apple's new Mac OS X has a Unix-like emulator running on top of a Mach kernel. That means that it is possible to run several user applications simultaneously.

A window manager can simulate the appearance of several programs running simultaneously, but this relies on each program obeying specific rules in order to achieve the illusion. Prior to Mac OS X, the MacIntosh was not a true multitasking system; if one program crashed, the whole system would crash. Similarly,

Windows 9x purported to be pre-emptive multitasking but many program crashes would also crash the entire system.

Windows NT is now a family of operating systems from Microsoft (including Windows 2000 and XP), based, in part, on the old VAX/VMS kernel from the Digital Equipment Corporation and the Windows 32 API. It has virtual memory and multi-threaded support for several processors. NT has a built-in object model and security framework which is amongst the most modern in use. Windows NT has been reincarnated now in the guise of Windows 2000 and XP, which adopt many of the successful features of the Novell system, such as consistent directory services.

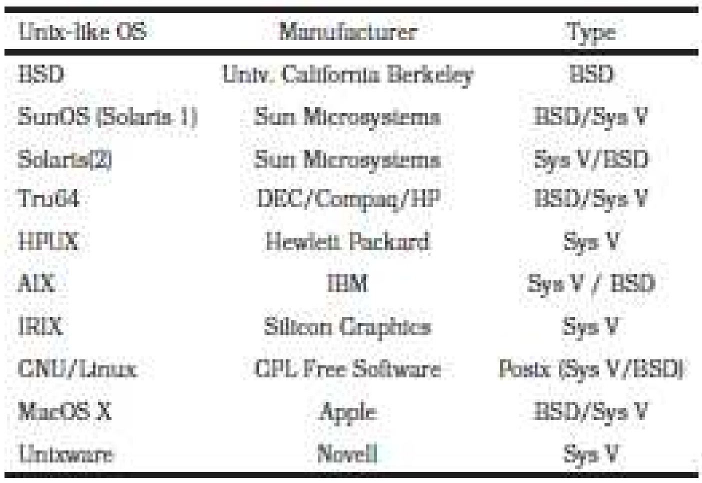
Later versions of Windows NT and Windows 2000 (a security and kernel enhanced version of NT) allow true multitasking and multiple logins also through a terminal server. Windows 2000 thus has comparable functionality to Unix in this respect.

IBM S/370, S/390 mainframe and AS/400 mini-computers are widely used in banks and large concerns for high level processing. These are fully multitasking systems of high calibre, supporting virtual machine architectures. These mainframe computers are now referred to as the IBM z-series computers, and the operating system is z/OS. Z/OS has a virtual hosting manager that can support multiple concurrent operating systems. Z-series computers have enjoyed a revival with the advent of GNU/Linux. IBM has reported running many thousands of concurrent Linux virtual kernels on their mainframe computers.

Unix is arguably the most important operating system today, both for its widespread use and its historical importance. We shall frequently refer to Unix-like operating systems below. 'Unix' (insofar as it is correct to call it that now) comes in many forms, developed by different manufacturers and enthusiasts.

Originally designed at AT&T, Unix split into two camps early on: BSD (Berkeley Software Distribution) and System V (or System 5) (AT&T license). The BSD version was developed as a research project at the University of California Berkeley (UCB). Many of the networking and user-friendly features originate from these modifications. With time, these two versions have been merged back together and most systems are now a mixture of both worlds. Historically BSD Unix has been most prevalent in universities, while System 5 has been dominant in business environments. In the 1990s Sun Microsystems and Hewlett Packard started a move towards System V, keeping only the most important features of the BSD system, but later

suppressed the visible System V aspects in favor of BSD again. Today, the differences are few, thanks to a de-facto standardization. A standardization committee for Unix called POSIX, formed by the major vendors and independent user groups, has done much to bring compatibility to the Unix world. Here are some common versions of Unix.



Unix-like OS	Manufacturer	Type
BSD	Univ. California Berkeley	BSD
SunOS (Solaris 1)	Sun Microsystems	BSD/Sys V
Solaris(2)	Sun Microsystems	Sys V/BSD
Tru64	DEC/Compaq/HP	BSD/Sys V
HPUX	Hewlett Packard	Sys V
AIX	IBM	Sys V / BSD
IRIX	Silicon Graphics	Sys V
GNU/Linux	GPL Free Software	Posix (Sys V/BSD)
MacOS X	Apple	BSD/Sys V
Unixware	Novell	Sys V

Note that multiple mergers have now stirred this mixture: Ultrix, OSF/1 and Digital Unix were products of DEC before the Compaq merger, Tru64 was what Compaq renamed Digital Unix after the merger, and now it is called HP Tru64 Unix.

The original BSD source code is now available to the public and the GNU/Linux source code is free (and open source) software. Unix is one of the most portable operating systems available today. It runs on everything from palm-computers to supercomputers. It is particularly good at managing large database applications and can run on systems with hundreds of processors.

Most Unix-like operating systems support symmetric multi-threaded processing and all support simultaneous logins by multiple users.

3.3.1 The legacy of insecure operating systems

The home computer revolution was an important development which spread cheap computing power to a large part of the world. As with all rapid commercial developments, the focus in developing home operating systems was on immediate functionality, not on planning for the future. The home computer revolution preceded the network revolution by a number of years and home computer operating systems did not address security issues. Operating systems developed during this period include Windows, MacIntosh, DOS, Amiga-DOS. All of these systems are completely insecure: they place *no limits* on what a determined user can do.

Fortunately these systems will slowly be replaced by operating systems which were designed with resource sharing (including networking) in mind. Still, there is a large number of insecure computers in use and many of them are now connected to networks. This should be a major concern for a system administrator. In an age where one is forced to take security extremely seriously, leaving insecure systems where they can be accessed physically or by the network is a potentially dangerous situation. Such machines should not be allowed to hold important data and they should not be allowed any privileged access to network services. We shall return to this issue in the chapter on security.

3.3.2 Securable operating systems

To distinguish them from insecure operating systems we shall refer to operating systems like Unix and NT as *securable* operating systems. This should not give the impression that Unix and NT are secure: by its nature, security is not an achievable goal, but an aspiration that includes accepted levels of risk. Nevertheless, these operating systems do have the mechanisms which make a basic level of preventative security possible.

A fundamental prerequisite for security is the ability to restrict access to certain system resources. The main reason why DOS, Windows 9x and the MacIntosh are so susceptible to virus attacks is because any user can change the operating system's files. Properly configured and bug-free Unix/NT systems are theoretically immune to such attacks, if privilege is not abused, because ordinary users do not have the privileges required to change system files. Unfortunately the key phrases *properly configured* and *bug-free* highlight the flaw in this dream.

In order to restrict access to the system we require a notion of *ownership* and *permission*. Ordinary users should not have access to the hardware devices of a secure operating system's files, only their own files, for then they will not be able do anything to compromise the security of the system. System administrators need access to the whole system in order to watch over it, make backups and keep it running. Secure operating systems thus need a privileged account which can be used by the system administrator when he/she is required to make changes to the system.

3.3.3 Shells or command interpreters

Today it is common for operating systems to provide graphical window systems for all kinds of tasks. These are often poorly suited to system administration because they only allow us to choose between pre-programmed operations which the program designers foresaw when they wrote the program. Most operating systems provide an alternative command line user interface which has some form of interpreted language, thus allowing users to express what they want with more freedom and precision. Windows proprietary shells are rudimentary; Unix shells are rich in complexity and some of them are available for installation on Windows.

Shells can be used to write simple programs called *scripts* or *batch files* which often simplify repetitive administrative tasks.

3.3.4 Privileged accounts

Operating systems that restrict user privileges need an account which can be used to configure and maintain the system. Such an account must have access to the whole system, without regard for restrictions. It is therefore called a privileged account.

In Unix the privileged account is called *root*, also referred to colloquially as the super-user. In Windows, the *Administrator* account is similar to Unix's root, except that the administrator does not have automatic access to everything as does root. Instead he/she must be first granted access to an object. However the Administrator always has the right to grant them self access to a resource so in practice this feature just adds an extra level of caution. These accounts place virtually no restriction on what the account holder can do. In a sense, they provide the privileged user with a skeleton key, a universal pass to any part of the system.

Administrator and root accounts should never be used for normal work: they wield far too much power. This is one of the hardest things to drill into novices, particularly those who have grown up using insecure operating systems. Such users are used to being able to do whatever they please. To use the privileged account as a normal user account would be to make the systems as insecure as the insecure systems we have mentioned above.

Principle (Minimum privilege). *Restriction of unnecessary privilege protects a system from accidental and malicious damage, infection by viruses and prevents users from concealing their actions with false identities. It is desirable to restrict users' privileges for the greater good of everyone on the network.*

Inexperienced users sometimes aspire to gain administrator/root privileges as a mark of status. This can generate the myth that the purpose of this account is to gain power over others. In fact the opposite is true: privileged accounts exist precisely because one does *not* want to have too much power, except in exceptional circumstances. The corollary to our principle is this:

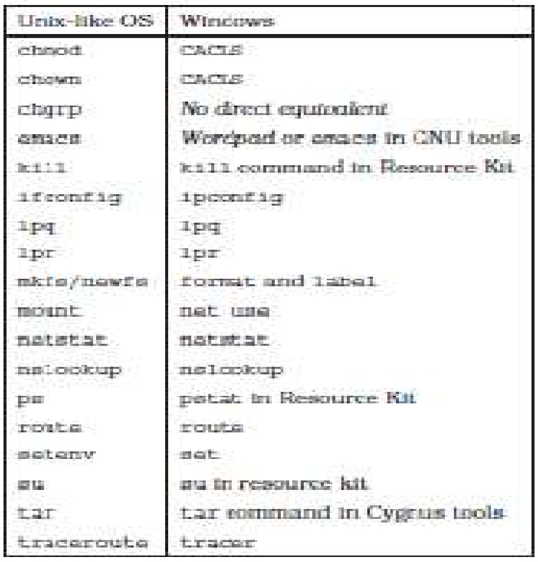
Corollary to principle (Minimum privilege). *No one should use a privileged root or Administrator account as a user account. To do so is to place the system in jeopardy. Privilege should be exercised only when absolutely necessary.*

One of the major threats to Internet security has been the fact that everyone can now be root/Administrator on their own host. Many security mechanisms associated with trusted ports, TCP/IP spoofing etc. are now broken, since all of the security of these systems lies in the outdated assumption that ordinary users will not have privileged access to network hardware and the kernel.

3.3.5 Comparing Unix-like and Windows computers

The two most popular classes of operating system today are Unix-like operating systems (i.e. those which are either derived from or inspired by System V or BSD) and Microsoft Windows NT-like operating systems. We shall only discuss Windows NT and later derivatives of the Windows family, in a network context. For the sake of placing the generalities in this material in a clearer context, it is useful to compare 'Unix' with Windows.

The file and directory structures of Unix and Windows are rather different, but it is natural that both systems have the same basic elements.



Unix-like OS	Windows
chmod	CACLS
chown	CACLS
chgrp	No direct equivalent
emacs	Wordpad or emacs in GNU tools
kill	kill command in Resource Kit
ifconfig	ipconfig
lpq	lpq
lpr	lpr
mkfs/newfs	format and label
mount	net use
netstat	netstat
nslookup	nslookup
ps	pstat in Resource Kit
route	route
setenv	set
su	su in resource kit
tar	tar command in Cygnus tools
traceroute	tracert

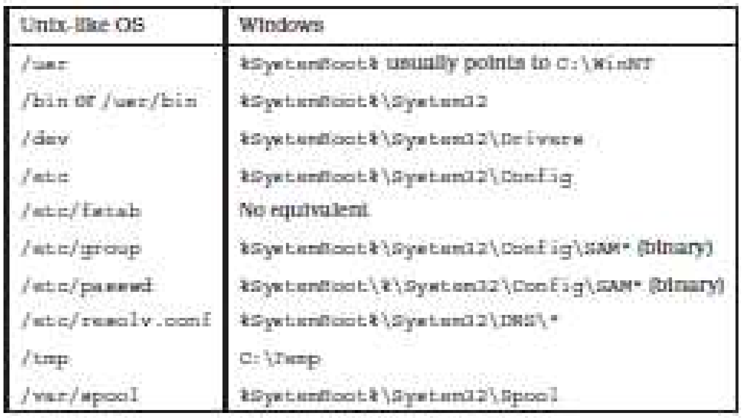
Table.1: Comparison of Unix and Windows shell commands.

Unix-like operating systems are many and varied, but they are basically similar in concept. It is not the purpose of this material to catalogue the complete zoological inventory of the 'Unix' world; our aim is to speak primarily of generalities which rise above such distinctions. Nonetheless, we shall occasionally need to distinguish the special features of these operating systems, and at least distinguish them from Windows. This should not detract from the fact that Windows has adopted much from the Unix cultural heritage, even though superficial attempts to hide this (e.g. renaming / with \ in filenames, changing the names of some commands etc.) might obscure the fact.

Windows NT, 2000, XP are multitasking operating systems from Microsoft which allow users to log in to a console or *workstation*. The consoles may be joined together in a network with common resources shared by an NT *domain*. An NT host is either a network server or a personal workstation. The basic Windows distribution contains only a few tools which can be used for network administration. The Resource Kit is an extra package of documentation and unsupported software which nonetheless provides many essential tools. Other tools can be obtained free of charge from the network.

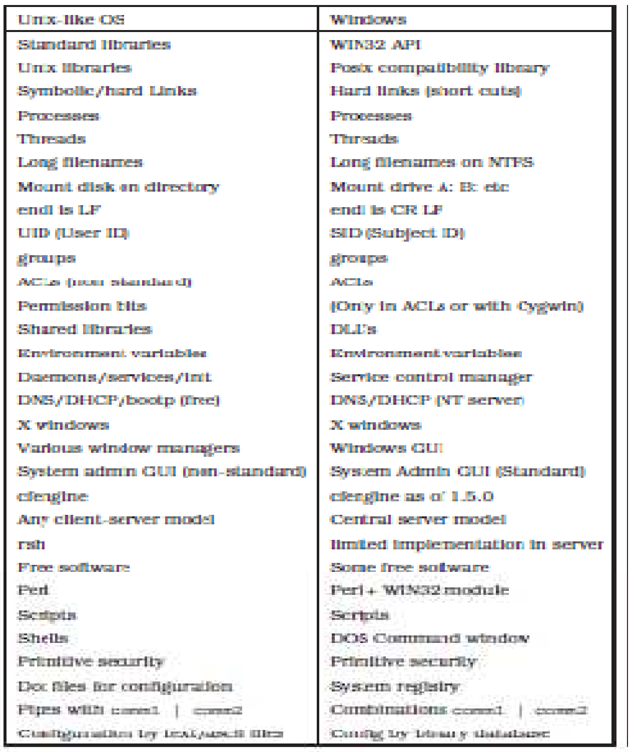
Windows did not have a remote shell login feature like Unix at the outset. One may now obtain a Terminal Server which gives Windows telnet-like functionality. This adds an important possibility: that of direct remote administration. The Free Perl Win32 package and related tools provides tools for solving a number of problems with NT from a script viewpoint.

Although we are ignoring many important operating systems by comparing just two main players, a comparison of Unix-like operating systems with NT covers most of the important differences. The latest offerings from the MacIntosh world, for instance, are based on emulation of BSD 4.4 Unix and MacOS on a Mach kernel, with features designed to compete with NT. IBM’s z-series operating



Unix-like OS	Windows
/usr	%SystemRoot% usually points to C:\WinNT
/bin or /usr/bin	%SystemRoot%\System32
/dev	%SystemRoot%\System32\Drivers
/etc	%SystemRoot%\System32\Config
/etc/fstab	No equivalent
/etc/group	%SystemRoot%\System32\Config\SAM* (binary)
/etc/passwd	%SystemRoot\%\System32\Config\SAM* (binary)
/etc/resolv.conf	%SystemRoot%\System32\DNS\*
/tmp	C:\Temp
/var/spool	%SystemRoot%\System32\Spool

Table 2: Comparison of Unix and Windows directories and files.



Unix-like OS	Windows
Standard libraries	WIN32 API
Unix libraries	Posix compatibility library
Symbolic/hard Links	Hard links (short cuts)
Processes	Processes
Threads	Threads
Long filenames	Long filenames on NTFS
Mount disk on directory	Mount drive A: B: etc
endl is LF	endl is CR LF
UID (User ID)	SID (Subject ID)
groups	groups
ACLs (non standard)	ACLs
Permission bits	(Only in ACLs or with Cygwin)
Shared libraries	DLL's
Environment variables	Environment variables
Daemons/services/init	Service control manager
DNS/DHCP/bootp (free)	DNS/DHCP (NT server)
X windows	X windows
Various window managers	Windows GUI
System admin GUI (non-standard)	System Admin GUI (Standard)
cfengine	cfengine as of 1.5.0
Any client-server model	Central server model
rsh	limited implementation in server
Free software	Some free software
Perl	Perl + WIN32 module
Scripts	Scripts
Shells	DOS Command window
Primitive security	Primitive security
Dot files for configuration	System registry
Pipes with comm1 | comm2	Combinations comm1 | comm2
Configuration by text/ascii files	Config by binary database

Table 3: Comparison of Unix and Windows software concepts.

System for mainframes has experienced a revival of interest since the GNU/Linux system was ported to run on its virtual engine.

Unix is important, not only for its endurance as the sturdy workhorse of the network, but also for its cultural significance. It has influenced so many other operating systems (including Windows) that further comparisons would be largely redundant. Let us note briefly then, for the record, the basic correspondences between Unix-like operating systems and Windows. Many basic commands are very similar. Tables 1,2 and 3 give some comparisons between Unix and Windows concepts.

Note: there are differences in nomenclature. What Windows refers to as pipes in its internal documentation is not what Unix refers to as pipes in its internal documentation.

A major problem for Windows has been the need for compatibility with DOS, through Windows 9x to NT. Since both DOS and Windows 9x are insecurable systems, this has led to conflicts of interest. Unix vendors have tried to keep step with Microsoft's impressive user interface work, in spite of the poor public image of Unix (often the result of private dominance wars between different Unix vendors) but the specially designed hardware platforms built by Unix vendors have had a hard time competing with inferior but cheaper technology from the PC world.

3.4 Filesystems

Files and filesystems are at the very heart of what system administration is about. Almost every task in host administration or network configuration involves making changes to files. We need to acquire a basic understanding of the principles of filesystems, so what better way than to examine some of the most important filesystems in use today. Specifically what we are interested in is the user interfaces to common filesystems, not the technical details which are rather fickle. We could, for instance, mention the fact that old filesystems were only 32 bit addressable and therefore supported a maximum partition size of 2GB or 4GB, depending on their implementation details, or that newer filesystems are 64 bit addressable and therefore have essentially no storage limits. We could mention the fact that Unix uses an index node system of block addressing, while DOS uses a tabular lookup system: the list goes on. These technical details are of only passing interest since they change at an alarming pace. What is more constant is the user functionality of the filesystems: how they allow file access to be restricted to groups of users, and what commands are necessary to manage this.

3.4.1 Unix file model

Unix has a hierarchical filesystem, which makes use of directories and subdirectories to form a tree. All filesystems on Unix-like operating systems are based on a system of *index nodes*, or *inodes*, in which every file has an index entry stored in a special part of the filesystem. The inodes contain an extensible system of pointers to the actual disk blocks which are associated with the file. The inode contains essential information needed to locate a file on the disk.

The top or start of the Unix file tree is called the root filesystem or '/'. Although the details of where common files are located differ for different versions of Unix, some basic features are the same.

The file hierarchy

The main subdirectories of the root directory together with the most important file are shown below. Their contents are as follows:

- /bin Executable (binary) programs. On most systems this is a separate directory to /usr/bin. In SunOS, this is a pointer (link) to /usr/bin.
- /etc Miscellaneous programs and configuration files. This directory has become very messy over the history of Unix and has become a dumping ground for almost anything. Recent versions of Unix have begun to tidy up this directory by creating subdirectories /etc/mail, /etc/inet etc.
- /usr This contains the main meat of Unix. This is where application software

  lives, together with all of the basic libraries used by the OS.
- /usr/bin More executables from the OS.
- /usr/sbin Executables that are mainly of interest to system administrators.
- /usr/local This is where users' custom software is normally added.
- /sbin A special area for (often statically linked) system binaries. They are placed here to distinguish commands used solely by the system administrator from user commands, and so that they lie on the system root partition, where they are guaranteed to be accessible during booting.
- /sys This holds the configuration data which go to build the system kernel.

  (See below.)
- /export Network servers only use this. This contains the disk space set aside for client machines which do not have their own disks. It is like a 'virtual disk' for diskless clients.
- /dev and /devices A place where all the 'logical devices' are collected. These are called 'device nodes' in Unix and are created by mknod. Logical devices are Unix's official entry points for writing to devices. For instance, /dev/console is a route to the system console, while /dev/kmem is a route for reading kernel memory. Device nodes enable devices to be treated as though they were files.
- /home (Called */users* on some systems.) Each user has a separate login directory where files can be kept. These are normally stored under /home by some convention decided by the system administrator.
- /root On newer Unix-like systems, root has been given a home-directory which is no longer the root of the filesystem '/'. The name root then loses its logic.
- /var System V and mixed systems have a separate directory for spooling.

  Under old BSD systems, /usr/spool contains spool queues and system data.

  /var/spool and /var/adm etc. are used for holding queues and system log files.

Every Unix directory contains two 'virtual' directories marked by a single dot and two dots.

```
Ls  -a
.    ..
```

The single dot represents the directory one is already in (the current directory).

The double dots mean the directory one level up the tree from the current location.

Thus, if one writes

```
Cd  /usr/share
Cd  ..
```

the final directory is /usr. The single dot is very useful in C programming if one wishes to read 'the current directory'. Since this is always called '.' there is no need to keep track of what the current directory really is. '.' and '..' are hard links to the current and parent directories, respectively.

3.4.2 Windows file model

The Windows operating system supports a variety of legacy filesystems for backward compatibility with DOS and Windows 9x. These older filesystems are insecure, in the sense that they have no mechanisms for restricting access to files. The filesystem NTFS was introduced with NT in order to solve this problem.

The filesystem has gone through a number of revisions and no doubt will go through many more before it reaches constancy. NTFS, like the Unix filesystem, is a hierarchical filesystem with files and directories. Each file or directory has an owner, but no group membership. Files do not have a set of default permission bits, as does Unix; instead they all have full-blooded The NTFS filesystem is indexed by a master file table, which serves an analogous function to Unix's inodes, though the details are somewhat different.

Filesystem layout

Drawing on its DOS legacy, Windows treats different disk partitions as independent floppy disks, labeled by a letter of the alphabet:

A: B: C: D: ...

For historical reasons, drive A: is normally the diskette station, while drive C: is the primary hard disk partition. Other drive names are assigned at random, but often H: is reserved for partitions containing users' home directories. Unlike Unix, different devices are not sewn seamlessly into a unified file tree, though this will probably change in a future release of Windows. Originally, DOS chose to deviate from its Unix heritage by changing the subdirectory separator from / to \.

Moreover, since each device is treated as a separate entity, there is a root directory on every disk partition:

A: B: C: ... and one has a notion of *current working drive*, as well as *current working directory*.

These distinctions often cause confusion amongst users who work with both Unix and Windows.

The layout of the Windows filesystem has changed through the different versions, in an effort to improve the structure. This description relates to NT 4.0. The system root is usually stored in C:\WinNT and is generally referred to by the system environment variable %System Root%.

- C:\I386 This directory contains binary code and data for the Windows operating system. This should normally be left alone.
- C:\Program Files This is Windows's official location for new software.

  Program packages which you buy should install themselves in subdirectories of this directory. More often than not they choose their own locations, however, often with a distressing lack of discipline.
- C:\Temp Temporary scratch space, like Unix's /tmp.
- C:\WinNT This is the root directory for the Windows system. This is mainly for operating system files, so you should not place new files under this directory yourself unless you really know what you are doing. Some software packages might install themselves here.
- C:\WinNT\config Configuration information for programs. These are generally

  binary files so the contents of Windows configuration files is not very

  interesting.
- C:\WinNT\system32 This is the so-called system root. This is where most system applications and data-files are kept.

3.4.3 Network filesystem models

Unix and Windows have two of the most prevalent filesystem interfaces, apart from DOS itself (which has only a trivial interface), but they are both stunted in their development. In recent years, filesystem designers have returned to an old idea which dates back to a project from Newcastle University, called the Newcastle Connection, an experimental distributed filesystem which could link together many computers seamlessly into a single file tree. To walk around the disk resources of the entire network, one simply used cd to change directory within a global file tree.

This idea of distributed filesystems was partially adopted by Sun Microsystems in developing their Network Filesystem (NFS) for Unix-like operating systems.

This is a distributed filesystem, for mainly local area networks. The use of open standards and a willingness to allow other vendors to use the technology quickly made NFS a de-facto standard in the Unix world, overtaking alternatives like RFS. However, owing to vendor disagreement, the Network Filesystem has been limited to the lowest common denominator Unix filesystem-model. Vendor-specific improvements are available, but these do not work in a heterogeneous environment and thus NFS is relatively featureless, by comparison with the functionality available on local disk filesystems. In spite of this, there is no denying that NFS has been very effective, as is testified by the huge number of sites which use it unconditionally.

3.4.4 Unix and Windows sharing

Filesystems can be shared across a network by any of the methods we have discussed above. We can briefly note here the correspondence of commands and methods for achieving network sharing.

With AFS and DCE/DFS, used mainly on Unix-like hosts, the security model is such that a computer becomes part of a cell or domain. Within such a cell, disk partitions are referred to as volumes. These can be replicated and shared with other computers. AFS cells on other server hosts can be attached to client hosts using the afsd program. A local cell can be published to the rest of the AFS speaking network by adding its attributes to a database. The resulting seamless file tree is visible under /afs. The visibility of files in this model is controlled by the Access Control Lists.

Unix-like hosts use NFS to share filesystems, by running the daemons (e.g. rpc.mountd and rpc.nfsd). Filesystems are made available for sharing by adding them to the file /etc/exports, on most systems, or confusingly to /etc/dfs/dfstab on SVR4 based Unix. The syntax in these files is particular to the flavor of the Unix-like operating system one is using. With some operating Systems, using /etc/exports, it is necessary to run the command exportfs –a to make the contents of the export file visible to the daemons which control access.

On SVR4 systems, like Solaris, there is a command called share for exporting filesystems, and the file /etc/dfs/dfstab is just a shell script containing a lot of share commands, e.g.

```
allhosts=nomad:vger:nomad.domain.country:vger.domain.country

 share -F nfs -o rw=$allhosts /site/server/local
```

Here the command shareall is the equivalent for exporting all filesystems in this file. It simply runs a shell script containing all such commands. The example above makes the directory tree /iu/server/local available to the hosts nomad and vger. Note that due to different name services implementations and their various behaviors, it is often necessary to use both the unqualified and fully qualified names of hosts when sharing.
On the client or receiving end, we attach a shared filesystem to a host by ‘mounting’ it. NFS filesystems are mounted in exactly the same way as they mount
a local disk, i.e. with the mount command, e.g.

```
mkdir -p /site/server/local
mount server:/site/server/local /site/server/local
```

Here we create a directory on which to mount a foreign filesystem and then mount it on a directory which has the same name as the original on the server. The original name and the new name do not have to be the same, but there is a point to this which we shall return to later. Assuming that the server-host granted us the right to mount the filesystem on our host, we now have access to the remote filesystem, as though it were a local disk. The only exception is the superuser root, who is granted the access rights of a user called nobody. The point of this is that the administrator on the client host is not necessarily the administrator on the server host, and has no obvious right to every user's files there. This mapping can be overridden if convenience outweighs the minor security it adds.

Windows filesystems on a server are shared, either using the GUI, or by executing the command

```
net share alias=F:\filetree
```

On the client side, the file tree can then be 'mounted' by executing the command

```
 net use X: \\serverhost\alias
```

This attaches the remote file tree, referenced by the alias, to Windows drive X:. One of the logistical difficulties with the Windows drive model is that drive associations are not constant, but might change when new hardware is detected.

Drive associations can be made to persist by adding a flag

```
 net use X: \\serverhost\alias /persistent: yes
```

to the mount command. This is not a perfect solution, but it works.

3.5 Networks

The network is the largest physical appendage to our computer systems, but it is also the least conspicuous, often hidden behind walls and in locked switching rooms, or passing invisibly through us as electromagnetic radiation. To most users, the network is a piece of magic which they have abruptly learned to take for granted, and yet, without it, modern computing practices would be impossible.

A network is a number of pathways for communication between two or more hosts. Networking is increasingly important, as computers are used more and more as devices for media access rather than for computation. Networking raises issues for system management at

many levels, from its deployment to its configuration and usage. We begin here, simply, by identifying the main components involved in this important subsystem.
The most simplistic way to ensure communication between *N* hosts would be to stretch a private cable between every pair of hosts on a network. This would require a cat's cradle of *N* network interfaces and *N* − 1 cables per host, i.e. *N(N* − 1*)/*2 links in total, which would be quite unmanageable and equally expensive. The challenge of networking is therefore to provide some kind of shared cable which is attached to several hosts simultaneously by means of a single *network interface*. .

3.5.1 Review of the OSI model

The International Standards Organization (ISO) has defined a model for describing communications across a network, called the OSI model, for *Open Systems Interconnect (reference model)*. This model is a generalized abstraction of how network communication can be and is implemented. The model does not fit every network technology perfectly, but it is widely used to discuss and refer to the layers of technology involved in networking, thus we begin by recapping this model. The OSI model describes seven layers of abstraction.

Layer	Name	Example
7	Application layer	Program protocol commands
6	Presentation layer	XDR or user routines
5	Session layer	RPC / sockets
4	Transport layer	TCP or UDP
3	Network layer	IP Internet protocol
2	Data link layer	Ethernet protocol
1	Physical layer	Cables, interfaces

At the lowest level, the sending of data between two machines takes place by manipulating voltages along wires. This means we need a device driver for the signaler, and something to receive the data at the other end – a way of converting the signals into bytes; then we need a way of structuring the data so that they make sense. Each of these elements is achieved by a different level of abstraction.

1. ***Physical layer:*** This is the sending a signal along a wire, amplifying it if it gets weak, removing noise etc. If the type of cable changes (we might want to reflect signals off a satellite or use fiber optics) we need to convert one kind of signal into another. Each type of transmission might have its own accepted ways of sending data (i.e. protocols).

2. ***Data link layer:*** This is a layer of checking which makes sure that what was sent from one end of a cable to the other actually arrived. This is sometimes called *handshaking*. The Ethernet protocol is layer 2, as is Token Ring. This level is labeled by Media Access Control (MAC) addresses.

3. ***Network layer:*** This is the layer of software which recognizes structure in the network. It establishes global identity and handles the delivery of data by manipulating the physical layer. The network layer needs to know something about addresses – i.e. where the data are going, since data might flow along many cables and connections to arrive where they are going. Layer 3 is the layer at which IP addresses enter.

4. ***Transport layer:*** We shall concentrate on this layer for much of what follows.
The transport layer builds 'packets' or 'datagrams' so that the network layer knows what is data and how to get the data to their destination. Because many machines could be talking on the same network all at the same time, data are broken up into short 'bursts'. Only one machine can talk over a cable at a time so we must have *sharing*. It is easy to share if the signals are sent in short bursts. This is analogous to the sharing of CPU time by use of time-slices. TCP and UDP protocols are encoded at this layer.

5. ***Session layer:*** This is the part of a host's operating system which helps a user program to set up a connection. This is typically done with *sockets* or the RPC.

6. ***Presentation layer:*** How are the data to be sent by the sender and interpreted by the receiver, so that there is no doubt about their contents? This is the role played by the external data representation (XDR) in the RPC system.

7. ***Application layer:*** The program which wants to send data has its own protocol layer, typically a command language encoding (e.g. GET, PUT in FTP or HTTP).
These layers are not always so cleanly cut. Today, networking technologies at all levels are mixing them up: routers and switches are merging layers 2 and 3, and routers that prioritize traffic need to know what application is being transported, so that the information can be fed into layers 2 and 3 in order to provide guarantees on performance (so-called Quality of Service). As always, the advantage of using a layered structure is that we can change the details of the lower layers without having to change the higher layers. Layers 1 to 4 are those which involve the transport of data across a network. We could change all of these without doing serious damage to the upper layers – thus as new technology arrives, we can improve network communication without having to rewrite software. That is precisely what is happening with new technologies such as IPv6 and MPLS.

3.5.2 Cables and interface technologies

Different vendors have invested in different networking technologies, with different Media Access Control (MAC) specifications. Most Unix systems use some form of Ethernet interface. IBM systems have employed Token Ring networking technology very successfully for their mainframes and AS/400 systems; they now support Ethernet also on their RS/6000 systems. Now most manufacturers provide solutions for both technologies, though Ethernet is undoubtedly popular for local area networks.

• *Bus/Ethernet approach:* Ethernet technology was developed by Xerox, Intel and DEC in 1976, at the Palo Alto Research Center (PARC). In the Ethernet bus approach, every host is connected to a common cable or bus. Only one host can be using a given network cable at a given instant. It is like a conference telephone call: what goes out onto a network reaches all hosts on that network (more or less) simultaneously, so everyone has to share the line by waiting for a suitable moment to say something. Ethernet is defined in the IEEE 802.3 standard documents. An Ethernet network is available to any host at any time, provided the line isn't busy. This is called CSMA/CD, or Carrier Sense Multiple Access/Collision Detect. A collision occurs when two hosts attempt to send signals simultaneously. CSMA/CD means that if a card has something to send, it will listen until no other card is transmitting, then start transmitting and listen if no other card starts transmitting at the same time. If another card began transmitting it will stop, wait for a random interval and try again. The original Ethernet, with a capacity of 10 megabits per second, could carry packets of 1518 bytes.

Today, Ethernet is progressing in leaps and bounds. Switched Ethernet running on twisted pair cables can deliver up to 100 megabits per second (100BaseT, fast Ethernet). Gigabit Ethernets are already common. The main limitation of Ethernet networks is the presence of collisions. When many hosts are talking, performance degrades quickly due to time wasted by hosts waiting to get a word in. In order to avoid collisions, packet sizes are limited. With a large number of small packets, it is easier to share the time between more hosts. Ethernet interfaces are assigned a unique MAC address when they are built. The initial numbers of the address identify each manufacturer uniquely. Full-duplex connections at 100MB are possible with fast Ethernets on dedicated cables where collisions cannot occur.

• *Token Ring/FDDI approach:* In the token ring approach, hosts are coupled to hubs or nodes each of which has two network interfaces and the hosts are connected in a uni-directional ring. The token ring is described in IEEE 802.5. The token ring is a deterministic protocol; if Ethernet embraces chaos, then the token ring demands order. No matter when a host wishes to transmit, it must wait for a passing token, in a specified time-slot. If a signal (token) arrives, a host can append something to the signal. If nothing is appended, the token is passed on to the next host which has the opportunity to do the same. Similarly, if the signal arriving at one of the interfaces is for the host itself then it is read. If it is not intended for the host itself, the signal is forwarded to the next host where the same applies. A common token ring based interface in use today is the optical FDDI (Fiber distributed data interface). Token rings can pass 16 megabits per second, with packet sizes of 18 kilobytes. The larger packet sizes are possible since there is no risk of collisions.
Like Ethernet interfaces, token ring interfaces are manufactured with a uniquely assigned MAC address.

• *Frame Relay* is an alternative layer 2 packet-switching protocol for connecting devices on a Wide Area Network (WAN) or backbone. It is used for point-to point connections, but is capable of basic switching, like ATM, so it can create virtual point-to-point circuits, where several switches might be involved. Frame relay is popular because it is relatively inexpensive. However, it is also being replaced in some areas by faster technologies, such as ATM. Frame relay has the advantage of being widely supported, and is better suited than ATM for data-only, medium-speed (56/64 Kbps, T1): the ratio of header size to frame size is typically much smaller than the overhead ratio for ATM.

• *ATM*, Asynchronous Transfer Mode technology, is a high capacity, deterministic, transmission technology developed by telephone companies in order to exploit existing copper telephone networks. ATM is a layer 2–3 hybrid technology. ATM is believed to be able to reach much higher transfer rates than Ethernet, since it disallows collisions and is optimized for switching. Its expense, combined with the increasing performance of fast Ethernet, has made ATM most attractive for high speed Internet backbones and Wide Area Networks, though some local area networks have been implemented as proof of principle.

Even with the bus approach, any host can be connected to several independent network segments if it has a network interface for each network it is attached to. Each network interface then has a separate network address; thus a host which is connected to several networks will have a different address on each network. A device which is coupled to several networks and which forwards data from one network to another is called a *router*.

Network signals are carried by a variety of means. These days copper cables are being replaced by fiber-optic glass transmission for long-distance communication and even radio links. In local area networks it is still usually copper cables which carry the signals. These cables usually carry Ethernet protocols over twisted pair (telephone-like) cables. Twisted pair lines are sometimes referred to as 10baseT, 100baseT etc. The numbers indicate the capacity of the line, 'base' indicates that the cable is used in a *baseband* system and the 'T' stands for twisted-pair. Each host has a single cable connecting it to a *multi-way repeater* or *hub*. Fiber-optic cables (FDDI, SONET, SDH) have varying appearances.

3.5.3 Connectivity

Network cables are joined together by hardware which makes sure that messages are transmitted from cable to segment in the right direction to reach their destinations. A host which is coupled to several network segments and which forwards data from one network to another is called a *router*. Routers not only forward data but they prevent the spread of network messages which other network segments do not need to know about. This limits the number of hosts which are sharing any given cable segment, and thus limits the traffic which any given host sees. Routers can also filter unwanted traffic for security purposes. A router knows which destination addresses lie on which of the networks it is connected to and it does not let message traffic spread onto irrelevant cables.

A *bridge* is a hardware device which acts like a filter on busy networks. A bridge works like a 'mini-router' and separates two segments of the same cable. A bridge knows which incoming cables do *not* offer a destination address and prevents traffic from spreading to this part of a cable. A bridge is used to isolate traffic on busy sections of a network or conversely to splice networks together. It is a primitive kind of switch.

A *repeater* is an amplifier that strengthens the network signal over long stretches of cable. A *multi-port repeater* also called a *hub* does the same thing and also splits one cable into *N* sub-cables for convenience. Hubs are common in twisted pair networks where it is necessary to fan a cable out into a star pattern from the hub to send one cable to each host.
A *switch* is a hub which can direct a message from one host cable directly to the intended host by routing the signal directly. The advantage with this is that other machines do not have to see the traffic between two hosts. Each pair of hosts has a virtual private cable. Switched networks are not immune to spies, net-sniffing or network listening devices, but they make it more difficult for the casual browser to see traffic that does not concern them. A switch performs many of the tasks of a router and vice versa. The difference is that a switch works at layer 2 of the OSI model (i.e. with MAC addresses), whereas a router works at layer 3 (IP addresses). A switch cannot route data on a world-wide basis.

When learning about a new network one should obtain a plan of the physical setup. If we have done our homework, then we will know where all of these boxes are on the network.

Note that, while it is common to refer to routing and switching as 'layer 3' and 'layer 2' in loose parlance, sticklers for correctness will find this is somewhat ill-defined. These labels mix up the OSI model with the IP model. However, since they roughly coincide at layers 2 and 3, we can identify layer 2 as 'Ethernets' (or equivalent) and layer 3 as the IP-addressable transport layer. Modern routing/ switching equipment is not so easily placed into either of these categories, however; network junction devices typically contain modules for both types of communication.

3.5.4 LANs, WANs and VLANs

In the 1980s and 1990s, most networks consisted of a hierarchy of routers, joined into a Wide Area Network (WAN). Each Local Area Network (or local community, such as a business or university) would have its own gateway router, connecting it to the rest of the world. The purpose of a router was two-fold:

- To forward traffic meant for remote locations along a suitable route, so that it would arrive at the right address.
- To prevent purely local traffic from leaking out of the local network and causing unnecessary congestion.

When an electrical signal passes along a cable it is like a light being switched on in a room. The picture of a network transmission as a stream of bytes travelling along a cable, like cars in a train, is often misleading.4 In local area networks, the distances are often so short that transmission is almost instantaneous and each bit fills an entire cable segment; though this depends on the data rate. Every bit, every 1 or 0, is a signal (a voltage or light pulse) on a cable which fills a space, the size of a wavelength, at about two-thirds of the speed of light in a vacuum – so, on short segments, this is often the entire cable. It is like sending Morse code with a lighthouse. Every part of the network sees the signal, but only the addressed recipient normally bothers to read it

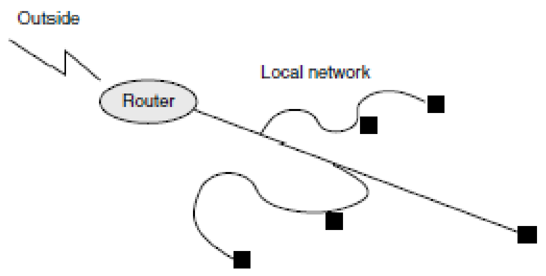


Figure 3: Traffic control with a router. Routers forward traffic that needs to leave a local network, and shield the external world from local traffic.

A router isolates one part of a network from another, both logically and physically. It will only forward the signal if the signal needs to travel along another segment to reach its destination address (see figure 3). The router is able to make this determination based on information about the topology of the network. This is an important function in the network: if every signal, sent by every computer, travelled along every cable in the world, communication would be impossible. Thus routers are essential to the scalability of networks as well as to the direction of traffic.

This simple model of network communications worked adequately for several years, but as the demands on networks increased, the load on routers became intolerable. There was therefore the need for a different architecture. This was provided by *switches*. Switches are topologically similar to routers, in that they act as a junction (often in star-formation) for several cables. The difference is that the switch knows nothing of the IP addresses or network segments joined to it. It routes and shields traffic by MAC address alone. This is cheaper and faster and can shield routers from purely local traffic, allowing them to concentrate on traffic to and from external sites. Like routers, switches prevent traffic from leaking along cables that it does not need to traverse; however, traditional switches segment only unicast, or node-to-node, traffic. Unlike routers, they do not normally limit broadcast traffic (packets that are addressed to all the nodes within the same IP network locale) or multicast traffic (packets that are distributed to a group of nodes). However, switch technology is advancing rapidly (see below). As switched networks have become more common, routers have continued to exist within the network, but they have been pushed toward the periphery of IP junctions.

As networks grow and traffic increases, one is forced to segment networks into more and more switched subnets to meet increasing performance demands. With these changes, broadcast and multicast traffic, that penetrates switch boundaries, has placed a greater burden on network bandwidth. In the worst case scenario, broadcast traffic can propagate out of control, leading to *broadcast storms* that paralyze a network.

VLANs (virtual LANs) are a step towards selective filtering at the switch level. They allow switches to protect swamped routers by offering different groups, or channels for related nodes. By limiting the distribution of broadcast, multicast and unicast traffic, they can help free up bandwidth, and reduce the need for expensive and complicated routing between switched networks, without involving routers. VLANs thus reinstate many of the advantages of routing-free LANs, but cheaply. Users and resources that communicate most frequently with each other can be grouped into common VLANs, regardless of physical location.

3.5.5 Protocols and encapsulation

Information transactions take place by agreed standards or *protocols*. Protocols exist to make sure that transmitted data are understood by the receiver in the way that the sender intended. On a network, protocols are required to make sure that data are understood, not only by the receiver, but by all the network hardware which carry them between source and destination. The data are wrapped up in envelope information which contains the address of the destination. Each transmission layer in the protocol stack (protocol hierarchy) is prefixed with some header information which contains the destination address and other data which identify it. The Ethernet protocol also has a trailer, see figure 4.

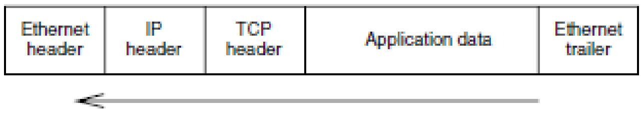


Figure 4: Protocol encapsulation.

Wrapping data inside envelope information is called *encapsulation* and it is important to understand the basics of these mechanisms. Network attacks make clever use of the features and flaws in these protocols and system administrators need to understand them in order to protect systems.

The Internet Family of protocols has been the basis of Unix networking for thirty years, since it was implemented as part of the Berkeley Software Distribution (BSD) Unix. The hierarchy is shown in figure 5.

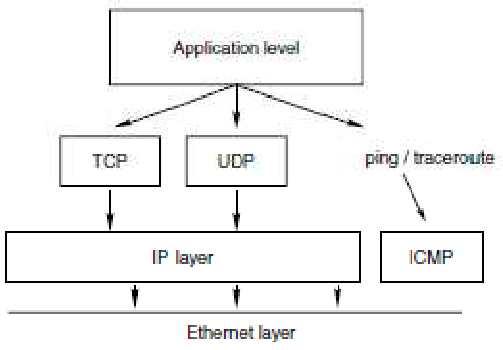


Figure 5: The Internet protocol hierarchy.

The transmission control protocol (TCP) is for reliable connection-oriented transfer. The user datagram protocol (UDP) is a rather cheaper connection-less service and the Internet control message protocol (ICMP) is used to transmit error messages and routing information for TCP/IP. These protocols have an address structure which is hierarchical and *routable*, which means that IP addresses can find their way from any host in the world to any other so long as they are connected. The Ethernet protocol does not know much more about the world than the cable it is attached to.

Windows supports at least three network protocols, running on top of Ethernet.

- *NETBEUI:* NETBIOS Extended User Interface, Microsoft's own network protocol. This was designed for small networks and is not routable. It has a maximum limit of 20 simultaneous users and is thus hardly usable.
- *NW Link/IPX:* Novell/Xerox's IPX/SPX protocol suite. Routable. Maximum limit of 400 simultaneous users.
- *TCP/IP:* Standard Internet protocols. The default for Windows-like and Unixlike systems. Novell Netware and Apple MacIntosh systems also support TCP/IP. There is no in-built limit to the number of simultaneous users.

Novell's Netware PC server software is based mainly on the IPX suite running on Ethernet hardware; MacIntosh networks have used their own proprietary Appletalk which will run on Ethernet or token ring hardware, but this is now being exchanged for TCP/IP. All platforms are converging on the use of TCP/IP for its open standard and its generality.

3.5.6 Data formats

There are many problems which arise in networking when hardware and software from different manufacturers have to exist and work together. Some of the largest computer companies have tried to use this to their advantage on many occasions in order to make customers buy only their products. An obvious example is the choice of network protocols used for communication. Both Apple and Microsoft have tried to introduce their own proprietary networking protocols. TCP/IP has won the contest because it was an *inter*-network protocol (i.e. capable of working on and joining together any hardware type) and also because it is a freely open standard.

Neither the Appletalk nor the NETBIOS protocols have either of these features. This illustrates how networking demands standards. That is not to say that some problems do not still remain. No matter how insistently one attempts to fuse operating systems in a network melting pot, there are basic differences in hardware and software which cannot be avoided. One example, which is occasionally visible to system administrators when compiling software, is the way in which different operating systems represent numerical data. Operating systems (actually the hardware they run on) fall into two categories known as *big endian* and *little endian*. The names refer to the *byte-order* of numerical representations.

The names indicate how large integers (which require say 32 bits or more) are stored in memory. Little endian systems store the least significant byte first, while big endian systems store the most significant byte first. For example, the representation of the number 34,677,374 has either of the forms shown in figure 6. Obviously if one is transferring data from one host to another, both

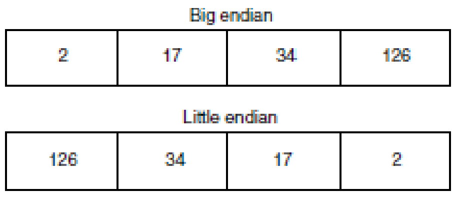


Figure 6: Byte ordering sometimes has to be specified when compiling software. The representation of the number 34,677,374 has either of these forms

hosts have to agree on the data representation otherwise there would be disastrous consequences. This means that there has to be a common standard of *network byte ordering*. For example, Solaris (SPARC hardware) uses network byte ordering (big endian), while Windows or Unix-like operating systems on Intel hardware use the opposite (little endian). Intel systems have to convert their data format every time ordered data are transmitted over the network.

3.6 IPv4 networks

TCP/IP networking is so important to networked hosts that we shall return to it several times during the course of this material. Its significance is cultural, historical and practical, but the first item in our agenda is to understand its logistic structure.

3.6.1 IP addresses

Every network interface on the Internet needs to have a unique number which is called its address. IP addresses are organized hierarchically so that they can be searched for by router networks. Without such a structure, it would be impossible to find a host unless it were part of the same cable segment. At present the Internet protocol is at version 4 and this address consists of four bytes, or 32 bits. In the future this will be extended, in a new version of the Internet protocol IPv6, to allow more IP addresses since we are rapidly using up the available addresses. The addresses will also be structured differently. The form of an IP address in IPv4 is

aaa.bbb.ccc.mmm

Some IP addresses represent networks, whereas others represent individual interfaces on hosts and routers. Normally an IP address represents a host attached to a network.

In every IPv4 address there are 32 bits. One uses these bits in different ways: one could imagine using all 32 bits for host addresses and keep every host on the same enormous cable, without any routers (this would be physically impossible in practice), or we could use all 32 bits for network addresses and have only one host per network (i.e. a router for every host). Both these extremes are silly; we are trying to save resources by sharing a cable between convenient groups of hosts, but shield other hosts from irrelevant traffic. What we want instead is to group hosts into clusters so as to restrict traffic to localized areas.

Networks were grouped *historically* into three classes called class A, class B and class C networks, in order to simplify traffic routing. Class D and E networks are also now defined, but these are not used for regular traffic. This rigid distinction between different types of network addresses has proved to be a costly mistake for the IPv4 protocol. Amongst other things, it means that only about two percent of the actual number of IP addresses can actually be used with this scheme. So-called *classless* addresses (CIDR) were introduced in the 1990s to patch the problem of the classed addressing, but not all deployed devices and protocol versions were able to understand the new classless addresses, so classed addressing will survive in books and legacy networks for some time.

The difference between class A, B and C networks lies in which bits of the IP addresses refer to the network itself and which bits refer to actual hosts within a network. Note that the details in these sections are subject to rapid change, so readers should check the latest details on the web.

Class A legacy networks

IP addresses from 1.0.0.0 to 127.255.255.255 are *class A* networks. Originally only 11.0.0.0 to 126.255.255.255 were used, but this is likely to change as the need for IPv4 address space becomes more desperate. In a class A network, the first byte is a network part and the last three bytes are the host address (see figure 7). This allows 126 possible networks (since network 127 is reserved for the loopback service). The number of hosts per class A network is 2563 minus reserved host addresses on the network. Since this is a ludicrously large number, none of the owners of class A networks are able to use all of their host addresses.

Class A networks are no longer issued (as class A networks), they are all assigned, and all the free addresses are now having to be reclaimed using CIDR. ClassA networks were intended for very large organizations (the U.S. government, Hewlett Packard, IBM) and are only practical with the use of a net mask which divides up the large network into manageable subnets. The default subnet mask

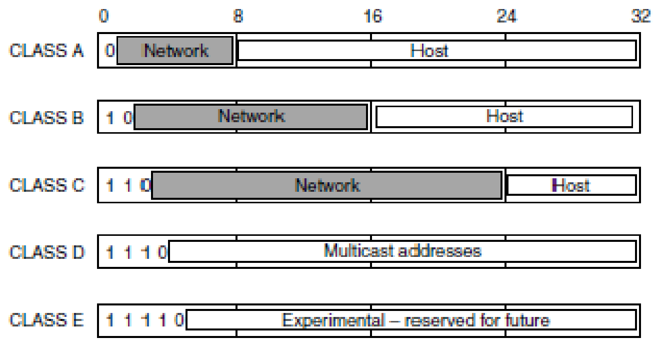


Figure 7: Bit view of the 32 bit IPv4 addresses.

Class B legacy networks

IP addresses from 128.0.0.0 to 191.255.0.0 are *class B* networks. There are 16,384 such networks. The first two bytes are the network part and the last two bytes are the host part. This gives a maximum of 2562 minus reserved host addresses, or 65,534 hosts per network. Class B networks are typically given to large institutions such as universities and Internet providers, or to institutions such as Sun Microsystems, Microsoft and Novell. All the class B addresses have now been allocated to their parent organizations, but many of these lease out these addresses to third parties. The default subnet mask is 255.255.0.0.

Class C legacy networks

IP addresses from 192.0.0.0 to 223.255.255.0 are *class C* networks. There are 2,097,152 such networks. Here the first three bytes are network addresses and the last byte is the host part. This gives a maximum of 254 hosts per network. The default subnet mask is 255.255.255.0. Class C networks are the most numerous and there are still a few left to be allocated, though they are disappearing with alarming rapidity.

Class D (multicast) addresses

Multicast networks form what is called the MBONE, or multicast backbone. These include addresses from 224.0.0.0 to 239.255.255.0. These addresses are not normally used for sending data to individual hosts, but rather for routing data to multiple destinations. Multicast is like a restricted broadcast. Hosts can 'tune in' to multicast channels by subscribing to MBONE services.

Class E (Experimental) addresses

Addresses 240.0.0.0 to 255.255.255.255 are unused and are considered experimental, though this may change as IPv4 addresses are depleted.

3.6.2 Subnets and broadcasts

What we refer to as a network might consist of very many separate cable systems, coupled together by routers and switches. One problem with very large networks is that *broadcast messages* (i.e. messages which are sent to every host) create traffic which can slow a busy network. In most cases broadcast messages only need to be sent to a subset of hosts which have some logical or administrative relationship, but unless something is done a broadcast message will by definition be transmitted to all hosts on the network. What is needed then is a method of assigning groups of IP addresses to specific cables and limiting broadcasts to hosts belonging to the group, i.e. breaking up the larger community into more manageable units. The purpose of subnets is to divide up networks into regions which naturally belong together and to isolate regions which are independent. This reduces the propagation of useless traffic, and it allows us to *delegate* and distribute responsibility for local concerns.

This logical partitioning can be achieved by dividing hosts up, through routers, into subnets. Each network can be divided into *subnets* by using a *netmask*. Each address consists of two parts: a *network address* and a *host address*. A system variable called the *netmask* decides how IP addresses are interpreted locally. The netmask decides the boundary between how many bits of the IP address will be kept for hosts and how many will be kept for the network location name. There is thus a trade-off between the number of allowed domains and the number of hosts which can be coupled to each subnet. Subnets are usually separated by routers, so the question is, how many machines do we want on one side of a router?

The netmask is most easily interpreted as a binary number. When looking at the netmask, we have to ask which bits are ones and which are zeros? The bits which are *ones* decide which bits can be used to specify the subnets within the domain. The bits which are zeros decide which are hostnames on each subnet. The local network administrator decides how the netmask is to be used.

The host part of an IP address can be divided up into two parts by moving the boundary between network and host part. The netmask is a variable which contains zeros and ones. Every one represents a network bit and every zero represents a host bit. By changing the value of the netmask, we can trade many hosts per network for many subnets with fewer hosts. A subnet mask can be used to separate hosts which also lie on the same physical network, thereby forcing them to communicate through the router.

3.6.3 Interface settings

The IP address of a host is set in the network interface. The Unix command if config (interface-configuration) or the Windows command ip config are used to set this. Normally the address is set at boot time by a shell script executed as part of the rc startup files. These files are often constructed automatically during the system installation procedure. The ifconfig command is also used to set the broadcast address and netmask for the subnet. Each system interface has a name. Here are the network interface names commonly used by different Unix types.

```
Sun            le0 / hme0
DEC ultrix     ln0
DEC OSF/1      ln0
HPUX           lan0
AIX            en0
GNU/Linux      eth0
IRIX           ec0
FreeBSD        ep0
Solarisx86     dnet0
```

Look at the manual entry for the system for the ifconfig command, which sets the Internet address, netmask and broadcast address. Here is an example on a SUN system with a Lance-Ethernet interface.

```
ifconfig le0 192.0.2.10 up netmask 255.255.255.0 broadcast 192.0.2.255
```

Normally we do not need to use this command directly, since it should be in the startup-files for the system, from the time the system was installed. However we might be working in single-user mode or trying to solve some special problem. A system might have been incorrectly configured.

3.6.4 Default route

Unless a host operates as a router in some capacity, it only requires a minimal routing configuration. Each host must define a *default route* which is a destination to which outgoing packets will be sent for processing when they do not belong to the subnet. This is the address of the router or gateway on the same network segment. It is set by a command like this:

```
route add default my-gateway-address 1
```

3.6.5 ARP/RARP

The Address Resolution Protocol (ARP) is a name service directory for translating from IP address to hardware, Media Access Control (MAC) address (e.g. Ethernet address). The ARP service is mirrored by a reverse lookup ARP service (RARP). RARP takes a hardware address and turns it into an IP address.

Ethernet MAC addresses are required when forwarding traffic from one device to another, on the same subnet. While it is the IP addresses that contain the structure of the Internet and permit routing, it is the hardware address to which one must deliver packets in the final instance; because IP addresses are encapsulated in Ethernet packets.

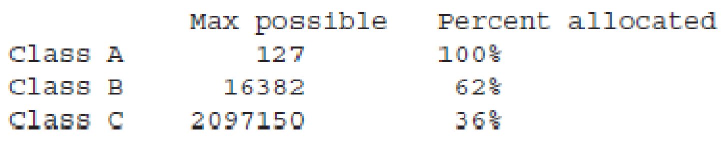
Hardware addresses are cached by each host on the network so that repeated calls to the service ARP translation service are not required. Addresses are checked later however, so that if an address from a host claiming to have a certain IP address originates from an incorrect hardware address (i.e. the packet does not agree with the information in the cache) then this is detected and a warning can be issued to the effect that two devices are trying to use the same IP address. ARP sends out packets on a local network asking the question 'Who has IP address xxx.yyy.zzz.mmm?' The host concerned replies with its hardware address.

For hosts which know their own IP address at boot-time these services only serve as confirmations of identity. Diskless clients (which have no place to store their IP address) do not have this information when they are first switched on and need to ask for it. All they know originally is the unique hardware (Ethernet) address which is burned into their network interface. In order to bring up and configure an Internet interface they must first use RARP to find out their IP addresses from a RARP server. Services like BOOTP or DHCP are used for this.
Also the Unix file /etc/ethers and rarpd can be used. The ARP protocol has no authentication mechanism, and it is therefore easily poisoned with incorrect data. This can be used by malicious parties to reroute packets to a different destination.

3.7 Address space in IPv4

As we have seen, the current implementation of the Internet protocol has a number of problems. The model of *classed* Internet addresses was connected to the design of early routing protocols. This has proved to be a poor design decision, leading to a sparse usage of the available addresses.

It is straightforward to calculate that, because of the structure of the IP addresses, divided into class A, B and C networks, something under two percent of the possible addresses can actually be used in practice. A survey from *Unix Review* in March 1998 showed that, of the total numbers of addresses, these area ready allocated:



	Max possible	Percent allocated
Class A	127	100%
Class B	16382	62%
Class C	2097150	36%

Of course, this does not mean that all of the allocated addresses are in active use. After all, what organization has 65,535 hosts? In fact the survey showed that under two percent of these addresses were actually in use. This is an enormous wastage of IP addresses. Amongst the class C networks, where smaller companies would like address space, the available addresses are being used up quickly, but amongst the class A networks, the addresses will probably never be used. A new addressing structure is therefore required to solve this problem. Three solutions have been devised.

3.7.1 Network Address Translation

In order to provide a 'quick fix' for organizations that required only partial connectivity, Network Address Translation (NAT) was introduced by a number of router manufacturer. In a NAT, a network is represented to the outside world by a single official IP address; it shields the remainder of its networked machines on a private network that (hopefully) uses non-routable addresses (usually 10.x.x.x). When one of these hosts on the private network attempts to contact an address on the Internet, the Network Address Translator creates the illusion that the request comes from the single representative address. The return data are, in turn, routed back to the particular host 'as if by magic' (see figure 8). NAT makes associations of this form:

(private IP, private port) <-> (public IP, public port)

It is important that the outside world (i.e. the true Internet) should not be able to see the private addresses behind a NAT. Using a private address in a public IP address is not just bad manners, it could quickly spoil routing protocols and preclude us from being able to send to the real owners of those addresses. NATs are often used in conjunction with a firewall.

Network address translation is a quick and cheap solution to giving many computers access to the Internet, but it has many problems. The most serious, perhaps, is that it breaks certain IP security mechanisms that rely on IP addresses, because IP addresses are essentially spoofed. Thus some network services will not run through a NAT, because the data stream looks as though it has been forged. Indeed, it has.

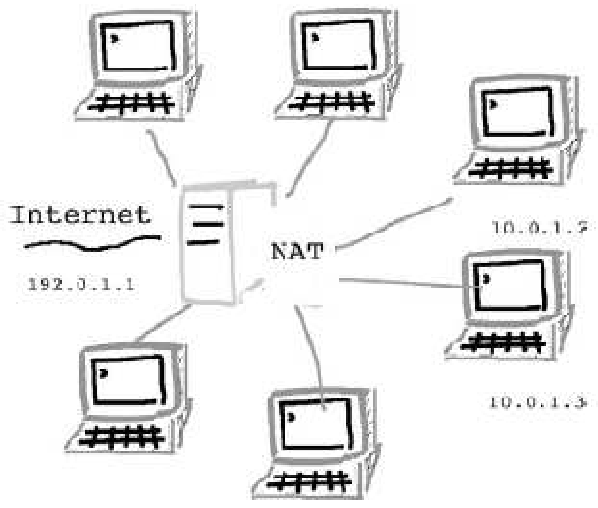


Figure 8: Network address translation masquerades many private addresses as a single IP address.

3.8 IPv6 networks

We have already mentioned the problems with IPv4 in connection with address allocation and routing. Other problems with IPv4 are that it is too easy to take control of a connection by guessing sequence numbers. Moreover there is no native support for encryption, Quality of Service guarantees or for mobile computing. All of these things are increasingly important, in a congested virtual community.

In an attempt to address these problems, the Internet Engineering Task Force (IETF) put together a workgroup to design a new protocol. Several suggestions were put forward, some of which attempted to bring the IP model closer to the OSI reference model (see table 2.10), however these suggestions were abandoned in favor of a simple approach that eliminated obsolete elements of IPv4 and extended addresses from 32 to 128 bits. The new IPv6 proposal was adopted for its inclusion of issues like Quality of Service (QoS) and mobility. With 128 bit addresses, even with a certain inefficiency of allocation, it is estimated that there will be enough IPv6 addresses to support a density of more than 10,000 IP addresses per square meter which ought to be enough for every toaster and wristwatch on the planet and beyond. The port space of IPv6 is shared with IPv4.

3.8.1 IPv6 addresses

Stepping up from 32 bits to 128 bits presents problems of representation for IPv6 addresses. If they were coded in the usual denary 'dotted' octet form, used by

0–3	Never used in a working version
4	The Internet as we know it
5	Stream protocol –ST –(never an IPng)
6	SIP → SIPP (Simple Internet protocol plus) → IPv6
7	IPv7 → TP/IX → CATNIP (died)
8	Pip (later joined SIP)
9	TUBA (died)
10–15	Not in use

Table 4: A history of projects for IP protocol development.

IPv4, addresses would be impossibly long and cumbersome. Thus a hexadecimal notation was adopted, together with some rules for abbreviation. Each pair of hexadecimal digits codes one byte, or eight bits, so addresses are 32 hexadecimal characters long, or eight blocks of 4 hex-numbers: e.g.

2001:0700:0700:0004:0290:27ff:fe93:6723

The addresses are prefixed in a classless fashion, like CIDR addresses, making them hierarchically delegable. The groups of four hexadecimal numbers are separated by a colon ':' -- to look like a 'big dot'. The empty colon set '::' stands for a string of 0 bits, or ':0000:'. Similarly, trailing zeros can be omitted.

Here is an example address:

```
2001 : 700 : 700 : 4 : 290 : 27ff : fe93 : 6723
***************               ++
```

The starred part is a delegated IP-series, given by an Internet addressing authority or service provider. The '++' numbers are usually 'ff' or some other padding. The remaining numbers are taken from the MAC (Media Access Control), e.g. Ethernet address of the network interface. This can be seen with:

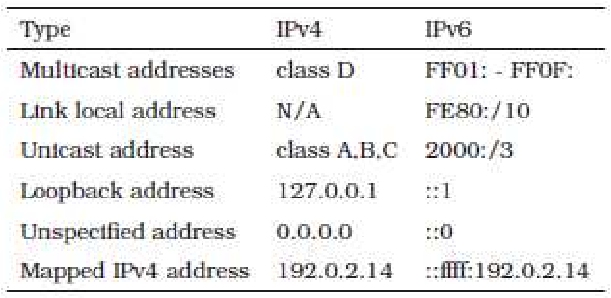
```
host$ ifconfig -a


eth0    Link encap:Ethernet HWaddr 00:90:27:93:67:23
        inet addr:128.39.74.16 Bcast:128.39.75.255 Mask:255.255.254.0
        inet6 addr: fe80::290:27ff:fe93:6723/10 Scope:Link
        inet6 addr: 2001:700:700:4:290:27ff:fe93:6723/64 Scope:Global
...
```

Thus, once a prefix has been provided by a local gateway, every host knows its global address at once – no manual address allocation is required. A host can have several IPv6 addresses however. Others can be assigned according to some procedure. A version of the dynamic host control protocol (DHCPv6) has been put forward for this purpose.

3.8.2 Address allocation

The IETF has designated the address range 2000::/3 to be global unicast address space that IANA may allocate to the Regional Internet Registries (RIR)s (see figure 9). IANA has allocated initial ranges of global unicast IPv6 address space from the 2001::/16 address block to the existing RIRs. The subsequent allocations of the 2000::/3 unicast address space are made by Regional Internet Authorities (RIRs), with their own allocation policies. End sites will generally be given /48,/64 or /128 assignments.



Type	IPv4	IPv6
Multicast addresses	class D	FF01: - FF0F:
Link local address	N/A	FE80:/10
Unicast address	class A,B,C	2000:/3
Loopback address	127.0.0.1	::1
Unspecified address	0.0.0.0	::0
Mapped IPv4 address	192.0.2.14	::ffff:192.0.2.14

Table 5: Some important IPv4 and IPv6 addresses compared.

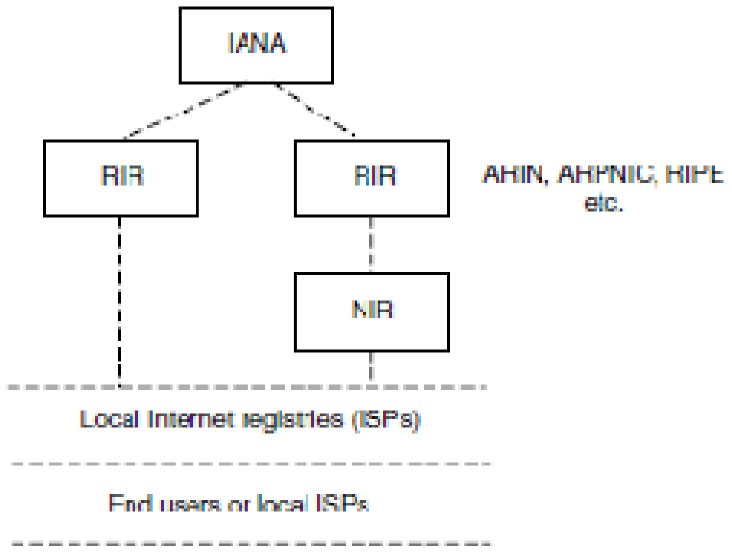


Figure 9: The hierarchy of Internet address delegation. IANA (Internet Assigned Numbers Authority) leads the administration of the Internet at the topmost level, and delegates authority to regional Internet registries (RIR) such as INTERNIC (US), APNIC (Asia-Pacific) and RIPE NCC (Europe). These, in turn, delegate to countries and thence to ISPs.

3.8.3 Auto configuration and neighbor discovery

With huge networks and unwieldy addresses, an important aspect of IPv6 is Auto configuration, including neighbor discovery protocols.

When an IPv4 host joins a local area network, it uses the ARP protocol to bind its IP address to its Ethernet MAC address. The Address Resolution Protocol (ARP), documented in RFC 826, is used to do this. It has also been adapted for other media, such as FDDI. ARP works by broadcasting a packet to all hosts on the local network. The packet contains the IP address the sender is interested in communicating with. Most hosts ignore the packet. The target machine, recognizing that the IP address in the packet matches its own, returns an answer.

To reduce the number of address resolution requests, a client (host, router or switch) normally caches resolved addresses for a short interval of time. The ARP cache is of a finite size, and would become full of incomplete and obsolete entries for computers that are not in use if it was allowed to grow without check; thus, it is periodically flushed of all entries. This deletes unused entries and frees space in the cache. It also removes any unsuccessful attempts to contact computers which are not currently running. Since it has no authentication mechanisms, the ARP cache can be poisoned by attackers allowing data to be redirected to the wrong receiver.

In IPv6, ARP is supplanted by a message-passing protocol for neighbor discovery that uses the IPv6 mechanisms on the link-level addresses. A new host can thus automatically discover a local IPv6 gateway to find a route to the outside world. A default route assignment does not normally require a manual assignment. When a gateway is found, a ‘scope global’ address is automatically assigned to the interface, based on the MAC address of the host, allowing routable communication. The same IPv6 address can be configured on several interfaces. If a gateway is not found, a host can still contact other IPv6 enabled hosts on the same VLAN using the ‘link local’ address that is configured at start up.

3.8.4 Mobile computing

IPv6 includes support for mobile routing. If a computing device belonging to a particular routing domain finds itself connected via a different routing environment, it first attempts to connect to its home router and establish a forwarding address. This allows packets sent to its fixed IP address to be forwarded to the new location, as well as establishing a direct route for all self-initiated communication. The forwarding addresses are called ‘care of’ (i.e. c/o) addresses.

SELF ASSESSMENT EXERCISES

1. Describe the main hardware components in a human–computer system.
2. What rules of thumb would you use for handling the different hardware components.
3. What effect does temperature have on computer systems?
4. What is the function of an operating system? (Hint: how do you define an operating system?)

4.0 CONCLUSION

We have discussed the main components of human-computer system and have also reviewed both Unix and Windows tools, you will need to analyze networks. This should make you self-sufficient at your system administration activities.

5.0 SUMMARY

In system administration, the word *system* is used to refer both to the operating system of a computer and often, collectively the set of all computers that cooperate in a network. There are three main components in a human–computer system: *Humans*, *Host computers and Network hardware.*

To be a system administrator it is important to have a basic appreciation of the frailties and procedures surrounding hardware. An operating system has a number of key elements: (i) a *technical layer of software* for driving the hardware of the computer, like disk drives, the keyboard and the screen; (ii) a *filesystem* which provides a way of organizing files logically, and (iii) a simple *user interface* which enables users to run their own programs and to manipulate their files in a simple way. Of central importance to an operating system is a core software system or *kernel* which is responsible for allocating and sharing the resources of the system between several running programs or *processes. T*he two most popular classes of operating system today are Unix-like operating systems (i.e. those which are either derived from or inspired by System V or BSD) and Microsoft Windows NT-like operating systems.

Files and filesystems are at the very heart of what system administration is about. Almost every task in host administration or network configuration involves making changes to files. ACLs, or access control lists are a modern replacement for file modes and permissions. With access control lists we can specify precisely the access rights to files for each user individually. The Windows operating system supports a variety of legacy filesystems for backward compatibility with DOS and Windows 9x.

A network is a number of pathways for communication between two or more hosts. Networking is increasingly important, as computers are used more and more as devices for media access rather than for computation. Networking raises issues for system management at many levels, from its deployment to its configuration and usage. On a network, protocols are required to make sure that data are understood, not only by the receiver, but by all the network hardware which carry them between source and destination.

Every network interface on the Internet needs to have a unique number which is called its address. IP addresses are organized hierarchically so that they can be searched for by router networks. Without such a structure, it would be impossible to find a host unless it were part of the same cable segment. At present the Internet protocol is at version 4 and this address consists of four bytes, or 32 bits. In the future this will be extended, in a new version of the Internet protocol IPv6, to allow more IP addresses since we are rapidly using up the available addresses. The addresses will also be structured differently. The form of an IP address in IPv4 is aaa.bbb.ccc.mmm.

6.0 TUTOR MARKED ASSIGNMENT

1. Summarize the similarities between Unix and Windows.

2a. How are files shared between users in Unix/Windows?
  b. How are files shared between computers in Unix/Windows?

3a. What is an IP address and what does it look like?
  b. Do class A, B, C IP addresses have any meaning today?
  c. What IPv4 addresses are reserved and why?

4a. What is meant by Network Address Translation, and what is its main purpose?
  b. Describe how IPv6 addresses differ from IPv4 addresses.

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

3. Hunt. C. (2002). TCP/IP Network Administration (3rd Ed.). OReilly.

4. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley

5. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

UNIT 2: NETWORKED COMMUNITIES

1.0 INTRODUCTION

System administration is not just about machines and individuals, it is about communities. There is the local community of users on multi-user machines; then there is the local area network community of machines at a site. Finally, there is the global community of all machines and networks in the world. The aim of this unit is to learn how to navigate network systems using standard tools, and place each piece of the puzzle into the context of the whole.

2.0 OBJECTIVES

At the end of this unit you should be able to:

- State the principle at work in a cooperative enterprise, such as a network
- Explain the role of policy in a network community
- Describe the social community structures in a human-computer system
- State the PROs and CONs of making a network completely uniform in the choice of hardware and software
- Know about network naming orientation
- Know how a computer knows its IP address
- Describe alternative models for organizing network resources
- Explain what the ping program does
- Know what is meant by a Uniform Reform Locator (ULR) and how it is used
- Know what the Domain Name Service (DNS) is

3.0 MAIN CONTENT

3.1 Communities and enterprises

The basic principle of communities is:

Principle (Communities). *What one member of a cooperative community does affects every other member and vice versa. Each member of the community therefore has a responsibility to consider the well-being of the other members of the community.*

When this principle is ignored, it leads to conflict. One attempts to preserve the stability of a community by making *rules*, *laws* or *policies*. The main difference between these is only our opinion of their severity: rules and laws do not exist because there are fundamental rights and wrongs, they exist because there is a need to summarize the consensus of opinion in a community group. A social rule thus has two purposes:

• To provide a widely accepted set of conventions that simplify decisions,
avoiding the need to think through things from first principles every time.
• To document the will of the community for reference.

Rules can never cover every eventuality. They are convenient approximations to reality that summarize common situations. An idealist might hope that rules would never be used as a substitute for thought, however this is just how they are used in practice. Rules simplify the judgment process for common usage, to avoid constant re-evaluation (and perhaps constant change).

We can apply this central axiom for the user community of a multiuser host:

Corollary to principle (Multiuser communities). *A multiuser computer system does not belong to any one user. All users must share the common resources of the system. What one user does affects all other users and vice versa. Each user has a responsibility to consider the effect of his/her actions on all the other users* and also for the world-wide network community:

Corollary to principle (Network communities). *A computer that is plugged into the network is no longer just our own. It is part of a society of machines which shares resources and communicates with the whole. What that machine does affects other machines. What other machines do affects that machine.*

The ethical issues associated with connection to the network are not trivial, just as it is not trivial to be a user in a multiuser system, or a member of a civil community. Administrators are, in practice, responsible for their organization's conduct to the entire rest of the Internet, by ensuring conformance with policy. This great responsibility should be borne wisely.

3.2 Policy blueprints

By placing a human–computer system into an environment that it has no direct control over, we open it up to many risks and random influences. If we hope to maintain a predictable system, it is important to find a way to relate to and make sense of these external factors. Moreover, if we wish to maintain a predictable system, then we need to know how to recognize it: what should the system look like and how should it behave? The tool for accomplishing this is policy.

Definition (Policy). *Policy is a statement of aims and wishes that is codified, as far as possible, into a formal blueprint for infrastructure and a schema of responses (contingencies) for possible events.*

A policy's aim is to maintain order in the face of the chaos that might be unleashed upon it– either from the environment that it does not control, or from a lack of control of its own component parts. Any system can spiral out of control if it is not held in check. By translating hopes and wishes into concrete rules and regimens, we build a model for what a predictable system should look like.

- A blueprint of infrastructure.
- Production targets (resource availability).
- Restriction of behavior (limiting authority, access).

- Stimulus–response checklists (maintenance).

A policy determines only an approximate state for a human–computer system – not a state in the sense of a static or frozen configuration, but rather a dynamical equilibrium or point of balance. Human–computer systems are not deterministic, but the aim of policy is to limit the unpredictable part of their behavior to the level of background noise.

3.3 System uniformity

The opportunity to standardize parts of a system is an enticing prospect that can potentially lead to great simplification; but that is not the full story. Given the chance to choose the hardware and software at a site, one can choose a balance between two extreme strategies: to standardize as far as possible, or to vary as much as possible. Curiously, it is not necessarily true that standardization will always increase predictability. That would be true in a static system – but in a real life, dynamical system we have to live with the background noise caused by the parts that we do not control.

Strategically, trying 'every which way', i.e. every possible variation on a theme can pay off in terms of productivity. Moreover, a varied system is less vulnerable to a single type of failure. Thus, if we look at the predictability of *productivity*, a certain level of variation can be an advantage. However, we must find an appropriate balance between these two principles:

Principle (Uniformity). *A uniform configuration minimizes the number of differences and exceptions one has to take into account later, and increases the static predictability of the system. This applies to hardware and software alike.*

Principle (Variety). *A variety of configurations avoids 'putting all eggs in one basket'. If some components are poor, then at least not all will be poor. A strategy of variation is a way of minimizing possible loss.*

It is wise for system administrators to spend time picking out reliable hardware and software. The more different kinds of system we have, the more difficult the problem of installing and maintaining them, but if we are uncertain of what is best, we might choose to apply a random sample in order to average out a potential loss. Ideally perhaps, one should spend a little time researching the previous experiences of others in order to find a 'best choice' and then standardize to a large degree.

PC hardware is often a mélange of random parts from different manufacturers. Much work can be saved by standardizing graphics and network interfaces, disk sizes, mice and any other devices that need to be configured. This means not only that hosts will be easier to configure and maintain, but also that it will be easier to buy extra parts or cannibalize systems for parts later. On the other hand, automated agents like cfengine can make the task of maintaining a variety of options a manageable task.

With software, the same principle applies: a uniform software base is easier to install and maintain than one in which special software needs to be configured in special ways. Fewer methods are available for handling the *differences* between systems; most administration practices are based on standardization. However, dependence on one software package could be risky for an organization. There is clearly a complex discussion around these issues.

3.4 User behavior: socio-anthropology

Most branches of computer science deal primarily with software systems and algorithms. System administration is made more difficult by the fact that it deals with communities and is therefore strongly affected by what human beings do. In short, a large part of system administration can be characterized as sociology or anthropology. A newly installed machine does not usually require attention until it is first used, but as soon as a user starts running programs and storing data, the reliability and efficiency of the system are tested. This is where the challenge of system administration lies.

The load on computers and on networks is a social phenomenon: it peaks in response to patterns of human behavior. For example, at universities and colleges network traffic usually peaks during lunch breaks, when students rush to the terminal rooms to surf on the web or to read E-mail. In industry the reverse can be true, as workers flee the slavery of their computers for a breath of fresh air (or polluted air). In order to understand the behavior of the network, the load placed on servers and the availability of resources, we have to take into account the users' patterns of behavior (see figure 1).

3.5 Clients, servers and delegation

At the heart of all cooperation in a community is a system of *centralization* and *delegation*. No program or entity can do everything alone, nor is everyone expected to do so. It makes sense for certain groups to specialize in performing certain jobs. That is the function of a society and good management.

Principle (Delegation I). *Leave experts to do their jobs. Assigning responsibility for a task to a body which specializes in that task is a more efficient use of resources.*

If we need to find out telephone numbers, we invent the directory enquiry service: we give a special body a specific job. They do the phone-number research (once and for everyone) and have the responsibility for dealing out the information on request. If we need a medical service, we train doctors in the specialized knowledge and trust them with the responsibility. That is much more efficient than expecting every individual to have to research phone numbers by themselves, or to study medicine personally. The advantage with a *service* is that one avoids repeating work unnecessarily and one creates special agents with an aptitude for their task.
In database theory, this process is called *normalization* of the system.

The principle of specialization also applies in system administration. Indeed, in recent years the number of client-server systems has grown enormously, because

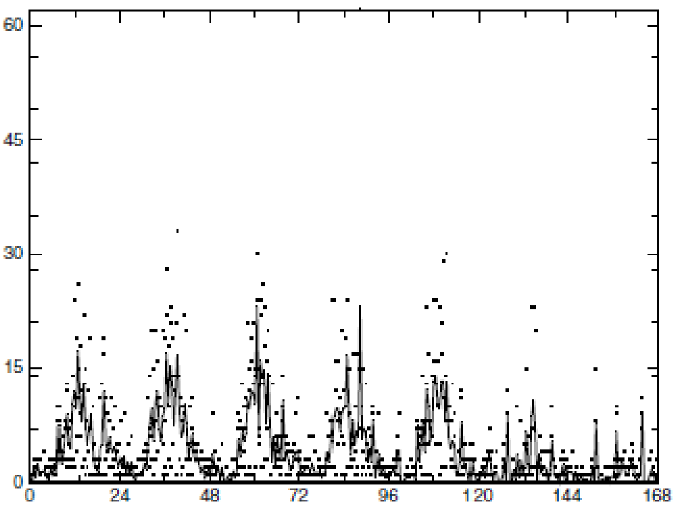


Figure 1: E-mail traffic at a College measured over the course of many weeks. The plot shows the weekly average from Monday to Sunday. Over each 24 hour period, there is a daily peak showing users' working hours, and during the week, there is a peak around midweek, and little activity during the weekends,

of possibilities offered by networking. Not only can we give a special daemon on one host a special job, but we can say that that daemon will do the job for every other host on the network also. As long as the load placed on the network does not lead to a bottleneck, this is a very efficient centralization of resources. Clearly, the client-server model is an extended way of sharing resources. In that sense it is like a distributed generalization of the kernel.

The client-server nomenclature has been confused by history. A server is not a host, but a program or process which runs on a host. A client is any process which requires the services of a server. In Unix-like systems, servers are called daemons. In Windows they are just called services. Unfortunately, it is common to refer to the host on which a server process runs as being a server. This causes all sorts of confusion.

The name 'server' was usurped, early on, for a very specific client-server relationship. A server is often regarded as a large machine which performs some difficult and intensive task for the clients (an array of workstations). This prejudice comes from the early days when many PC-workstations were chained together in a network to a single PC which acted as file-server, and printer server, sharing a disk and printer with all of the machines. The reason for this architecture, at the time, was that the operating system of that epoch MS-DOS was not capable of multitasking, and thus the best solution one could make was to use a new PC for each new task. This legacy of one-machine, one-user, one-purpose, still pervades newer PC operating system philosophy. Meanwhile, Unix and later experimental operating systems have continued a general policy of 'any machine, any job', as part of the vision of *distributed operating systems*. There are many reasons for choosing one strategy or the other.

In fact a server-host can be anything from a Cray to a laptop. As long as there is a process which executes a certain service, the host is a server-host.

3.6 Host identities and name services

Whenever computers are coupled together, there is a need for each to have an individual and unique identity. This need has been recognized many times, by different system developers, and the result is that today's computer systems can have many different names which identify them in different contexts. The outcome is confusion. For Internet-enabled machines, the IP address of the host is usually sufficient for most purposes. A host can have all of the following:

- *Host ID:* Circuit board identity number. Often used in software licensing.
- *Install name:* Configured at installation time. This is often compiled into the kernel, or placed in a file like /etc/hostname. Solaris adds to the multiplicity by also maintaining the install name in /etc/hostname.le0 or an equivalent file for the appropriate network interface, together with several files in /etc/net/*/hosts.
- *Application level name:* Any name used by application software when talking to other hosts.
- *Local file mapping:* Originally the Unix /etc/hosts file was used to map IP addresses to names and vice versa. Other systems have similar local files, to avoid looking up on network services.
- *Network Information Service:* A local area network database service developed by Sun Microsystems. This was originally called Yellow Pages and many of its components still bear the 'yp' prefix.
- *Network level address(es):* Each network interface can be configured with an IP address. This number converts into a text name through a name service.
- *Link level address(es):* Each network interface (Ethernet/FDDI etc.) has a hardware address burned into it at the factory, also called its MAC address, or media access control address. Some services (e.g. RARP) will turn this into a name or an IP address through a secondary naming service like DNS.
- *DNS name(s):* The name returned by a domain name server (DNS/BIND) based on an IP address key.
- *WINS name(s):* The name returned by a WINS server (Microsoft's name server) based on an IP address. WINS was deprecated as of Windows 2000.

Different hardware and software systems use these different identities in different ways. The host ID and network level addresses simply exist. They are unique and nothing can be done about them, short of changing the hardware. For the most part they can be ignored by a system administrator. The network level MAC address is used by the network transport system for end-point data delivery, but this is not something which need concern most system administrators. The network hardware takes care of itself.

At boot-time, each host needs to obtain a unique identity. In today's networks that means a unique IP address per interface and an associated name for convenience or to bind the multiple IP addresses together. The purpose of this identity is to uniquely identify the host amongst all of the others on the world-wide network. Although every network interface has a unique Ethernet address or token ring address, these addresses do not fall into a hardware-independent hierarchical structure. In other words Ethernet addresses cannot be used to route messages from one side of the planet to the other in a simple way. In order to make that happen, a system like TCP/IP is required. At boot-time then each host needs to obtain an Internet identity. It has two choices:

- Ask for an address to be provided from a list of free addresses. (DHCP or BOOTP protocols)
- Always use the same IP address, stored on its system configuration files. (Requires correct information on the disk)

The first of these possibilities is sometimes useful for terminal rooms containing large numbers of identical machines. In that case, the specific IP address is unimportant as long as it is unique. The second of these is the preferred choice for any host which has special functions, particularly hosts which provide network services. Network services should always be at a well-known, static location.

From the IP address a name can be automatically attached to the host through an Internet *naming service*. There are several services which can perform this conversion. DNS, NIS and WINS are the three prevalent ones. DNS is the superior service, based on a world-wide database; it can determine hostname to IP address mappings for any host in the world. NIS (Unix) and WINS (Windows) are local network services which are essentially redundant as name services. They continue to exist because of other functions which they can perform.

As far as any host on a TCP/IP network is concerned, a host is its IP address and any names associated with that address. Any names which are used internally by the kernel, or externally, are quite irrelevant. The difficulty with having so many names, quite apart from any confusion which humans experience, is that naming conflicts can cause internal problems. This is an operating system dependent problem but, as a general rule, if we are forced to use more than one naming service, we must be careful to ensure complete consistency between them.

The only world-wide service in common use today is DNS (the Domain Name Service) whose common implementation is called BIND (Berkeley Internet Name Domain). This associates IP addresses with a list of names. Every host in the DNS has a *canonical name*, or official name, and any number of *aliases*. For instance, a host which runs several important services might have the canonical name

mother.domain.country

and aliases,

www.domain.country
ftp.domain.country

DNS binds a local network to the world-wide Internet in several important ways. It makes it possible for data belonging to organizations to be spread across the surface of the planet (or beyond) at any location, and yet still maintain a transparent naming structure. E-mail services use the DNS to route mail.

WINS (Windows Internet Name Service) was a proprietary system built by Microsoft for Windows. Since any local host can register data in this service, it was insecure and is therefore inadvisable in any trusted network. WINS has now been replaced by DNS as of Windows 2000.

Under Windows, each system has an alphanumeric name which is chosen during the installation. A domain server will provide an SID (security ID) for the name which helps prevent spoofing. When Windows boots it broadcasts the name across the network to see whether it is already in use. If the name is in use, the user of the workstation is prompted for a new name.

The security of a name service is of paramount importance, since so many other services rely on name services to establish identity. If one can subvert a name service, hosts can be tricked into trusting foreign hosts and security crumbles.

3.7 Common network sharing models

During the 1970s it was realized that expensive computer hardware could be used most cost-efficiently (by the maximum number of people) if it was available *remotely*, i.e. if one could communicate with the computer from a distant location and gain access to its resources. Inter-system communication became possible, in stages, through the use of modems and UUCP batch transfer and later through real-time wide area networks.

The large mainframe computers which served sometimes hundreds of users were painfully slow for interactive tasks, although they were efficient at batch processing. As hardware became cheaper many institutions moved towards a model of smaller computers coupled to file-servers and printers by a network. This solution was relatively cheap but had problems of its own. At this time the demise of the mainframe was predicted. Today, however, mainframe computers are very much alive for computationally intensive tasks, while the small networked workstation provides access to a world of resources via the Internet.

Dealing with networks is one of the most important aspects of system administration today. The network is our greatest asset and our greatest threat. In order to be a system administrator it is necessary to understand how and *why* networks are implemented, using a world-wide protocol: the Internet protocol family. Without getting too heavily bogged down in details which do not concern us at an elementary level, we shall explore these themes throughout the remainder of this material.

3.7.1 Constraints on infrastructure

Different operating systems support different ideas about how network services should be used. We are not always free to use the hardware resources as we would like. Operating system technologies restrict the kind of infrastructures it is possible to build in practice (see figure 2).

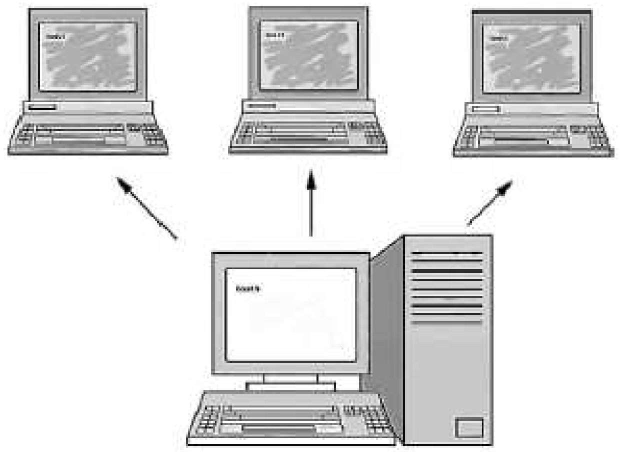


Figure 2: Some network infrastructure models single out a special server-host, which is used to consolidate network services and resources. Such centralization has many administrative advantages, but it concentrates load and can create a bottleneck.

• *Unix:* Much, if not all, of Unix's success can be attributed to the astonishing freedom which is granted to its users and administrators. Without a doubt, Unix-like operating systems are the most configurable and adaptable ever created. This has kept Unix at the forefront of new technology but has also created a class of operating systems rather like disorganized piles of treasure in Aladdin's cave.

Unix-like operating systems are not tied to any specific model for utilizing network resources, though vendors sometimes introduce specific technologies for sharing, which favor a particular kind of model. (This is almost viewed as a treasonable offence and is usually quickly rejected in favor of software which offers greater freedom.) Unix lets us decide how we want the network to look. Any Unix system can perform any function, as server, client or both. A Unix network is fully distributed; there is no requirement about centralization of resources, but central models are commonly used.Unix contains troves of tools for making many hosts work together and share resources, but each host can also be configured as a stand-alone system. Each host either has a fixed IP address, or can be assigned one dynamically at boot-time by a service such as *BOOTP* or *DHCP*.
• *Windows:* Windows networks are built around a specific model. There are two types of Windows system with separate software licenses: *workstations* and *servers*. Windows can work as a stand-alone system or as a workstation, integrated into a system of network services. Windows revolves around a model in w-hich programs are run on a local workstation, but where network services and resources are kept and run on a centralized server. IP addresses may be fixed or may be assigned automatically by a network service such as *BOOTP* or *DHCP*. Several Windows servers can coexist. Each server serves a logical group of hosts, users and services called a *domain*. Domains are now merged into an Active Directory model that provides a logical directory structure for network services. Client machines subscribe to as many domains as they wish or have permission to join. Windows supports two kinds of organizational groups: *workgroups* in which hosts share a simple peer-to-peer network, perhaps with Windows

9x machines, and *domains* which have a central authority through a domain server. Domains provide a common framework including user-ids (SIDs in Windows language), passwords and user profiles. Domains have a common user-database and a common security policy. Any host which subscribes to a domain inherits the users and the security policy of the domain. Windows domains can be simulated by Unix-like hosts.

Windows 2000 is a reincarnation of Windows NT 4.0. It redresses many of the shortcomings of the NT domain model by moving towards Novell-like directory services as its new model for resource organization. It allows remote services such as terminal login, which was only introduced as an afterthought in NT.

3.7.2 User preference storage

Software packages often allow users to store preferences about the way that software should look and behave. Such data are stored in some kind of information repository. Another issue for networked systems is where software preference data should be stored for users. There are two possibilities here which correspond approximately to the Unix approach and the Windows approach.

• *Windows:* Under earlier Windows, each user was assumed to have his or her own personal workstation which would not normally be used by other users. Current Windows versions allow logins by multiple users. Configuration data or preferences which the user selects are stored locally on the system disk in a location provided by the operating system. Later versions of Windows (NT, 2000, XP etc.) solve these problems by maintaining *user profiles* which are stored on the domain server in a \profiles subdirectory of the system-root. These data are copied into the local workstation when a user logs on to a domain server.

• *Unix/Shared:* Under Unix, each user sets up personal preferences in his or her personal *dot files* which are stored in private user space. More general global preferences are stored in a directory of the administrator's choice. Traditionally this has been the directory /etc.

The difficulties associated with having a fixed location for the configuration information which lies in the system files are several. In any single-user operating system, one user can overwrite another user's preferences simply by changing them since the system is not capable of telling the difference between users. This is a fundamental problem which indicates that single-user operating systems are basically unsuited to a shared environment. For a multi-user, networked world, the following points must be considered:

• When the operating system is reinstalled, configuration information can easily be lost or overwritten if they are stored in an operating system directory.
• In a distributed environment, where users might not sit at the same physical workstation day after day, the user's personal configuration data will not follow him or her from machine to machine.

On a Unix system, it is easy to specify the locations of configuration files in software and these can then be kept separate from operating system files, e.g. on a different disk partition so that they are immune to accidental deletion by system reinstallation. The main problem with Unix is the lack of any uniformity of approach. In the future, there might be a semblance of uniformity.

3.8 Local network orientation and analysis

A network community is an organism, working through the orchestrated cooperation of many parts. We need to understand its operation carefully in order to make it work well. The choices we make about the system can make it easy to understand, or difficult to understand, efficient or inefficient. This is the challenge of community planning.

Within a local area network, a top-down approach is useful for understanding host interrelationships. We therefore begin at the local network level, i.e. at the level of the collective society of machines.

In most daily situations, one starts with a network already in place, i.e. we do not have to build one from scratch. For an administrator, it is important to know what hardware one has to work with and where everything is to be found; how it is organized (or not) and so on.

Principle (Resource map). *A resource map of a site aids the* predictability *of the system by allowing an administrator to learn about the parts of the system, understand interrelationships and prepare a contingency plan for expected problems with the specific elements.*

Here is a checklist:

- How does the network physically fit together? (What is its topology?)
- How many different subnets does the network have?
- What are their network addresses?
- Find the router addresses (and the default routes) on each segment.
- What is the net mask?
- What hardware is there in the network? (hosts, printers etc.)
- Which function does each host/machine have on the network?
- Where are the key network services located?

Some hardware can be efficiently identified and queried using SNMP technology. Most newer network hardware supports some kind of querying using SNMP protocols. This is a form of network communication which talks directly to the device and extracts its hardware profile. Without SNMP, identifying hardware automatically is problematical. One author has proposed using the Unix log service syslogd to track hardware configurations. An overview of network services can sometimes be obtained using port-scanning software, such as nmap, though this should be agreed in advance to avoid misunderstandings. Many network intrusion attempts begin with port scans; these can make security conscious administrators nervous.

Of course, when automated methods fail, one can always resort to a visual inspection. In any event, an organization needs some kind of *inventory list* for the purpose of insurance or theft, if not merely for good housekeeping. A rough overview of all this information needs to be assembled in system administrators' minds, in order to understand the challenge ahead.

Having thought about the network in its entirety, we can drop down a level and begin to think about individual host machines. We need to know hosts both from the viewpoint of hardware and software.

• What kind of machines are on the network? What are their names and addresses and where are they? Do they have disks. How big? How much memory do they have? If they are PCs, which screen cards do they have?
• How many CPUs do the hosts have?
• What operating systems are running on the network?
• What kind of network cables are used?
• Where are hubs/repeaters/the router or other network control boxes located?
Who is responsible for maintaining them?
• What is the hierarchy of responsibility?

There is information about the local environment:

• What is the local time zone?
• What broadcast address convention is used? 255 or the older 0?
• Find the key servers on these networks.

– Where are the network disks located? Which machine are they attached to?
– Which name service is in use (DNS, NIS or NIS plus)?
– Where is the inevitable WWW/HTTP service? Who is running pirate servers?

Finding and recording this information is an important learning process, and the information gathered will prove invaluable for the task ahead. Of course, the information will change as time goes by. Networks are not static; they grow and evolve with time, so we must remain vigilant in pursuit of the moving target.

3.8.1 Network naming orientation

Familiarizing oneself with an organization's network involves analyzing the network's hosts and all of their interrelationships. It is especially important to know who is responsible for maintaining different parts of the network. It might be us or it might be someone else. We need to know whom to contact when something is going wrong over which we have no control ourselves. The most obvious way to view an organization is by its logical structure. This is usually reflected in the names of different machines and domains. Whom do we call if the Internet connection is broken? What service contracts exist on hardware, what upgrade possibilities are there on software? What system is in use for making backups? How does one obtain a backup should the need arise? In short, it is essential to know where to begin in solving any problem which might arise, and whom to call if the responsibility for a problem lies with someone else.

The Internet is permeated by a *naming scheme* which, naturally, is used to describe organizational groupings of Internet addresses. We can learn a lot by inspecting the name data for an organization. Indeed, many organizations now see this as a potential security hazard and conceal their naming strategies from outsiders. The Domain Name Service (DNS) is the Internet’s primary naming service. It not only allows us to name hosts, but also whole organizations, placing many different IP addresses under a common umbrella. The DNS is thus a hierarchical organization of machine names and addresses. Organizations are represented by *domains* and a domain is maintained either by or on behalf of each organization. Global domains are divided into countries, or groupings like .com and .org, and *sub-domains* are set up within larger domains, so that a useful name can be associated with the function of the organization. To analyze our own network, we begin by asking: who runs the DNS domain above ours?

For our organizational enquiry, we need an overview of the hosts which make up our organization. A host list can be obtained from the DNS using nslookup/dig or Nslookup etc. (unless that privilege has been revoked by the DNS administrator. If there are Unix systems on the network, one can learn a lot without physical effort by logging onto each machine and using the uname command to find out what OS is being used:

```
sunshine%  uname  -a
SunOS  nexus  5.9  Generic_112233-04  sun4u  sparc


gnu%  uname  -a
Linux gnu 2.4.10-4GB #1 Fri Sep 28 17:20:21 GMT 2001 i686 unknown
```

This tells us that host nexus is a SunOS kernel version 5.9 (colloquially known as Solaris 2.9) system with a sun4u series processor and that host gnu is a GNU/Linux system kernel version 2.4.10. If the uname command doesn’t exist, then the operating system is an old dinosaur from BSD 4.3 days and we have to find out what it is by different means. Try the commands arch and mach.

Knowing the operating system of a host is not sufficient. We also need to know what kind of resources the host has to offer the network, so that we can later plan
the distribution of services. Thus we need to dig deeper:

- How much memory does a host have? (Most systems print this when they boot. Sometimes the information can be coaxed out of the system in other ways.) What disks and other devices are in use?
- Use locate and find and which and whereis to find important directories and software. How is the software laid out?
- What software directories exist? /usr/local/bin, /local/bin?

• Do the Unix systems have a C compiler installed? This is often needed for installing software. Finding out information about other operating systems, such as Windows, which we cannot log onto is a tedious process. It must be performed by manual inspection, but the results are important nonetheless.

3.8.2 Using nslookup and dig

The nslookup program is for querying the Domain Name Service (DNS). On Unix it has now been officially deprecated and replaced by a new program, dig or host, in the source implementations of the BIND software. On Windows one has Nslookup. It is still in widespread use, however, both in Unix and Windows milieux. Moreover, IPv6 lookup does not work in all implementations of nslookup. The name service provides a mapping or relationship between Internet numbers and Internet names, and contains useful information about domains: both our own and others. The first thing we need to know is the domain name. This is the suffix part of the Internet name for the network. For instance, suppose our domain is called example.org. Hosts in this domain have names like hostname.example.org.

If you don't know your DNS domain name, it can probably be found by looking at the file /etc/resolv.conf on Unix hosts. For instance:

```
gnu% more /etc/resolv.conf
domain example.org
nameserver 192.0.2.10
nameserver 192.0.2.17
nameserver 192.0.2.244
```

Also most Unix systems have a command called domainname. This prints the name of the local Network Information Service (NIS) domain which is not the same thing as the DNS domain name (though, in practice, many sites would use the same name for both). Do not confuse the output of this command with the DNS domain name.

Once you know the domain name, you can find out the hosts which are registered in your domain by running the name service lookup program nslookup, or dig.

```
gnu% nslookup
Default Server: mother.example.org
Address: 192.0.2.10
>
```

nslookup always prints the name and the address of the server from which it obtains its information. Then you get a new prompt > for typing commands. Typing help provides a list of the commands which nslookup understands.

3.8.3 Ping and traceroute

The most basic tools for testing network connectivity are ping and traceroute (tracert on Windows). These tools determine network connectivity, host availability and overall latency. The ping command sends the network equivalent of a sonar ping to a remote interface:

```
 host$ ping www.gnu.org
 www.gnu.org is alive
```

The command returns with a simple message to say whether or not the interface is up, i.e. whether the host is online or not. The -s flag causes packets to be sent at regular intervals, with sequence numbers and latency timings visible. This is useful for gauging transit times and network line capacity:

```
 host$ ping -s www.gnu.org
          PING www.gnu.org (199.232.41.10) from 80.111.2.134 : 56(84) bytes of data.
64 bytes from www.gnu.org (199.232.41.10): seq=1 ttl=236 t=225.569 ms
64 bytes from www.gnu.org (199.232.41.10): seq=2 ttl=236 t=153.235 ms
```

Pings on free Unix-like operating systems behave like ping -s on older systems, i.e. it defaults into periodic transmission mode.

The traceroute command sends UDP packets to the destination, with stepwise incremental ‘time to live’ fields, which then provoke a ‘time out’ error at each router in turn, thus mapping out the route taken. Hosts that are attached to the same subnet do not normally pass through a router, thus there is a single hop which is directly to the destination address by unicast:

```
host$ /usr/sbin/traceroute pax.example.org
traceroute to pax.example.org (128.39.89.4), 30 hops max,
40 byte packets
          1 pax.example.org (128.39.89.4) 0.682 ms 0.414 ms 0.402 ms
```

3.8.4 Uniform resource locators (URLs)

URLs became well known, as a concept, in connection with the World Wide Web. The principle of referring to resources by a standardized name format can be adopted here too. Each operating system has a model for laying out its files in a standard pattern, but user files and local additions are usually left unspecified. Choosing a sound layout for data can make the difference between an incomprehensible chaos and a neat orderly structure. An orderly structure is useful not only for the users of the system, but also when making backups. Some of the issues are:

- Disk partitions are associated with drives or directory trees when connected to operating systems. These need names.
- Naming schemes for files and disks are operating system dependent.

• The name of a partition should reflect its function or contents.
• In a network the name of a partition ought to be a URL, i.e. contain the name of the host.
• It is good practice to consolidate file storage into a few special locations rather than spreading it out all over the network. Moreover, a basic principle in cataloging resources is:

Principle (One name for one object I). *Each unique resource requires a unique name, which labels it and describes its function.* with the corollary:

Corollary to principle (Aliases). *Sometimes it is advantageous to use aliases or pointers to unique objects so that a generic name can point to a specific resource. The number of aliases should be kept to a minimum, to avoid confusion.*

Data kept on many machines can be difficult to manage, compared with data collected on a few dedicated file-servers. Also, insecure operating systems offer files on a local disk no protection.
The URL model of file naming has several advantages. It means that one always knows the host-provider and function of a network resource. Also it falls nicely into a hierarchical directory pattern. For example, a simple but effective scheme is to use a three-level mount-point for adding disks: each user disk is mapped onto a directory with a name of the form

/site/host/content.

This scheme is adequate even for large organizations and can be extended in obvious ways. Others prefer to build up names around services, e.g.

/nfs/host/content One objection to the naming scheme above is that the use of the server name ties a resource to a particular server host, and thus makes it difficult to move resources around. Technologies like amd (automount), AFS, DFS (the Open Group's), and Dfs (Microsoft's) help address this issue and can make the filesystem based on a logical layout rather than an actual physical location. On the other hand, location independence can always be secured with aliases (symbolic) or with truly distributed filesystems. Moving actual resources is always a relatively non-trivial operation, and a naming scheme like that above yields clarity for a minimum of work.

In DOS-derived operating systems one does not have the freedom to 'mount' network filesystems into the structure of the local disk; network disks always

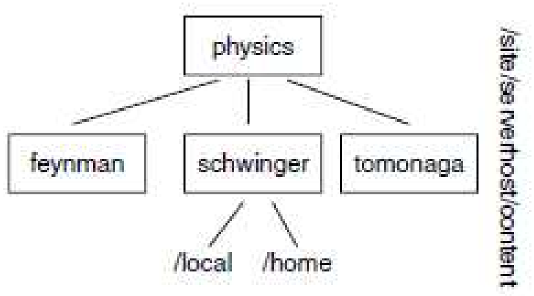


Figure 3: A universal naming scheme (URL) for network resources makes distributed data comprehensible

become a special 'drive', like H: or I: etc. It is difficult to make a consistent view of the disk resources with this system, however future Windows systems will have seamless integration and one can already use filesystems like the DFS on NT which do support this model.

Within an organization a URL structure provides a global naming scheme, like those used in true network filesystems like AFS and DFS. These use the name of the host on which a resource is physically located to provide a point of reference. This is also an excellent way of labeling backups of partitions since it is then immediately clear where the data belong. A few rules of thumb allow this naming scheme to live painlessly alongside traditional Unix naming schemes.

- When mounting a remote filesystem on a host, the client and server directories should always have exactly the same name, to avoid confusion and problems later.
- The name of every filesystem mount-point should be unique and tell us something meaningful about where it is located and what its function is.
- For tradition, one can invoke the corollary and use an alias to provide a generic reference point for specific resources. For instance, names like /usr/local can be used to point to more accurate designations like /site/ host/local. On different clients, the alias /usr/local might point to a filesystem on a single server, or to filesystems on many servers. The purpose of an alias is to hide this detail, while the purpose of the filesystem designation is to identify it. This satisfies all needs and is consistent.
- It doesn't matter whether software compiles the path names of special directories into software as long as we follow the points above.

For example, the following scheme was introduced at a College. The first link in the mount-point is the department of the organization or, in our case, the university faculty which the host belongs to; the second link is the name of the host to which the disk is physically connected, and the third and final link is a name which reflects the contents of the partition.
Some examples:

/site/hostname/content

/research/grumpy/local
/research/happy/home1
/research/happy/home2

/sales/slimy/home1

/physics/einstein/data
/biology/pauling/genome-db

The point of introducing this scheme was two-fold:

- To instantly be able to identify the server on which the disk resource physically resided.
- To instantly be able to identify the correct locations of files on backup tapes, without any special labeling of the tapes

.

System administrators are well known for strong opinions, and many practicing system administrators will strongly disagree with this practice. However, one should have an excellent reason to ignore a systematic approach.

3.8.5 Choosing server-hosts

Choosing the best host for a service is an issue with several themes. The main principles have to do with efficiency and security and can be summarized by the following questions.

- Does traffic have to cross subnet boundaries?
- Do we avoid unnecessary network traffic?
- Have we placed insecure services on unimportant hosts?

Service requests made to servers on different subnets have to be routed. This takes time and uses up switching opportunities which might be important on a heavily loaded network. Some services (like DNS) can be mirrored on each subnet, while others cannot be mirrored in any simple fashion. Unnecessary network traffic can be reduced by eliminating unnecessary dependencies of one service on another.

Example Suppose *we are setting up a file-server (WWW or FTP). The data which these servers will serve to clients lie on a disk which is physically attached to some host. If we place the file-server on a host which does not have direct physical access to the disks, then we must first use another network service (e.g. NFS) as a proxy in order to get the data from the host with the disk attached. Had we placed the file-server directly on the host with the disk, the intermediate step would have been unnecessary and we could approximately halve the amount of network traffic.*

We can codify this advice as a principle: avoid making one service reliant on another.

Principle (Inter-dependency). *The more dependent a service is, the more vulnerable it is to failure. With fewer dependencies, there are fewer possible failure modes, and therefore predictability and reliability are increased.*

Some services are already reliant on others, by virtue of their design. For example, most services are reliant on the DNS.

3.8.6 Distributed filesystems and mirroring

The purpose of a network is to share resources amongst many hosts. Making files available to all hosts from a common source is one of the most important issues in setting up a network community. There are three types of data which we have to consider separately:

- Users' home directories.
- Software or binary data (architecture specific).
- Other common data (architecture unspecific).

Since users normally have network accounts which permit them to log onto any host in the network, user data clearly have to be made available to all hosts.The same is not true of software, however. Software only needs to be shared between hosts running comparable operating systems. A Windows program will not run under GNU/Linux (even though they share a common processor and machine code), nor will an SCO Unix program run under Free BSD. It does not make sense to share binary filesystems between hosts, unless they share a common architecture. Finally, sharable data, such as manual information or architecture independent databases, can be shared between any hosts which specifically require access to them.

How are network data shared? There are two strategies:

- Use of a shared filesystem (e.g. NFS, AFS or Novell Netware).
- Remote disk mirroring.

Using a network filesystem is always possible, and it is a relatively cheap solution, since it means that we can minimize the amount of disk space required to store data, by concentrating the data on just a few servers. The main disadvantage with use of a network filesystem is that network access rates are usually much slower than disk access rates, because the network is slow compared with disks, and a server has to talk to many clients concurrently, introducing *contention* or competition for resources. Even with the aggressive caching schemes used by some network filesystems, there is usually a noticeable difference in loading files from the network and loading files locally.

We would like to minimize load on the network if possible. A certain amount of network traffic can be avoided by mirroring software rather than sharing with a network filesystem. Mirroring means copying every file from a source filesystem to a remote filesystem. This can be done during the night when traffic is low and, since software does not change often, it does not generate much traffic for upgrades after the initial copy.

Mirroring is cheap on network traffic, even during the night, During the daytime, when users are accessing the files, they collect them from the mirrors. This is both faster and requires no network bandwidth at all. Mirroring cannot apply to users' files since they change too often, while users are logged onto the system, but it applies very well to software. If we have disk space to spare, then mirroring software partitions can relieve the load of sharing.
There are various options for disk mirroring.

The benefits of mirroring can be considerable, but it is seldom practical to give every workstation a mirror of software. A reasonable compromise is to have a group of file-servers, synchronized by mirroring from a central source. One would expect to have at least one file-server per subnet, to avoid router traffic, money permitting.

SELF ASSESSMENT EXERCISES

1. What is the main principle at work in any cooperative enterprise, such as a network or community with limited resources?
2. Explain the role of policy in a community.
3. Are rules meant for humans comparable to rules meant for machines? Explain.
4. Draw a diagram of the physical topology of your local network, showing
routers, switches, cables and other hardware.

4.0 CONCLUSION

We cannot learn anything about a community of networked computer systems without knowing where all the machines are, both physically and in the network, what their purposes are, and how they interrelate to one another. Normally we do not start out by building a network of computers from nothing, rather we inherit an existing network, serviceable or not; thus the first step is to acquaint ourselves with the system at hand.

5.0 SUMMARY

System administration is not just about machines and individuals, it is about communities. There is the local community of users on multi-user machines; then there is the local area network community of machines at a site. Finally, there is the global community of all machines and networks in the world.

At the heart of all cooperation in a community is a system of *centralization* and *delegation*. No program or entity can do everything alone, nor is everyone expected to do so. It makes sense for certain groups to specialize in performing certain jobs. That is the function of a society and good management.

Whenever computers are coupled together, there is a need for each to have an individual and unique identity. This need has been recognized many times, by different system developers, and the result is that today's computer systems can have many different names which identify them in different contexts. The outcome is confusion. For Internet-enabled machines, the IP address of the host is usually sufficient for most purposes.

A network community is an organism, working through the orchestrated cooperation of many parts. We need to understand its operation carefully in order to make it work well. The choices we make about the system can make it easy to understand, or difficult to understand, efficient or inefficient. This is the challenge of community planning..

The most basic tools for testing network connectivity are ping and traceroute (tracert on Windows). These tools determine network connectivity, host availability and overall latency. The ping command sends the network equivalent of a sonar ping to a remote interface:

URLs became well known, as a concept, in connection with the World Wide Web. The principle of referring to resources by a standardized name format can be adopted here too. Each operating system has a model for laying out its files in a standard pattern, but user files and local additions are usually left unspecified. Choosing a sound layout for data can make the difference between an incomprehensible chaos and a neat orderly structure. An orderly structure is useful not only for the users of the system, but also when making backups.

6.0 TUTOR MARKED ASSIGNMENTS

1. What are the pros and cons of making a network completely uniform in the choice of hardware and software?

2a. What is a MAC address?
 b. What is the service that relates Internet Domain Names to IP addresses?
 c. What is the service that relates IP addresses to MAC addresses?

3. What is meant by a name service? Name two widely used name services that contain IP addresses and one that contains Ethernet addresses.

4. What is the Domain Name Service? How do hosts depend on this service?
Suppose that the data in the DNS could be corrupted. Explain how this could be a security risk.

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

3. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley

4. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

5. Subramanian, M. (2000). Network Management: Principles and Practice, Addison-Wesley

UNIT 3: HOST MANAGEMENT

1.0 INTRODUCTION

The foregoing units have explored the basics of how hosts need to function within a network community; we are now sufficiently prepared to turn our attention to the role of the individual host within such a network.

2.0 OBJECTIVES

At the end of the unit, you should be able to:

- Understand the procedures for starting up and shutting down computers safely
- Know how to install operating systems and software
- Know how to Configure and personalize workstations
- Understand the concept of dual-homed host
- Understand how to do disk partitioning
- Understand how to format and build filesystems

3.0 MAIN CONTENT

3.1 Computer startup and shutdown

The two most fundamental operations which one can perform on a host are to start it up and to shut it down. With any kind of mechanical device with moving parts, there has to be a procedure for shutting it down. One does not shut down any machine in the middle of a crucial operation, whether it be a washing machine in the middle of a program, an aircraft in mid-flight, or a computer writing to its disk.

With a multitasking operating system, the problem is that it is never possible to predict when the system will be performing a crucial operation in the background. For this simple reason, every multitasking operating system provides a procedure for shutting down safely. A safe shutdown avoids damage to disks by mechanical interruption, but it also synchronizes hardware and memory caches, making sure that no operation is left incomplete.

3.3.1 Booting Unix

Normally it is sufficient to switch on the power to boot a Unix-like host. Sometimes you might have to type ‘boot’ or ‘b’ to get it going. Unix systems can boot in several different modes or *run levels*. The most common modes are called multi-user mode and single-user mode. On different kinds of Unix, these might translate into run-levels with different numbers, but there is no consensus. In single-user mode no external logins are permitted. The purpose of single-user mode is to allow the system administrator access to the system without fear of interference from other users. It is used for installing disks or when repairing filesystems, where the presence of other users on the system would cause problems.

The Unix boot procedure is controlled entirely by the init program; init reads a configuration file called /etc/inittab. On older BSD Unices, a file called /etc/rc meaning 'run commands' and subsidiary files like rc.local was then called to start all services. These files were no more than shell scripts. In the System V approach, a directory called (something like) /etc/rc.d is used to keep one script per service. /etc/inittab defines a number of run-levels, and starts scripts depending on what run-level you choose. The idea behind inittab is to make Unix installable in packages, where each package can be started or configured by a separate script. Which packages get started depends on the run-level you choose.

The default form for booting is to boot in multi-user mode. We have to find out how to boot in single-user mode on our system, in case we need to repair a disk at some point.

3.3.2 Shutting down Unix

Anyone can start a Unix-like system, but we have to be an administrator or 'superuser' to shut one down correctly. Of course, one could just pull the plug, but this can ruin the disk filesystem. Even when no users are touching a keyboard anywhere, a Unix system can be writing something to the disk – if we pull the plug, we might interrupt a crucial write-operation which destroys the disk contents. The correct way to shut down a Unix system is to run one of the following programs.

- halt: Stops the system immediately and without warning. All processes are killed with the TERM-inate signal 15 and disks are synchronized.

- reboot: As halt, but the system reboots in the default manner immediately.

- shutdown: This program is the recommended way of shutting down the system. It is just a friendly user-interface to the other programs, but it warns the users of the system about the impending shutdown and allows them to finish what they are doing before the system goes down.

Here are some examples of the shutdown command. The first is from BSD Unix:

```
shutdown -h +3 "System halting in three minutes, please log out"

shutdown -r +4 "System rebooting in four minutes"
```

The -h option implies that the system will halt and not reboot automatically. The-r option implies that the system will reboot automatically. The times are specified in minutes.

System V Unix R4 (e.g. Solaris) has a different syntax which is based on its system of run-levels. The shutdown command allows one to switch run-levels in a very general way. One of the run-levels is the 'not running' or 'halt' run-level. To halt the system, we have to call this.

```
shutdown -i 5 -g 120 "Powering down os...."
```

The -i 5 option tells SVR4 to go to run-level 5, which is the power-off state. Run-level 0 would also suffice here. The -g 120 option tells shutdown to wait for a grace-period of 120 seconds before shutting down. Note that Solaris also provides a BSD version of shutdown in /usr/ucb.

Never assume that the run-levels on one system are the same as those on another.

3.3.3 Booting and shutting down Windows

Booting and shutting down Windows is a trivial matter. To boot the system, it is simply a matter of switching on the power. To shut it down, one chooses shutdown from the Start Menu.

There is no direct equivalent of single-user mode for Windows, though 'secure mode' is sometimes invoked, in which only the essential device drivers are loaded, if some problem is suspected. To switch off network access on a Windows server so that disk maintenance can be performed, one must normally perform a reboot and connect new hardware while the host is down. Filesystem checks are performed automatically if errors are detected. The plug'n'play style automation of Windows removes the need for manual work on filesystems, but it also limits flexibility.

The Windows boot procedure on a PC begins with the BIOS, or PC hardware. This performs a memory check and looks for a boot-able disk. A boot-able disk is one which contains a master boot record (MBR). Normally the BIOS is configured to check the floppy drive A: first and then the hard-disk C: for a boot block. The boot block is located in the first sector of the boot-able drive. It identifies which partition is to be used to continue with the boot procedure. On each primary partition of a boot-able disk, there is a boot program which 'knows' how to load the operating system it finds there. Windows has a menu-driven boot manager program which makes it possible for several OSs to coexist on different partitions.

Once the disk partition containing Windows has been located, the program NTLDR is called to load the kernel. The file BOOT.INI configures the defaults for the boot manager. After the initial boot, a program is run which attempts to automatically detect new hardware and verify old hardware. Finally the kernel is loaded and Windows starts properly.

3.4 Configuring and personalizing workstations

Permanent, read–write storage changed PCs from expensive ping-pong games into tools for work as well as pleasure. Today, disk space is so cheap that it is not uncommon for even personal workstations to have several hundreds of gigabytes of local storage.

Flaunting wealth is the sport of the modern computer owner: more disk, more memory, better graphics. Why? Because it's there. This is the game of free enterprise, encouraged by the availability of home computers and personal workstations. Not so many years before such things existed, however, computers only existed as large multiuser systems, where hundreds of users shared a few kilobytes of memory and a processor no more powerful than a now arthritic PC. Rational resource sharing was not just desirable, it was the only way to bring computing to ordinary users. In a network, we have these two conflicting interests in the balance.

3.4.1 Personal workstations or 'networkstations'?

Today we are spoiled, often with more resources than we know what to do with. Disk space is a valuable resource which can be used for many purposes. It would be an ugly waste to allow huge areas of disk to go unused, simply because small disks are no longer manufactured; but, at the same time, we should not simply allow anyone to use disk space as they please, just because it is there.

Operating systems which have grown out of home computers (Windows and MacIntosh) take the view that, whatever is left over of disk resources is for the local owner to do with as he or she pleases. This is symptomatic of the idea that one computer belongs to one user. In the world of the network, this is an inflexible model. Users move around organizations; they ought not to be forced to take their hardware with them as they move. Allowing users to personalize workstations is thus a questionable idea in a network environment.

Network sharing allows us to make disk space available to all hosts on a network, e.g. with NFS, Netware or DFS. This allows us to make disk space available to all hosts. There are positives and negatives with sharing, however. If sharing was a universal panacea, we would not have local disks: everything would
be shared by the network. This approach has been tried: diskless workstations, network computers and X-terminals have all flirted with the idea of keeping all disk resources in one place and using the network for sharing. Such systems have been a failure: they perform badly, are usually more expensive than an
off-the-shelf PC and they simply waste a different resource: network bandwidth. Some files are better placed on a local disk: namely the files which are needed often, such as the operating system and temporary scratch files, created in the processing of large amounts of data.

In organizing disk space, we can make the best use of resources, and separate:

- Space for the operating system.
- Space which can be shared and made available for all hosts.
- Space which can be used to optimize local work, e.g. temporary scratch space, space which can be used to optimize local performance (avoid slow networking).
- Space which can be used to make distributed backups, for multiple redundancies.

These independent areas of use need to be separated from one another, by partitioning disks.

3.4.2 Partitioning

Disks can be divided up into partitions. Partitions physically divide the disk surface into separate areas which do not overlap. The main difference between two partitions on one disk and two separate disks is that partitions can only be accessed one at a time, whereas multiple disks can be accessed in parallel.

Disks are partitioned so that files with separate purposes cannot be allowed to spill over into one another's space. Partitioning a disk allows us to reserve a fixed amount of space for a particular purpose, safe in the knowledge that nothing else will encroach on that space. For example, it makes sense to place the operating system on a separate partition, and user data on another partition. If these two independent areas shared common space, the activities of users could quickly choke the operating system by using up all of its workspace.

In partitioning a system, we have in mind the issues described in the previous section and try to size partitions appropriately for the tasks they will fulfill. Here are some practical points to consider when partitioning disks:

• Size partitions appropriately for the jobs they will perform. Bear in mind that operating system upgrades are almost always bigger than previous versions, and that there is a general tendency for everything to grow.
• Bear in mind that RISC (e.g. Sun Sparc) compiled code is much larger than CISC compiled code (e.g. software on an Intel architecture), so software will take up more space on a RISC system.
• Consider how backups of the partitions will be made. It might save many complications if disk partitions are small enough to be backed up in one go with a single tape, or other backup device.

Choosing partitions optimally requires both experience and forethought. Thumb rules for sizing partitions change constantly, in response to changing RAM requirements and operating system sizes, disk prices etc. In the early 1990s many sites adopted diskless or partially diskless solutions, thus centralizing disk resources. In today's climate of ever cheaper disk space, there are few limitations left.

Disk partitioning is performed with a special program. On PC hardware, this is called fdisk or cfdisk. On Solaris systems the program is called, confusingly, format. To repartition a disk, we first edit the partition tables. Then we have to write the changes to the disk itself. This is called *labelling* the disk. Both of these tasks are performed from the partitioning programs. It is important to make sure manually that partitions do not overlap. The partitioning programs do not normally help us here. If partitions overlap, data will be destroyed and the system will sooner or later get into deep trouble, as it assumes that the overlapping area can be used legitimately for two separate purposes.

Partitions are labelled with logical device names in Unix. As one comes to expect, these are different in every flavor of Unix. The general pattern is that of a separate device node for each partition, in the /dev directory, e.g. /etc/sd1a, /etc/sd1b, /dev/dsk/c0t0d0s0 etc.

The introduction of meta-devices and logical volumes in many operating systems allows one to ignore disk partitions to a certain extent. Logical volumes provide seamless integration of disks and partitions into a large virtual disk which can be organized without worrying about partition boundaries. This is not always desirable, however. Sometimes partitions exist for protection, rather than merely for necessity.

3.4.3 Formatting and building filesystems

Disk formatting is a way of organizing and finding a way around the surface of a disk. It is a little bit like painting parking spaces in a car park. We could make a car park in a field of grass, but everything would get rapidly disorganized. If we paint fixed spaces and number them, then it is much easier to organize and reuse space, since people park in an orderly fashion and leave spaces of a standard, reusable size. On a disk surface, it makes sense to divide up the available space into sectors or blocks. The way in which different operating systems choose to do this differs, and thus one kind of formatting is incompatible with another.

The nomenclature of formatting is confused by differing cultures and technologies. Modern hard disks have intelligent controllers which can map out the disk surface independently of the operating system which is controlling them. This means that there is a kind of factory formatting which is inherent to the type of disk. For instance, a SCSI disk surface is divided up into *sectors*. An operating system using a SCSI disk then groups these sectors into new units called *blocks* which are a more convenient size to work with, for the operating system. With the analogy above, it is a little like making a car park for trucks by grouping parking spaces for cars. It also involves a new set of labels. This regrouping and labeling procedure is called *formatting* in PC culture and is called *making a filesystem*. Making a filesystem also involves setting up an infrastructure for creating and naming files and directories. A filesystem is not just a labeling scheme, it also provides functionality.

If a filesystem becomes damaged, it is possible to lose data. Usually filesystem checking programs called disk doctors, e.g. the Unix program fsck (filesystem check), can be used to repair the operating system's map of a disk. In Unix filesystems, data which lose their labeling get placed for human inspection in a special directory which is found on every partition, called lost+found.

The filesystem creation programs for different operating systems go by various names. For instance, on a Sun host running SunOS/Solaris, we would create a filesystem on the zeroth partition of disk 0, controller zero with a command like this to the raw device:

```
newfs -m 0 /dev/rdsk/c0t0d0s0
```

The newfs command is a friendly front-end to the mkfs program. The option –m 0, used here, tells the filesystem creation program to reserve zero bytes of special space on the partition. The default behavior is to reserve ten percent of the total partition size, which ordinary users cannot write to. This is an old mechanism for preventing filesystems from becoming too full. On today's disks, ten percent of a partition size can be many files indeed, and if we partition our cheap, modern disks correctly, there is no reason not to allow users to fill them up completely.

This partition is then made available to the system by mounting it. This can either be performed manually:

```
 mount  /dev/dsk/c0t0d0s0  /mountpoint/directory
```

or by placing it in the filesystem table /etc/vfstab.

GNU/Linux systems have the mkfs command, e.g.

```
Mkfs  /dev/hda1
```

The filesystems are registered in the file /etc/fstab. Other Unix variants register disks in equivalent files with different names, e.g. HPUX in /etc/checklist (prior to 10.x) and AIX in /etc/filesystems.

On Windows systems, disks are detected automatically and partitions are assigned to different logical drive names. Drive letters C: to Z: are used for non-floppy disk devices. Windows assigns drive letters based on what hardware it finds at boot-time. Primary partitions are named first, then each secondary partition is assigned a drive letter. The format program is used to generate a filesystem on a drive. The command format /fs:ntfs /v:spare F: would create an NTFS filesystem on drive F: and give it a volume label 'spare'.
The older, insecure filesystem FAT can also be chosen, however this is not recommended. The GUI can also be used to partition and format inactive disks.

3.4.4 Swap space

In Windows operating systems, virtual memory uses filesystem space for saving data to disk. In Unix-like operating systems, a preferred method is to use a whole, unformatted partition for virtual memory storage.

A virtual memory partition is traditionally called the swap partition, though few modern Unix-like systems 'swap' out whole processes, in the traditional sense. The swap partition is now used for *paging*. It is virtual memory scratch space, and uses direct disk access to address the partition. No filesystem is needed, because no functionality in terms of files and directories is needed for the paging system.

The amount of available RAM in modern systems has grown enormously in relation to the programs being run. Ten years ago, a good rule of thumb was to allocate a partition twice the size of the total amount of RAM for paging. On heavily used login servers, this would not be enough. Today, it is difficult to give any firm guidelines, since paging is far less of a problem due to extra RAM, and there is less uniformity in host usage.

3.4.5 Filesystem layout

We have no choice about the layout of the software and support files which are installed on a host as part of 'the operating system'. This is decided by the system designers and cannot easily be changed. Software installation, user registration and network integration all make changes to this initial state, however. Such additions to the system are under the control of the system administrator and it is important to structure these changes according to logical and practical principles which we shall consider below.

A working computer system has several facets:

- The operating system software distribution,
- Third party software,
- Users' files,
- Information databases,
- Temporary scratch space.

These are logically separate because:

- They have different functions,
- They are maintained by different sources,
- They change at different rates,
- A different policy of backup is required for each.

Most operating systems have hierarchical filesystems with directories and subdirectories. This is a powerful tool for organizing data. Disks can also be divided up into partitions. Another issue in sizing partitions is how you plan to make a backup of those partitions. To make a backup you need to copy all the data to some other location, traditionally tape. The capacity of different kinds of tape varies quite a bit, as does the software for performing backups.

The point of directories and partitions is to separate files so as not to mix together things which are logically separate. There are many things which we might wish to keep separate: for example,

- User home directories
- Development work
- Commercial software
- Free software
- Local scripts and databases.

One of the challenges of system design is in finding an appropriate directory structure for all data which are not part of the operating system, i.e. all those files which are locally maintained.

Principle (Separation I). *Data which are separate from the operating system should be kept in a separate directory tree, preferably on a separate disk partition. If they are mixed with the* *operating system file tree it makes reinstallation or upgrade of the operating system unnecessarily difficult.*

The essence of this is that it makes no sense to mix logically separate file trees. For instance, users' home directories should never be on a common partition with the operating system. Indeed, filesystems which grow with a life of their own should never be allowed to consume so much space as to throttle the normal operation of the machine.

These days there are few reasons for dividing the files of the operating system distribution into several partitions (e.g. /, /usr). Disks are large enough to install the whole operating system distribution on a single independent disk or partition. If you have done a good job of separating your own modifications from the system distribution, then there is no sense in making a backup of the operating system distribution itself, since it is trivial to reinstall from source (CD-ROM or ftp file base). Some administrators like to keep /var on a separate partition, since it contains files which vary with time, and should therefore be backed up.

Operating systems often have a special place for installed software. Regrettably they often break the above rule and mix software with the operating system's file tree. Under Unix-like operating systems, the place for installed third party software is traditionally /usr/local, or simply /opt. Fortunately under Unix, separate disk partitions can be woven anywhere into the file tree on a directory boundary, so this is not a practical problem as long as everything lies under a common directory. In Windows, software is often installed in the same directory as the operating system itself; also Windows does not support partition mixing in the same way as Unix so the reinstallation of Windows usually means reinstallation of all the software as well.

Data which are installed or created locally are not subject to any constraints, however; they may be installed anywhere. One can therefore find a naming scheme which gives the system logical clarity. This benefits users and management issues. Again we may use directories for this purpose. Operating systems which descended from DOS also have the concept of drive numbers like A:, B:, C: etc. These are assigned to different disk partitions. Some Unix operating systems have virtual filesystems which allow one to add disks transparently without ever reaching a practical limit. Users never see partition boundaries. This has both advantages and disadvantages since small partitions are a cheap way to contain groups of misbehaving users, without resorting to disk quotas.

3.4.6 Object orientation: separation of independent issues

The computing community is currently riding a wave of affection for object orientation as a paradigm in computer languages and programming methods. Object orientation in programming languages is usually presented as a fusion of two independent ideas: classification of data types and access control based on scope. The principle from which this model has emerged is simpler than this, however: it is simply the observation that information can be understood and organized most efficiently if *logically independent* items are kept separate.3 This simple idea is a powerful discipline, but like most disciplines it requires a strong will on the part of a system administrator in order to avoid a decline into chaos. We can restate the earlier principle about operating system separation now more generally:

Principle (Separation II). *Data which are logically separate belong in separate directory trees, perhaps on separate filesystems.*

The basic filesystem objects, in order of global to increasingly local, are *disk partition*, *directory* and *file*. As system administrators, we are not usually responsible for the contents of files, but we do have some power to decide their organization by placing them in carefully labelled directories, within partitions. Partitions are useful because they can be dumped (backed-up to tape, for instance) as independent units. Directories are good because they hide and group related files into units.

Many institutions make backups of the whole operating system partition because they do not have a system for separating the files which they have modified, or configured specially. The number of actual files one needs to keep is usually small. For example

- The password and group databases
- Kernel configuration
- Files in /etc like services, default configurations files
- Special startup scripts.

It is easy to make a copy of these few files in a location which is independent of the locations where the files actually need to reside, according to the rules of the operating system.

A good solution to this issue is to make *master copies* of files like /etc/group, /etc/services, /etc/sendmail.cf etc., in a special directory which is separate from the OS distribution. For example, you might choose to collect all of these in a directory such as /local/custom and to use a script, or cfengine to make copies of these master files in the actual locations required by the operating system. The advantages to this approach are

- RCS version control of changes is easy to implement
- Automatic backup and separation
- Ease of distribution to other hosts.

The exception to this rule must be the password database /etc/passwd which is actually altered by an operating system program /bin/passwd rather than the system administrator. In that case the script would copy from the system partition to the custom directory.

Keeping a separate disk partition for software that you install from third parties makes clear sense. It means that you will not have to reinstall that software later when you upgrade your operating system. The question then arises as to how such software should be organized within a separate partition.

Traditionally, third party software has been installed in a directory under /usr/local or simply /local. Software packages are then dissected into libraries, binaries and supporting files which are installed under /local/lib, /local/bin and /local/etc, to mention just a few examples. This keeps third party software separate from operating system software, but there is no separation of the third party software. Another solution would be to install one software package per directory under /local.

3.6 Installation of the operating system

The installation process is one of the most destructive things we can do to a computer. Everything on the disk will disappear during the installation process. One should therefore have a plan for restoring the information if it should turn out that reinstallation was in error.

Today, installing a new machine is a simple affair. The operating system comes on some removable medium (like a CD or DVD) that is inserted into the player and booted. One then answers a few questions and the installation is done.

Operating systems are now large so they are split up into packages. One is expected to choose whether to install everything that is available or just certain packages. Most operating systems provide a package installation program which helps this process.

In order to answer the questions about installing a new host, information must be collected and some choices made:

- We must decide a name for each machine.
- We need an unused Internet address for each.
- We must decide how much virtual memory (swap) space to allocate.
- We need to know the local netmask and domain name.
- We need to know the local timezone.

We might need to know whether a Network Information Service (NIS) or Windows domain controller is used on the local network; if so, how to attach the new host to this service. When we have this information, we are ready to begin.

3.6.3 Windows

One inserts a CD-ROM and boots. Here it is preferable to begin with an already partitioned hard-drive (the installation program is somewhat ambiguous with regard to partitions). On rebooting, we are asked whether we wish to install Windows anew, or repair an existing installation. This is rather like the GNU/Linux rescue disk. Next we choose the filesystem type for Windows to be installed on, either DOS or NTFS. There is clearly only one choice: installing on a DOS partition would be irresponsible with regard to security. Choose NTFS.

Windows reboots several times during the installation procedure, though this has improved somewhat in recent versions. The first time around, it converts its default DOS partition into NTFS and reboots again. Then the remainder of the installation proceeds with a graphical user interface. There are several installation models for Windows workstations, including regular, laptop, minimum and custom. Having chosen one of these, one is asked to enter a license key for the operating system. The installation procedure asks us whether we wish to use DHCP to configure the host with an IP address dynamically, or whether a static IP address will be set. After various other questions, the host reboots and we can log in as Administrator.

Windows service packs are patch releases which contain important upgrades. These are refreshingly trivial to install on an already-running Windows system. One simply inserts them into the CD-ROM drive and up pops the Explorer program with instructions and descriptions of contents. Clicking on the install link starts the upgrade. After a service pack upgrade, Windows reboots predictably and then we are done. Changes in configuration require one to reinstall service packs, however.

3.6.4 Dual boot

There are many advantages to having both Windows and GNU/Linux (plus any other operating systems you might like) on the same PC. This is now easily achieved with the installation procedures provided by these two operating systems. It means, however, that we need to be able to choose the operating system from a menu at boot time. The boot-manager GRUB that is now part of GNU/Linux distributions performs this task very well, so one scarcely needs to think about this issue anymore. Note, however, that it is highly advisable to install Windows before installing GNU/Linux, since the latter tends to have more respect for the former than vice versa! GNU/Linux can preserve an existing Windows partition, and even repartition the disk appropriately.

3.6.6 Diskless clients

Diskless workstations are, as per the name, workstations which have no disk at all. They are now rare, but with the increase of network speeds, they are being discussed again in new guises such as ‘thin clients’.

Diskless workstations know absolutely nothing other than the MAC address of their network interface (Ethernet address). In earlier times, when disks were expensive, diskless workstations were seen as a cheap option. Diskless clients require disk space on a server-host in order to function, i.e. some other host which does have a disk, needs to be a disk server for the diskless clients. Most vendors supply a script for creating diskless workstations. This script is run on the server-host.

When a diskless system is switched on for the first time, it has no files and knows nothing about itself except the Ethernet address on its network card. It proceeds by sending a RARP (reverse address resolution protocol) or BOOTP or DHCP request out onto the local subnet in the hope that a server (in.rarpd) will respond by telling it its Internet address. The server hosts must be running two services: rpc.boot paramd and tftpd, the trivial file transfer program. This is another reason for arguing against diskless clients: these services are rather insecure and could be a security risk for the server host. A call to the rpc.bootparamd daemon transfers data about where the diskless station can find a server, and what its swap-area and root directory are called in the file tree of this server. The root directory and swap file are mounted using the NFS. The diskless client loads its kernel from its root directory and thereafter everything proceeds as normal. Diskless workstations swap to files rather than partitions. The command mkfile is used to create a fixed-size file for swapping.

3.6.7 Dual-homed host

A host with two network interfaces, both of which are coupled to a network, is called a dual-homed host. Dual-homed hosts are important in building *firewalls* for network security. A host with two network interfaces can be configured to automatically forward packets between the networks (act as a bridge) or to block such forwarding. The latter is normal in a firewall configuration, where it is left to proxy software to forward packets only after some form of inspection procedure. Most vendor operating systems will configure dual-network interfaces automatically, with forwarding switched off. One must then decide how the IP addresses are to be registered in the DNS service. Will the host have the same name on both interfaces, or will it have a different name?

SELF ASSESSMENT EXERCISES

1. Personal workstations or 'networkstations'? Discuss.
2. What are the procedures for shutting down computers safely at your site?
3. How do startup and shutdown procedures differ between Unix and Windows?
4. What different filesystems are in use on Windows hosts? What are the pros and cons of each?

4.0 CONCLUSION

We have seen in this unit that If we focus on too small a part of the entire system initially, time and effort can be wasted configuring hosts in a way that does not take into account the cooperative aspects of the network. That would be a recipe for failure and only a prelude to later reinstallation.

5.0 SUMMARY

Most vendors will only provide immediate support for individual hosts or, in the best case, clusters of hosts manufactured by them. They will almost never address the issue of total network solutions, without additional cost, so their recommendations often fall notably short of the recommendable in a real network. We have to be aware of the big picture when installing and configuring hosts.

The two most fundamental operations which one can perform on a host are to start it up and to shut it down. With any kind of mechanical device with moving parts, there has to be a procedure for shutting it down. Anyone can start a Unix-like system, but we have to be an administrator or 'superuser' to shut one down correctly. Of course, one could just pull the plug, but this can ruin the disk filesystem. Booting and shutting down Windows is a trivial matter. To boot the system, it is simply a matter of switching on the power. To shut it down, one chooses shutdown from the Start Menu.

The installation process is one of the most destructive things we can do to a computer. Everything on the disk will disappear during the installation process. One should therefore have a plan for restoring the information if it should turn out that reinstallation was in error.

A host with two network interfaces, both of which are coupled to a network, is called a dual-homed host. Dual-homed hosts are important in building *firewalls* for network security.

Software installation is a similar problem to that of operating system installation. However, third party software originates from a different source than the operating system; it is often bound by license agreements and it needs to be distributed around the network. Some software has to be compiled from source. We therefore need a thoughtful strategy for dealing with software.

6.0 TUTOR-MARKED ASSIGNMENTS

1. How does object orientation, as a strategy, apply to system administration?

2a. List the different ways to install an operating system on a new computer from
a source.
  b. What is meant by a dual-homed host?

3. Describe a checklist or strategy for familiarizing yourself with the layout of a new operating system file hierarchy.

4. Describe how to install Unix software from source files.

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Burke, J. R.(2004). Network Management Concepts and Practice: a Hands-on Approach. Pearson.

3. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

4. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley.

5. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

6. Subramanian, M. (2000). Network Management: Principles and Practice, Addison-Wesley.

UNIT 4: USER MANAGEMENT

1.0 INTRODUCTION

Without users, there would be few challenges in system and network administration. Users are both the reason that computers exist and their greatest threat. This unit focuses on how to interface humans to computers.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- Discuss the various issues in user management
- State the nine-step approach to user support
- Know why it is important to identify users by their username.
- Know the role password plays in identifying users
- Know the main health risks in the use of computers.

3.0 MAIN CONTENT

3.1 Issues

User management is about interfacing humans to computers. This brings to light a number of issues:

- Accounting: registering new users and deleting old ones.
- Comfort and convenience.
- Support services.
- Ethical issues.
- Trust management and security.

Some of these (account registration) are technological, while others (support services) are human issues. Comfort and convenience lies somewhere in between.
User management is important because the system exists to be used by human beings, and they are both friend and enemy.

3.2 User registration

One of the first issues on a new host is to issue accounts for users. Surprisingly this is an area where operating system designers provide virtually no help. The tools provided by operating systems for this task are, at best, primitive and are rarely suitable for the task without considerable modification. For small organizations, user registration is a relatively simple matter. Users can be registered at a centralized location by the system manager, and made available to all of the hosts in the network by some sharing mechanism, such as a login server, distributed authentication service or by direct copying of the data. There are various mechanisms for doing this, and we shall return to them below.

For larger organizations, with many departments, user registration is much more complicated. The need for centralization is often in conflict with the need for delegation of responsibility. It is convenient for autonomous departments to be able to register their own users, but it is also important for all users to be registered under the umbrella of the organization, to ensure unique identities for the users and flexibility of access to different parts of the organization. What is needed is a solution which allows local system managers to be able to register new users in a global user database. User account administration has been discussed many times. The special problems of each institution and work environment are reflected in these works.

PC server systems like NT and Netware have an apparent advantage in this respect. By forcing a particular administration model onto the hosts in a network, they can provide straightforward delegation of user registration to anyone with domain credentials. Registration of single users under NT can be performed remotely from a workstation, using the

```
net user username password /ADD /domain
```

command. While most Unix-like systems do not provide such a ready-made tool, many solutions have been created by third parties. The price one pays for such convenience is an implicit *trust relationship* between the hosts. Assigning new user accounts is a security issue, thus to grant the right of a remote user to add new accounts requires us to trust the user with access to that facility.

It is rather sad that no acceptable, standardized user registration methods have been widely adopted. This must be regarded as one of the unsolved problems of system administration. Part of the problem is that the requirements of each organization are rather different. Many Unix-like systems provide shell scripts or user interfaces for installing new users, but most of these scripts are useless, because they follow a model of system layout which is inadequate for a network environment, or for an organization's special needs.

3.2.1 Local and network accounts

Most organizations need a system for centralizing passwords, so that each user will have the same password on each host on the network. In fixed model computing environments such as NT or Novell Netware, where a login or domain server is used, this is a simple matter. In larger organizations with many departments or sub-domains it is more difficult.

Both Unix and NT support the creation of accounts locally on a single host, or 'globally' within a network domain. With a local account, a user has permission to use only the local host. With a network account, the user can use any host which belongs to a network *domain*. Local accounts are configured on the local host itself. Unix registers local users by added them to the files /etc/passwd and /etc/shadow. In NT the Security Accounts Manager (SAM) is used to add local accounts to a given workstation.

For network accounts, Unix-like systems have widely adopted Sun Microsystems’ Network Information Service (NIS), formerly called Yellow Pages or simply YP, though this is likely to be superceded and replaced by the more widely accepted standard LDAP in the next few years. The NIS-plus service was later introduced to address a number of weaknesses in NIS, but this has not been widely adopted. NIS is reasonably effective at sharing passwords, but it has security implications: encrypted passwords are distributed in the old password format, clearly visible, making a mockery of shadow password files. NIS users have to be registered locally as users on the master NIS server; there is no provision for remote registration, or for delegation of responsibility. NT uses its model of domain servers, rather like a NIS, but including a registration mechanism. A user in the SAM of a primary domain controller is registered within that domain and has an account on any host which subscribes to that domain.

An NT domain server involves not only shared databases but also shared administrative policies and shared security models. A host can subscribe to one or more domains and one domain can be associated with one another by a trust relationship. When one NT domain ‘trusts’ another, then accounts and groups defined in the *trusted* domain can be used in the *trusting* domain. NIS is indiscriminating in this respect. It is purely an authentication mechanism, implying no side-effects by the login procedure.

Other models of network computing include Kerberos and the Open Software Foundation’s Distributed Computing Environment (DCE). The former is a service brokering system with a common authentication service. The latter is a distributed user environment which can be used to provide a seamless world-wide distributed network domain. The DCE has been ported to both Unix and NT and requires a special login authentication after normal login to Unix/NT.

To summarize, rationalized user registration is a virtually unsupported problem in most operating systems. The needs of different organizations are varied and no successful solution to the problem has been devised and subsequently adopted as a standard. Networks are so common now that we have to think of the network first. Whether it happens today or tomorrow, at any given site, users will be moving around from host to host. They will need access to system resources wherever they are. It follows that they need distributed accounts. In creating a local solution, we have to bear in mind some basic constraints.

Principle (Distributed accounts). *Users move around from host to host, share data and collaborate. They need easy access to data and workstations all over an organization.*

Standardizing usernames across all platforms simplifies both the logistics of user management and opens the way for cross-platform compatibility. User names longer than eight characters can cause problems with Unix-like systems and FTP services. Users normally expect to be able to use the same password to log onto any host and have access to the same data, except for hosts with special purposes.

Suggestion (Passwords). *Give users a common username on all hosts, of no more than eight characters. Give them a common password on all hosts, unless there is a special reason not to do so. Some users never change their passwords unless forced to, and some users never even log* *in, so it is important to assign good passwords initially. Never assign a simple password and assume that it will be changed.*

Perl scripts are excellent ways of making user installation scripts which are tailored to local needs. For an excellent discussion of this on NT. Interactive programs are almost useless since users are seldom installed one by one. At universities hundreds of students are registered at the same time. No system administrator would type in all the names by hand. More likely they would be input from some administrative list generated by the admissions department.
The format of that list is not a universal standard, so no off-the-shelf software package is going to help here.

3.2.2 Unix accounts

To add a new user to a Unix-like host we have to:

- Find a unique username, user-id (uid) number and password for the new user.
- Update the system database of user accounts, e.g. add a line to the file /etc/passwd for Unix (or on the centralized password server of a network) for the new user.
- Create a login directory (home directory) for the user.
- Choose a shell for the user (if appropriate).
- Copy some configuration files like .cshrc or .profile into the new user's directory, or update the system registry.

Because every site is different, user registration requires different tools and techniques in almost every case. For example: where should users' home directories be located? GNU/Linux has an adduser script which assumes that the user will be installed on the local machine under /home/user, but many users belong to a network and their disk space lies physically on a different host which is mounted by NFS.

Unix developers have created three different password file formats which increase the awkwardness of distributing passwords. The traditional password file format is the following:

```
mark:Ax7Wc1Kd8ujo2:123:456:Mark Burgess:/home/mark:/bin/tcsh
```

The first field is a unique username (up to eight characters) for the user. The second is an encrypted form of the user's password; then comes the user-id (a unique number which represents the user and is equivalent to the username) and default group-id (a unique number which represents the default group of users to which this user belongs). The fifth field is the so-called GECOS field, which is usually just the full name of the user. On some systems, comma-separated entries may be given for full name, office, extension and home phone number. The sixth field is the home directory for the user (the root directory for the user's private virtual machine). Finally the seventh field is the user's default shell. This is the command interpreter which is started when the user logs in.

Newer Unix-like systems make use of *shadow password files*, which conceal the encrypted form of the password for ordinary users. The format of the password file is then the same as above, except that the second password field contains only an ‘x’, e.g.:

```
 mark:x:123:456:Mark  Burgess:/home/mark:/bin/tcsh
```

There is then a corresponding line in /etc/shadow with the form

```
mark:Ax7Wc1Kd8ujo2:6445::::::
```

or with an MD5 password hash, on some systems. The shadow file is not readable by ordinary users. It contains many blank fields which are reserved for the special purpose of password aging and other expiry mechanisms. See the manual page for ‘shadow’ for a description of the fields. The only number present by default is the time at which the password was last changed, measured in the number of days since 1. Jan. 1970.

The third form of password file is used by the BSD 4.4 derived operating systems.

```
mark:Ax7Wc1Kd8ujo2:3232:25::0:0:Mark Burgess:/home/mark:/bin/tcsh
```

It has extra fields, which are not normally used. These systems also have an optimization: in addition to the master password file base, they have a compiled binary database for rapid lookup. Administrators edit the file /etc/master.password and then run the pwd mkdb command to compile the database which is actually used for lookups. This generates text and binary versions of the password database.
Entries might have to be added to the group membership file /etc/group, E-mail system and quota database, depending on local requirements.

3.2.3 Windows accounts

Single Windows accounts are added with the command

```
 net user username  password  /ADD  /domain
```

or using the GUI. Windows does not provide any assistance for mass registration of users. The additional Resource Kit package contains tools which allow lists of users to be registered from a standard file format, with addusers.exe, but only at additional cost.

Windows users begin in the root directory by default. It is customary to create a \users directory for home directories. Network users conventionally have their home directory on the domain server mapped to the drive H:. There is only a single choice of shell (command interpreter) for NT, so this is not specified in the user registration procedure. Several possibilities exist for creating user profiles and access policies, depending on the management model used.

3.3 Account policy

Most organizations need a strict policy for assigning accounts and opening the system for users. Users *are* the foremost danger to a computing system, so the responsibility of owning an account should not be dealt out lightly. There are many ways in which accounts can be abused. Users can misuse accounts for villainous purposes and they can abuse the terms on which the account was issued, wasting resources on personal endeavors. For example, in Norway, where education is essentially free, students have been known to undergo semester registration simply to have an account, giving them essentially free access to the Internet and a place to host their web sites.

Policy rules are required for guiding user behavior, and also for making system rules clear.

Experience indicates that simple rules are always preferable, though this is so far unsubstantiated by any specific studies. A complex and highly specific rule, that is understood only by its author, may seem smart, but most users will immediately write it off as being nonsense. Such a rule is ill advised because it is opaque. The reason for the rule is not clear to all parties, and thus it is unlikely to be respected.

Principle (Simplest is best). *Simple rules make system behavior easy to understand. Users tolerate rules if they understand them and this tends to increase their behavioral predictability.*

What should an account policy contain?

1. Rules about what users are allowed/not allowed to do.
2. Specifications of what mandatory enforcement users can expect, e.g. tidying of garbage files.

Any account policy should contain a clause about weak passwords. If weak passwords are discovered, it must be understood by users that their account can be closed immediately. Users need to understand that this is a necessary security initiative. Closing Unix accounts can be achieved simply by changing their default shell in /etc/passwd to a script such as

```
#!/bin/sh

echo "/local/bin/blocked.passwd was run" | mail sysadm
 /usr/bin/last -10 | mail sysadm

 message=’
Your account has been closed because your password was found to be vulnerable to attack. To reopen your account, visit the admin office, carrying some form of personal identification.
 ’
echo "$message"

sleep 10
```

```
exit 0
```

Although this does not prevent them from doing simple things on an X-windows console running a login manager, like xdm, it does prevent them from logging in remotely, and it gets their attention. A more secure method is to simply replace their encrypted password with NP or *, which prevents them from being authenticated.

It is occasionally tempting to create guest accounts for visitors and transient users. NT has a ready-made guest account, which is not disabled by default on some versions of NT. Guest accounts are a bad idea, because they can be used long after a visitor has gone, they usually have weak or non-existent passwords and therefore are an open invitation to attack the system.

SELF ASSESSMENT EXERCISES

1. List the main issues in user management.
2. Where are passwords stored in Unix-like and Windows computers?
3. What is meant by a distributed account?
4. What special considerations are required for distributed accounts?

4.0 CONCLUSION

The role of the computer, as a tool, has changed extensively throughout history. From John von Neumann's vision of the computer as a device for predicting the weather, to a calculator for atomic weapons, to a desktop typewriter, to a means of global communication, computers have changed the world and have reinvented themselves in the process. System administrators must cater to all needs, and ensure the stability and security of the system.

5.0 SUMMARY

User management is about interfacing humans to computers. This brings to light a number of issues:

- Accounting: registering new users and deleting old ones.
- Comfort and convenience.
- Support services.
- Ethical issues.
- Trust management and security.

One of the first issues on a new host is to issue accounts for users. Surprisingly this is an area where operating system designers provide virtually no help. The tools provided by operating systems for this task are, at best, primitive and are rarely suitable for the task without considerable modification

Most organizations need a strict policy for assigning accounts and opening the system for users. Users *are* the foremost danger to a computing system, so the responsibility of owning an account should not be dealt out lightly. There are many ways in which accounts can be abused. Users can misuse accounts for villainous purposes and they can abuse the terms on which the account was issued, wasting resources on personal endeavors.

All users require help at some time or another. The fact that normal users are not *privileged users* means that they must occasionally rely on a super user to clean up a mess, or fix a problem which is beyond their control. If we are to distinguish between privileged and non-privileged users, we cannot deny users this service.

6.0 TUTOR-MARKED ASSIGNMENTS

1. What issues are associated with the installation of a new user account?
Discuss this with a group of classmates and try to turn your considerations into a policy checklist.

2. One of the central problems in account management is the distribution of passwords. If we are unable (or unwilling) to use a password distribution system like NIS, passwords have to be copied from host to host. Assume that user home-directories are shared amongst all hosts. Write a script which takes the password file on one host and converts it into all of the different file formats used by different Unix-like OSs, ready for distribution.

3. Describe the available support services for users at your site. Could these be improved? What would it cost to improve support services (can you estimate the number of man-hours, for instance) to achieve the level of support which you would like?

4. Discuss the following: Human beings are not moral creatures, we are creatures of habit. Thus law and policy enforcement is about making ethical choices habitual ones.

7.0 REFERNCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Burke, J. R.(2004). Network Management Concepts and Practice: a Hands-on Approach. Pearson.

3. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

4. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley.

5. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

6. Subramanian, M. (2000). Network Management: Principles and Practice, Addison-Wesley.

MODULE 3: NETWORK MANAGEMENT

UNIT 1: METHODS OF NETWORK ADMINISTRATION: SNMP & RMON

UNIT 2: CONFIGURATION AND MAINTENANCE

UNIT 3: DIAGNOSTICS, FAULT AND CHANGE MANAGEMENT

UNIT 4: MONITORING AND SYSTEM PERFORMANCE TUNING

UNIT 1: METHODS OF NETWORK ADMINISTRATION

1.0 INTRODUCTION

This unit describes functions common to most network-management architectures and protocols. It also presents the five conceptual areas of management as defined by the International Organization for Standardization (ISO). We then explore the Simple Network Management Protocol (SNMP) and Remote Network Monitoring {RMON} as methods of network administration.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- Describe Network Management Administration
- Describe the SNMP management model
- Explain how SNMP can be used to watch over and configure network devices
- State the limitations of SNMP
- Understand SNMP tools
- Explain what RMON is and its use

3.0 MAIN CONTENT

3.1 What Is Network Management?

Network management means different things to different people. In some cases, it involves a solitary network consultant monitoring network activity with an outdated protocol analyzer. In other cases, network management involves a distributed database, auto polling of network devices, and high-end workstations generating real-time graphical views of network topology changes and traffic. In general, network management is a service that employs a variety of tools, applications, and devices to assist human network managers in monitoring and maintaining networks.

A Historical Perspective

The early 1980s saw tremendous expansion in the area of network deployment. As companies realized the cost benefits and productivity gains created by network technology, they began to add networks and expand existing networks almost as rapidly as new network technologies and products were introduced. By the mid-1980s, certain companies were experiencing growing pains from deploying many different (and sometimes incompatible) network technologies.

The problems associated with network expansion affect both day-to-day network operation management and strategic network growth planning. Each new network technology requires its own set of experts. In the early 1980s, the staffing requirements alone for managing large, heterogeneous networks created a crisis for many organizations. An urgent need arose for automated network management (including what is typically called network capacity planning) integrated across diverse environments.

Network Management Architecture

Most network management architectures use the same basic structure and set of relationships. End stations (managed devices), such as computer systems and other network devices, run software that enables them to send alerts when they recognize problems (for example, when one or more user-determined thresholds are exceeded). Upon receiving these alerts, management entities are programmed to react by executing one, several, or a group of actions, including operator notification, event logging, system shutdown, and automatic attempts at system repair.

Management entities also can poll end stations to check the values of certain variables. Polling can be automatic or user-initiated, but agents in the managed devices respond to all polls. Agents are software modules that first compile information about the managed devices in which they reside, then store this information in a management database, and finally provide it (proactively or reactively) to management entities within network management systems (NMSs) via a network management protocol. Well-known network management protocols include the Simple Network Management Protocol (SNMP) and Common Management Information Protocol (CMIP). Management proxies are entities that provide management information on behalf of other entities. Figure 1 depicts a typical network management architecture.

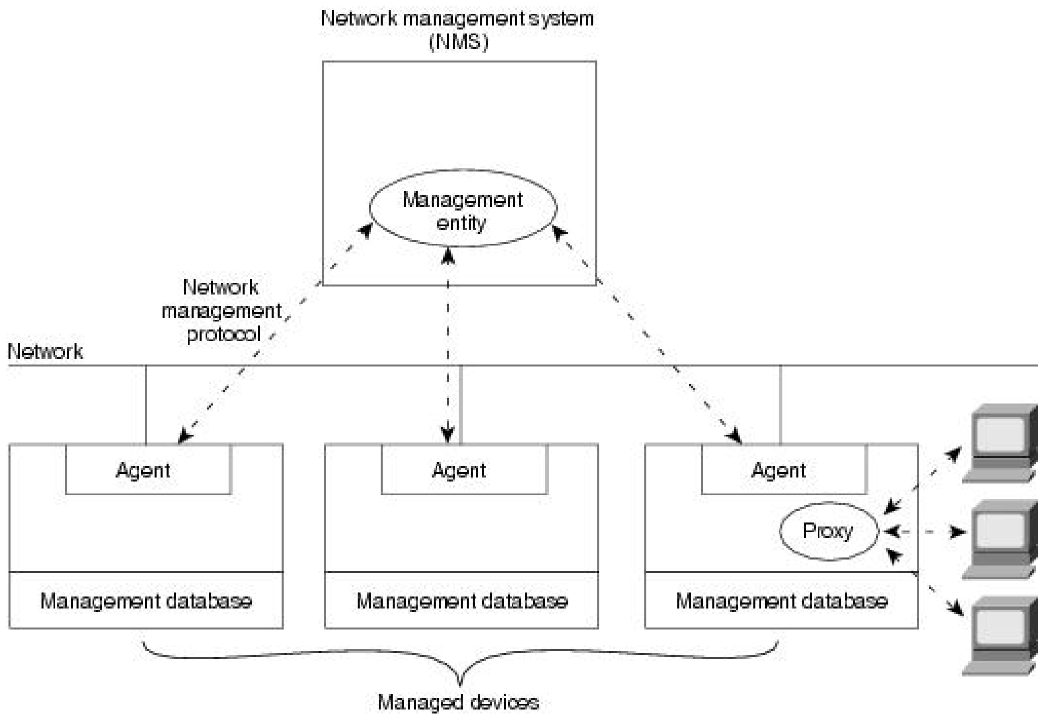


Figure 1 A Typical Network Management Architecture Maintains Many Relationships

ISO Network Management Model

The ISO has contributed a great deal to network standardization. Its network management model is the primary means for understanding the major functions of network management systems. This model consists of five conceptual areas, as now discussed.

Performance Management

The goal of *performance management* is to measure and make available various aspects of network performance so that internetwork performance can be maintained at an acceptable level. Examples of performance variables that might be provided include network throughput, user response times, and line utilization.

Performance management involves three main steps. First, performance data is gathered on variables of interest to network administrators. Second, the data is analyzed to determine normal (baseline) levels. Finally, appropriate performance thresholds are determined for each important variable so that exceeding these thresholds indicates a network problem worthy of attention.

Management entities continually monitor performance variables. When a performance threshold is exceeded, an alert is generated and sent to the network management system.

Each of the steps just described is part of the process to set up a reactive system. When performance becomes unacceptable because of an exceeded user-defined threshold, the system reacts by sending a message. Performance management also permits proactive methods: For example, network simulation can be used to project how network growth will affect performance metrics. Such simulation can alert administrators to impending problems so that counteractive measures can be taken.

Configuration Management

The goal of *configuration management* is to monitor network and system configuration information so that the effects on network operation of various versions of hardware and software elements can be tracked and managed.

Each network device has a variety of version information associated with it. An engineering workstation, for example, may be configured as follows:

- Operating system, Version 3.2
- Ethernet interface, Version 5.4
- TCP/IP software, Version 2.0
- NetWare software, Version 4.1
- NFS software, Version 5.1
- Serial communications controller, Version 1.1
- X.25 software, Version 1.0
- SNMP software, Version 3.1

Configuration management subsystems store this information in a database for easy access. When a problem occurs, this database can be searched for clues that may help solve the problem.

Accounting Management

The goal of *accounting management* is to measure network utilization parameters so that individual or group uses on the network can be regulated appropriately. Such regulation minimizes network problems (because network resources can be apportioned based on resource capacities) and maximizes the fairness of network access across all users.

As with performance management, the first step toward appropriate accounting management is to measure utilization of all important network resources. Analysis of the results provides insight into current usage patterns, and usage quotas can be set at this point. Some correction, of course, will be required to reach optimal access practices. From this point, ongoing measurement of resource use can yield billing information as well as information used to assess continued fair and optimal resource utilization.

Fault Management

The goal of *fault management* is to detect, log, notify users of, and (to the extent possible) automatically fix network problems to keep the network running effectively. Because faults can cause downtime or unacceptable network degradation, fault management is perhaps the most widely implemented of the ISO network management elements.

Fault management involves first determining symptoms and isolating the problem. Then the problem is fixed and the solution is tested on all-important subsystems. Finally, the detection and resolution of the problem is recorded.

Security Management

The goal of *security management* is to control access to network resources according to local guidelines so that the network cannot be sabotaged (intentionally or unintentionally) and sensitive information cannot be accessed by those without appropriate authorization. A security management subsystem, for example, can monitor users logging on to a network resource and can refuse access to those who enter inappropriate access codes.

Security management subsystems work by partitioning network resources into authorized and unauthorized areas. For some users, access to any network resource is inappropriate, mostly because such users are usually company outsiders. For other (internal) network users, access to information originating from a particular department is inappropriate. Access to Human Resource files, for example, is inappropriate for most users outside the Human Resources department.

Security management subsystems perform several functions. They identify sensitive network resources (including systems, files, and other entities) and determine mappings between sensitive network resources and user sets. They also monitor access points to sensitive network resources and log inappropriate access to sensitive network resources.

3.2 SNMP network management

The ability to read information about the performance of network hardware via the network itself is an attractive idea. Suppose we could look at a router on the second floor of a building half a mile away and immediately see the load statistics, or the number of rejected packets it has seen; or perhaps the status of all printers on a subnet. That would be useful diagnostic information. Similar information could be obtained about software systems on any host.

The Simple Network Management Protocol (SNMP) is a protocol designed to do just this.

SNMP was spawned in 1987 as a Simple Gateway Monitoring Protocol, but was quickly extended and became a standard for network monitoring. SNMP was designed to be small and simple enough to be able to run on even minor pieces of network technology like bridges and printers. The model of configuration management is particularly suited to non-interactive devices like printers and static network infrastructure that require an essentially static configuration over long periods of time. SNMP has since been extended, with less success, to some aspects of host management such as workstations and PC network server configuration; however, the static model of configuration is less appropriate here since users are constantly perturbing servers in unpredictable ways and, combined with the unimpressive security record of early versions, this has discouraged its use for host management.

SNMP now exists in three versions. The traditional SNMP architecture is based on two entities: managers and agents. SNMP managers execute management applications, while SNMP agents mediate access to management variables. These variables hold simple typed values and are arranged into groups of scalars of single-valued variables or into conceptual tables of multi-valued variables. The set of all variables on a managed system is called the Management Information Base (MIB).

SNMP has often been criticized for the weak security of its agents, which are configured by default with a clear text password of ‘public’. Version 3 of the SNMP protocol was finally agreed on and published in December 2002 in order to address these problems, using strong encryption methods. If or when this version becomes widespread, SNMP will be as secure as any other network service.

SNMP supports three operations on devices: *read*, *write* and *notify* (through ‘traps’). The management console can read and modify the variables stored on a device and issue notifications of special events. SNMP access is mediated by a server process on each hardware node (the agent), which normally communicates by UDP/IP on ports 161 and 162. Modern operating systems often run SNMP daemons or services which advertise their status to an SNMP-capable manager. The services are protected by a rather weak password which is called the *community string*.

Because SNMP is basically a ‘peek–poke’ protocol for simple values, its success depends crucially on the ability of the Management Information Bases or MIBs to correctly characterize the state of devices, and how the agents translate MIB values into real actions. For monitoring workload (e.g. load statistics on network interfaces or out-of-paper signals on a printer), this model is clearly quite good. Indeed, even host-based tools (ps, top, netstat etc.) use this approach for querying resource tables on more complex systems. Moreover, in the case of dumb network devices, whose behavior is essentially fixed by a few parameters or lists of them (printers, switches etc.), this model even meets the challenge of configuration reasonably well. The notify functionality is implemented by 'traps' or events that can be configured in the SNMP agent. Each event type is defined by the SNMP software of the device being managed, e.g. Cisco routers have general traps, such as:

coldStart
linkDown
linkUp
authenticationFailure
egpNeighborLoss
reload
tcpConnectionClose
ciscoConfigManEvent

that are triggered by interface changes and management events.

The variables that exist in an MIB are formally defined in so-called MIB modules that are written in a language called the Structure of Management Information (SMI). The SMI provides a generic and extensible name-space for identifying MIB variables. Due to a lack of higher level data structuring facilities, MIB modules often appear as a patchwork of individual variable definitions rather than a class structure in Java or other object-oriented languages. An SNMP request specifies the information it wants to read/write by giving the name of an instance of the variable to read or write in a request. This name is assigned in the formal MIB module. There are standard MIB modules for address translation tables, TCP/IP statistics and so on. These have default parameters that can be altered by SNMP: system name, location and human contact; interface state (up/down), hardware and IP address, IP state (forwarding gateway/not) IP TTL, IP next HOP address, IP route age and mask, TCP state, neighbor state, SNMP trap enabling, and so on.

Although SNMP works fairly well for monitoring clusters of static devices, its success in managing hosts, including workstations and servers, is more questionable. The MIB model is very difficult to tailor to hosts where users are interacting with the system constantly. Hosts are no longer basically predictable devices with approximately constant state; their state changes dynamically in response to user interaction, and in response to the services they perform. The SNMP management philosophy, i.e. of communicating with an agent for reading and writing, is only effective when the rate of change of state of a device is slow, i.e. when the device changes only at about the same rate as the rate of change of policy itself. Moreover, the complexity of operations that can be carried out by a dumb SNMP agent, based only on MIB 'push button' instructions and a bilateral communication with a management interface, is limited. Thus, in practice, SNMP can only be used to detect problems, not to repair them. This is not so much a problem in the model, but in the manner in which it is implemented. If SNMP agents contained on-board intelligence, then the communication model could be retained. One is then into the territory of more intelligent agent systems like cfengine and PIKT.

Despite the shortcomings of SNMP for host operations, many operating systems do define their own MIBs for the collection of system performance data and even for essential configuration parameters. Some commercial network management systems like Hewlett Packard's OpenView work by reading and writing MIBs using SNMP client-server technology. Most Unix variants, Novell and NT now also support SNMP. Their MIBs can be used even to collect information such as the names of users who are logged on. This information is not particularly relevant to the problem of resource management and can even be considered a security risk, thus many remain sceptical about the use of SNMP on hosts.

In 1997, SNMPv3 was put forward in order to provide stronger security, particularly in the authentication of the manager–agent connection. This has helped to allay some of the fears in using SNMP where it is appropriate, but it does not make the task of tailoring the MIB model easier.

SNMP seems to be increasing in popularity for monitoring network hardware (routers and switches etc.), but like any public information database, it can also be abused by network attackers. SNMP is a prime target for abuse and some sites choose to disable SNMP services altogether on hosts, using it only for monitoring network transport hardware.

Suggestion (SNMP containment). *Sites should filter SNMP packets to and from external networks to avoid illegal access of these services by intruders.*

3.3 SNMP tools

In spite of its limitations, SNMP remains the protocol of choice for the management of most network hardware, and many tools have been written to query and manage SNMP enabled devices.

The fact that SNMP is a simple read/write protocol has motivated programmers to design simple tools that focus more on the SNMP protocol itself than on the semantics of the data structures described in MIBs. In other words, existing tools try to be generic instead of doing something specific and useful. Typical examples are so-called MIB browsers that help users to browse and manipulate raw MIB data. Such tools usually only understand the machine-parseable parts of a MIB module – which is just adequate to shield users from the bulk of the often arcane numbers used in the protocol. Other examples are scripting language APIs which provide a 'programmer-friendly' view on the SNMP protocol. However, in order to realize more useful management application, it is necessary to understand the semantics of and the relationships between MIB variables. Generic tools require that the users have this knowledge – which is however not always the case.

PHP

The PHP server-side web page language (an enhanced encapsulation of C) is perhaps the simplest way of extracting MIB data from devices, but it is just a generic, low-level interface.

PHP makes use of the Net SNMP libraries. For example, here is a simple PHP web page that prints all of the SNMP variables for a device and allows the data to be viewed in a web browser:

```
 <?php

 $a = snmpwalk("printer.example.org", "public", "");
 For  ($i=0;  $i  <  count($a);  $i++)
  {
     Echo  "$a[$i] <br>";
  }

?>
```

The community string is written here with its default values ‘public’, but it is assumed that this has been changed to something more private. PHP is well and freely documented online, in contrast with Perl. For monitoring small numbers of devices, and for demonstrating the principles of SNMP, this is an excellent tool. However, for production work, something more sophisticated will be required by most users.

Perl, Tcl etc.

There are several SNMP extensions for Perl; a widely used Perl SNMP API is based on the NET-SNMP implementation and supports SNMPv1, SNMPv2c and SNMPv3.

The problem with Perl is that it only puts a brave face on the same problems that PHP has: namely, it provides only a low-level interface to the basic read/write operations of the protocol. There is no intelligence to the interface, and it requires a considerable amount of programming to do real management with this interface.

Another SNMP interface worthy of mention is the Tcl extension, Scotty.

SCLI

One of the most effective ways of interacting with any system is through a command language. With language tools a user can express his or her exact wishes, rather than filtering them through a graphical menu.

The scli package was written to address the need for rational command line utilities for monitoring and configuring network devices. It utilizes a MIB compiler called smidump to generate C stub code. It is easily extensible with a minimum of knowledge about SNMP.

The programs contained in the scli package are specific rather than generic. Generic SNMP tools such as MIB browsers or simple command line tools (e.g. snmpwalk) are hard to use since they expose too many protocol details for most users. Moreover, in most cases, they fail to present the information in a format that is easy to read and understand. A nice feature of scli is that it works like other familiar Unix commands, such as netstat and top, and generates a feeling of true investigative interaction.

3.4 TCP/IP Remote Network Monitoring (RMON)

The Simple Network Management Protocol (SNMP) defines both a framework and a specific protocol for exchanging network information on a TCP/IP internetwork. The general model used by SNMP is that of a network management station (NMS) that sends requests to SNMP agents running on managed devices. The SNMP agents may also initiate certain types of communication by sending *trap* messages to tell the NMS when particular events occur.

This model works well, which is why SNMP has become so popular. However, one fundamental limitation of the protocol and the model it uses is that it is oriented around the communication of network information from SNMP agents that are normally part of regular TCP/IP devices, such as hosts and routers. The amount of information gathered by these devices is usually somewhat limited, because obviously hosts and routers have “real work to do”—that is, doing the jobs of being hosts and routers. They can't devote themselves to network management tasks.

Thus, in situations where more information is needed about a network than is gathered by traditional devices, administrators often use special hardware units called *network analyzers*, *monitors* or *probes*. These are dedicated pieces of equipment that are connected to a network and used strictly for the purpose of gathering statistics and watching for events of interest or concern to the administrator. It would obviously be very useful if these devices could use SNMP to allow the information they gather to be retrieved, and to let them generate traps when they notice something important.

To enable this, the *Remote Network Monitoring (RMON)* specification was created. RMON is often called a protocol, and you will sometimes see SNMP and RMON referred to as “the TCP/IP network management protocols”. However, RMON really isn't a separate protocol at all—it defines no protocol operations. RMON is in fact part of SNMP, and the RMON specification is simply a management information base (MIB) module that defines a particular set of MIB objects for use by network monitoring probes. Architecturally, it is just one of the many MIB modules that comprise the SNMP Framework.

Key Concept: SNMP *Remote Network Monitoring (RMON)* was created to enable the efficient management of networks using dedicated management devices such as network analyzers, monitors or probes. RMON is often called a “protocol”, but does not in fact define any new protocol operations; it is a MIB module for SNMP that describes objects that permit advanced network management capabilities.

RMON Standards

The first standard documenting RMON was RFC 1271, *Remote Network Monitoring Management Information Base*, published in 1991. RFC 1271 was replaced by RFC 1757 in 1995, which made a couple of changes to the specification. RFC 2819, May 2000, updates RMON to use the new Structure of Management Information version 2 (SMIv2) specification that is part of SNMPv2 but is functionally the same as RFC 1757.

SELF ASSESSMENT EXERCISES

1a *Name the different areas of network management.*

b *What are the goals of performance management?*

c *What are the goals of configuration management?*

d *What are the goals of security management?*

.2. What is an MIB?

4.0 C0NCLUSION

The unit has provided you with fundamentals and theoretical foundations of Network Administration ell well as practical skills needed to manage networks.

5.0SUMMARY

Network management is a service that employs a variety of tools, applications, and devices to assist human network managers in monitoring and maintaining networks.

The Simple Network Management Protocol (SNMP) defines both a framework and a specific protocol for exchanging network information on a TCP/IP internetwork. This model works well, which is why SNMP has become so popular. However, one fundamental limitation of the protocol and the model it uses is that it is oriented around the communication of network information from SNMP agents that are normally part of regular TCP/IP devices, such as hosts and routers. The amount of information gathered by these devices is usually somewhat limited, because obviously hosts and routers have “real work to do”—that is, doing the jobs of being hosts and routers. They can't devote themselves to network management tasks.

Thus, in situations where more information is needed about a network than is gathered by traditional devices, administrators often use special hardware units called *network analyzers*, *monitors* or *probes*. These are dedicated pieces of equipment that are connected to a network and used strictly for the purpose of gathering statistics and watching for events of interest or concern to the administrator. It would obviously be very useful if these devices could use SNMP to allow the information they gather to be retrieved, and to let them generate traps when they notice something important.

To enable this, the *Remote Network Monitoring (RMON)* specification was created. RMON is often called a protocol, and you will sometimes see SNMP and RMON referred to as “the TCP/IP network management protocols”. However, RMON really isn't a separate protocol at all—it defines no protocol operations. RMON is in fact part of SNMP, and the RMON specification is simply a management information base (MIB) module that defines a particular set of MIB objects for use by network monitoring probes. Architecturally, it is just one of the many MIB modules that comprise the SNMP Framework.

6.0 TUTOR-MARKED ASSIGNMENTS

1. Describe the SNMP network management
2. Explain how SNMP can be used to watch over and configure network devices.

   What are the limitations of SNMP?

3. Describe the Network Management Architecture
4. Discuss the TCP/IP Remote Network Monitoring (RMON)

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Burke, J. R.(2004). Network Management Concepts and Practice: a Hands-on Approach. Pearson.

3. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

4. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley.

5. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

6. Subramanian, M. (2000). Network Management: Principles and Practice, Addison-Wesley.

UNIT 2: CONFIGURATION AND MAINTENANCE

1.0 INTRODUCTION

We are now faced with two overlapping issues: how to make a computer system operate in the way we have intended, and how to keep it in that state over a period of time. Configuration management which we shall discuss is the administration of *state* in hosts or network hardware.

2.0 OBJECTIVES

At the end of the unit, you should be able to:

- Understand what is meant by configuring management in the context of network administration
- Explain the role of policy in determining device configuration.
- Explain what is meant by change management
- Know about script languages for customization of the system
- Know the need for automation of configuration.

3.0 MAIN CONTENT

3.1 System configuration policy

Another side of network setup is the policies, practices and procedures which are used to make changes to or to maintain the system as a whole, i.e. what humans decide as part of the system administration process.

System administration is often a collaborative effort between several administrators. It is therefore important to have agreed policies for working so that everyone knows how to respond to 'situations' which can arise, without working against one another. A system policy also has the role of summarizing the attitudes of an organization to its members and its surroundings and often embodies security issues. As Howell cites from Pogo, '*We have met the enemy, and he us!*' A system policy should contain the issues we have been discussing in the foregoing chapters. There are issues to be addressed at each level: network level, host level, user level.

It is crucial that everyone agrees on policy matters. Although a policy can easily be an example of blind rule-making, it is also a form of communication. A policy documents acceptable behavior, but it should also document what response is appropriate in a crisis. Only then are we assured of an orchestrated response to a problem, free of conflicts and disagreements. What is important is that the document does not simply become an exercise in bureaucracy, but is a living guide to the *practice* of network community administration. A system policy can include some or all of the following:

• *Organization:* What responsibility will the organization take for its users' actions? What responsibility will the organization take for the users' safety.
Who is responsible for what? Has the organization upheld its responsibilities to the wider network community? Measures to prevent damage to others and from others.
• *Users:* Allowing and forbidding certain types of software. Rigid control over space (quotas) or allow freedom, but police the system with controls. Choice of default configuration. A response to software piracy. A response to anti-social behavior and harassment of others (spamming, obnoxious news postings etc.). Are users allowed to play games, if so when? Are users allowed to chat online? Are users allowed to download files such as MP3 or pornography?
Policy on sharing of accounts (i.e. preferably not). Policy on use of IRC robots and other automatic processes which collect large amounts of data off-line. Policy on garbage collection when disks become full: what files can legitimately be deleted?
• *Network:* Will the network be segmented, with different access policies on different subnets? Will a firewall be used? What ports will be open on which subnets, and which will be blocked at the router? What services will be run?
• *Mail:* Limit the size of incoming and outgoing mail. Spam filtering. Virus controls.
• *WWW:* Allowing or forbidding user CGI scripts. Guidelines for allowed content of web pages. Policy regarding advertising on web pages. Load restrictions: what to do if certain pages generate too much traffic. Policy on plagiarism and illegal use of imagery.
• *Printing:* How many pages can be printed? Is printing of personal documents allowed? Should there be a limit to the number of pages which can be printed at one time (large documents hold up the print queue)?
• *Security:* Physical security of hosts. Backup schedule. Who is allowed to be master of their own hosts? Can arbitrary users mount other users' home directories or mailboxes with NFS on their private PCs (this means that they have automatic access to everyone's personal files)? What access controls should be used on files? Password policy (aging, how often should passwords change) and policy on closing accounts which have been compromised.

3.2 Methods: controlling causes and symptoms

Component-based software development is a central theme in modern design, and it has almost exclusively followed the path of trading control over algorithmic detail for limited freedoms through configurable parameters to standardized 'methods'. Sun Microsystems' Java technology and Microsoft's .NET are two recent developments that exemplify this trend. The need for *configuration* is thus a feature of all modern software systems, and what was previously an issue of programming a sequence of imperative logic, is now an issue of administrating a few basic choices. Thus programming is increasingly turning into system administration.

As we move ever more towards standardized methods and algorithms, the process of programming becomes increasingly one of administering the few remaining choices, as configuration options.

Software engineers seldom think of the process of configuring components as a system administration issue, because ‘system administration’ is commonly assumed to apply only to the low-level infrastructure such as hardware and software installation. Nevertheless, it is important to understand the equivalence of these issues, because many flaws in software systems are provoked and exploited because software engineers make naive assumptions about infrastructure. Similarly, software engineers seldom think carefully about how software will be configured in practice across large installation bases. System administrators are thus left to improvise the best from a bad lot.

Clearly, there is a trade-off between detailed control and increasing standardization.

This trade-off is always a dilemma in the policy-making and government of both man and machine. There are ethical issues here also. By adopting standardized methods, one removes freedom of choice from the end user, or programmer; it is an authoritarian strategy that some users find disturbing, because it assumes that a high-level, standard authority knows better than a low-level technician, which is seldom true in a knowledge-based society; but it also has clear benefits of simplification to offer. There is economy in standardization, and – correctly implemented – a standard can find great leverage in the experiences of experts.
Great power requires great responsibility, however, and should not be wielded in an inflexible way.

In this case, standardization to an unsatisfactory state leads to strategies for relieving symptoms rather than curing them at source. Although it is preferable to fix problems at source, it is not always possible to do so. There will always be a need to distinguish between short-term patches and long-term patches, since the rate of software correction is much less than the rate at which errors are discovered.

3.3 Change management

The opposite side of the coin in configuration and maintenance is the management of significant changes, e.g. upgrades, redesign and replacement. Can such things be done without disruption to service? Does this idea contradict the idea of convergence? Planning changes of infrastructure can be dealt with using two general strategies:

- Deconstruction followed by reconstruction.
- Change of policy description followed by convergence to a new state.

We might call these ‘change’ and ‘organic growth’ respectively. Traugott and Huddleston introduced the idea of infrastructure management to describe the construction of systems from the bottom up. Traugott has later argued that this infrastructure needs to be maintained in much the same way as building it in the first place. Change management then becomes a reconstruction of infrastructure. An ideologically ‘convergent’ approach would be to try to gradually change aspects of policy and allow the system to converge towards the state associated with the change.

To date, no study has been performed to compare these two approaches for major changes. Clearly, the larger the magnitude of a change, the closer these two approaches must become. The amount of work required to perform large changes through differential adjustment grows significantly with the magnitude of the change. At some point, the benefit of adjustment rather than reconstruction becomes ambiguous. Many system administrators will doubtless feel more comfortable with starting from scratch when large changes need to be made as a matter of convenience.

Why would people go over to the new, if the old still works? When making changes, one must not forget the issue of service provision and reliability. Temporary redundancy of service is a sensible precaution in a mission-critical environment. If something should go wrong during a change, service must continue. Securing predictability during a change is a tricky business, because the conditions under which a system is performing its function are changing. Change management can thus be viewed as a problem in risk or fault management.

3.4 Automation of host configuration

The need for automation has become progressively clearer as sites grow and the complexity of administration increases. Some advocates have gone in for a distributed object model. Others have criticized a reliance on network services.

3.4.1 Tools for automation

Most system administration tools developed and sold today (insofar as they exist) are based either on the idea of *control interfaces* (interaction between administrator and machine to make manual changes) or on the cloning of existing reference systems (mirroring}. One sees graphical user interfaces of increasing complexity, but seldom any serious attention to autonomous behavior.

Many ideas for automating system administration have been reported; Most of these have been ways of generating or distributing simple shell or Perl scripts. Some provide ways of cloning machines by distributing files and binaries from a central repository. In spite of the creative effort spent developing the above systems, few if any of them can survive in their present form in the future. As indicated by Evard, analyzing many case studies, what is needed is a greater level of abstraction.

Vendors have also built many system administration products. Their main focus in commercial system administration solutions has been the development of man–machine interfaces for system management. They are mainly control-based systems which give responsibility to humans, but some can be used to implement partial immunity type schemes by instructing hosts to execute automatic scripts. However, they are not comparable to cfengine in their treatment of automation, they are essentially management frameworks which can be used to activate scripts.

Tivoli is probably the most advanced and wide-ranging product available. It is a Local Area Network (LAN) management tool based on CORBA and X/Open standards; it is a commercial product, advertised as a complete management system to aid in both the logistics of network management and an array of configuration issues. As with most commercial system administration tools, it addresses the problems of system administration from the viewpoint of the business community, rather than the engineering or scientific community. Tivoli admits bidirectional communication between the various elements of a management system. In other words, feedback methods could be developed using this system. The apparent drawback of the system is its focus on application-level software rather than core system integrity. Also it lacks abstraction methods for coping with real-world variation in system setup.

Tivoli's strength is in its comprehensive approach to management. It relies on encrypted communications and client-server interrelationships to provide functionality including software distribution and script execution. Tivoli can activate scripts but the scripts themselves are a weak link. No special tools are provided here; the programs are essentially shell scripts with all of the usual problems.
Client-server reliance could also be a problem: what happens if network communications are prevented?

Tivoli provides a variety of ways for activating scripts:

- Execute by hand when required.
- Schedule tasks with a cron-like feature.
- Execute an action (run a task on a set of hosts, copy a package out) in response to an event.

Tivoli's Enterprise Console includes a language Prolog for attaching actions to events. Tivoli is clearly impressive but also complex. This might also be a weakness. It requires a considerable infrastructure in order to operate, an infrastructure which is vulnerable to attack.

HP OpenView is a commercial product based on SNMP network control protocols. Openview aims to provide a common configuration management system for printers, network devices, Windows and HPUX systems. From a central location, configuration data may be sent over the local area network using the SNMP protocol. The advantage of Openview is a consistent approach to the management of network services; its principal disadvantage, in the opinion of the author, is that the use of network communication opens the system to possible attack from hacker activity. Moreover, the communication is only used to alert a central administrator about perceived problems. Little automatic repair can be performed and thus the human administrator is simply overworked by the system.

Sun's Solstice system is a series of shell scripts with a graphical user interface which assists the administrator of a centralized LAN, consisting of Solaris machines, to initially configure the sharing of printers, disks and other network resources. The system is basically old in concept, but it is moving towards the ideas in HP Openview.

Host Factory is a third party software system, using a database combined with a revision control system which keeps master versions of files for the purpose of distribution across a LAN. Host Factory attempts to keep track of changes in individual systems using a method of revision control. A typical Unix system might consist of thousands of files comprising software and data. All of the files (except for user data) are registered in a database and given a version number. If a host deviates from its registered version, then replacement files can be copied from the database. This behavior hints at the idea of an immune system, but the heavy-handed replacement of files with preconditioned images lacks the subtlety required to be flexible and effective in real networks. The blanket copying of files from a master source can often be a dangerous procedure. Host Factory could conceivably be combined with cfengine in order to simplify a number of the practical tasks associated with system configuration and introduce more subtlety into the way changes are made. Currently Host Factory uses shell and Perl scripts to customize master files where they cannot be used as direct images. Although this limited amount of customization is possible, Host Factory remains essentially an elaborate cloning system.

3.4.2 Monitoring tools

Monitoring tools have been in proliferation for several years They usually work by having a daemon collect some basic auditing information, setting a limit on a given parameter and raising an alarm if the value exceeds acceptable parameters. Alarms might be sent by mail, they might be routed to a GUI display or they may even be routed to a system administrator's pager. Network monitoring advocates have done a substantial amount of work in perfecting techniques for the capture and decoding of network protocols. Programs such as etherfind, snoop, tcpdump and bro, as well as commercial solutions such as Network Flight Recorder, place computers in 'promiscuous mode', allowing them to follow the passing data-stream closely. The thrust of the effort here has been in designing systems for collecting data, rather than analyzing them extensively. The monitoring school advocates storing the huge amounts of data on removable media such as CD, to be examined by humans at a later date if attacks should be uncovered. The analysis of data is not a task for humans, however. The level of detail is more than any human can digest and the rate of its production and the attention span and continuity required are inhuman. Rather we should be looking at ways in which machine analysis and pattern detection could be employed to perform this analysis – and not merely after the fact. In the future, adaptive neural nets and semantic detection will likely be used to analyze these logs in real time, avoiding the need to even store the data in raw form.

Unfortunately there is currently no way of capturing the details of every action performed by the local host, analogous to promiscuous network monitoring, without drowning the host in excessive auditing. The best one can do currently is to watch system logs for conspicuous error messages.

Visualization is now being recognized as an important tool in understanding the behavior of network systems. This reinforces the importance of investing in a documentable understanding of host behavior, rather than merely relating experiences and beliefs.

3.4.3 A generalized scripting language

Customization of the system requires us to write programs to perform special tasks. Perl was the first of a group of scripting languages including python, tcl and scheme, to gain acceptance in the Unix world. It has since been ported to Windows operating systems also. Perl programming has, to some extent, replaced much shell programming as the Free Software lingua franca of system administration. More recently Python, PHP and Tcl have been advocated also.

The Perl language is a curious hybrid of C, Bourne shell and C-shell, together with a number of extra features which make it ideal for dealing with text files and databases. Since most system administration tasks deal with these issues, this places Perl squarely in the role of system programming. Perl is semi-compiled at runtime, rather than interpreted line-by-line like the shell, so it gains some of the advantages of compiled languages, such as syntax check before execution and so on. This makes it a safer and more robust language. It is also portable (something which shell scripts are not). Although introduced as a scripting language, like all languages, Perl has been used for all manner of things for which it was never intended. Scripting languages have arrived on the computing scene with an alacrity which makes them a favorable choice to anyone wanting to get code running quickly. This is naturally a mixed blessing. What makes Perl a winner over many other special languages is that it is simply too convenient to ignore for a wide range of frequently required tasks. By adopting the programming idioms of well-known languages, as well as all the basic functions in the C library, Perl ingratiates itself to system administrators and becomes an essential tool.

SELT ASSESSMENT EXERCISES

1. Explain why system configurations tend to fall into a state of disorder over time.
2. What is meant by change management?

4.0 CONCLUSION

Configuration and maintenance are clearly related issues. Maintenance is simply configuration in the face of creeping decay. All systems tend to decay into chaos with time. There are many reasons for this decline, from deep theoretical reasons about thermodynamics, to the more intuitive notions above wear and tear. To put it briefly, it is clear that the number of ways in which a system can be in order is far fewer than the number of ways in which a system can be in a state of disorder, thus statistically any random change in the system will move it into disorder, rather than the other way around.

5.0 SUMMARY

Configuration management is the administration of *state* in hosts or network hardware An other side of network setup is the policies, practices and procedures which are used to make changes to or to maintain the system as a whole, i.e. what humans decide as part of the system administration process.

System administration is often a collaborative effort between several administrators. It is therefore important to have agreed policies for working so that everyone knows how to respond to ‘situations’ which can arise, without working against one another. A system policy also has the role of summarizing the attitudes of an organization to its members and its surroundings and often embodies security issues.

The opposite side of the coin in configuration and maintenance is the management of significant changes, e.g. upgrades, redesign and replacement.

The need for automation has become progressively clearer as sites grow and the complexity of administration increases. Many ideas for automating system administration have been reported; Most of these have been ways of generating or distributing simple shell or Perl scripts. Some provide ways of cloning machines by distributing files and binaries from a central repository. In spite of the creative effort spent developing the above systems, few if any of them can survive in their present form in the future.

Monitoring tools have been in proliferation for several years. They usually work by having a daemon collect some basic auditing information, setting a limit on a given parameter and raising an alarm if the value exceeds acceptable parameters.

Customization of the system requires us to write programs to perform special tasks. Perl was the first of a group of scripting languages including python, tcl and scheme, to gain acceptance in the Unix world. It has since been ported to Windows operating systems also.

6.0 TUTOR-MARKED ASSIGNMENTS

1. What is meant by configuration management in the context of network and system administration?

2. How is configuration information stored by devices?

3. Summarize the alternatives available for automating host management. What limitations does each of the alternatives have?

4. Discuss System configuration policy

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

3. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley

4. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

UNIT 3: DIAGNOSTICS, FAULT AND CHANGE MANAGEMENT

1.0 INTRODUCTION

All complex systems behave unexpectedly some of the time. They fail to operate within the limits addressed by policy and the reason for this can be clearly understood by comparing information content. Policy is generally a set of simplistic, high level general rules that cannot capture the same level of detail as the true human–computer interaction in its real environment. One must therefore expect failure and plan for it. If a failure occurs in a manner that was expected, its effects can be controlled and mitigated.

This unit is about learning what to expect of a non-deterministic system: how to understand its flaws, and how to insure oneself against the unexpected.

2.0 OBJECTIVES

At the end of the unit, you should be able to:

- Know what are faults
- Explain the strategies for finding faults and correcting them
- Know why we should repair errors.
- Establish cause and effect of faults

3.0 MAIN CONTENT

3.1 Fault tolerance and propagation

How do errors penetrate a system? Faults travel from part to part as if in a network of interconnections. If errors can propagate freely, then a small error in one part of a system can have consequences for another part. By studying different kinds of network, we can learn about the likelihood of error propagation.

Networks come in a variety of forms. Figure 1 shows the progression from a highly ordered, centralized structure to a decentralized form, to a generalized mesh. This classification was originally discussed by Paul Baran of RAND Corporation in 1964 as part of a project to develop a communications system that would be robust to failure in the case of a nuclear attack.

The same argument applies to the propagation of errors though any set of interconnected parts.

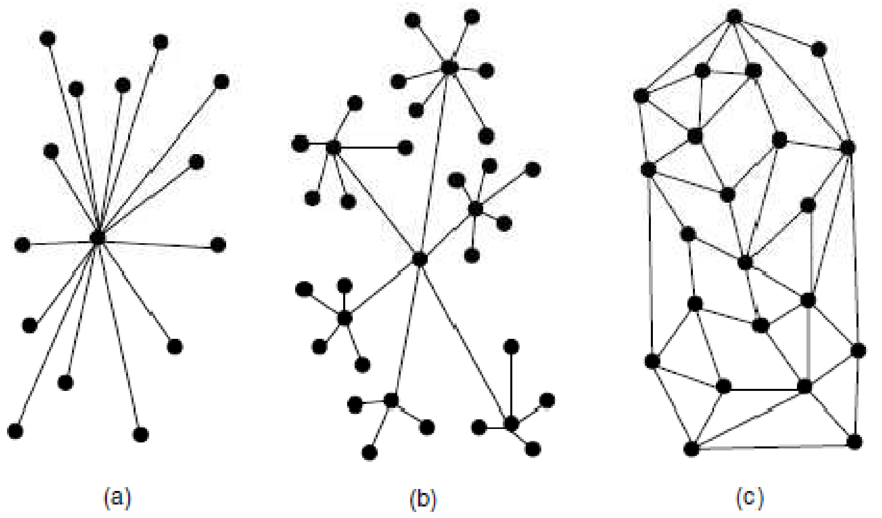


Figure 1: Network topologies: (a) centralized, (b) decentralized or hierarchical, and (c) distributed mesh.

Many complex systems exhibit a surprising degree of tolerance against errors. This is because they have in-built redundancy. Certain types of network also have this property in the routes between the nodes. If we think of networks not so much in the sense of communication lines between computers, but as abstract links between different dependent parts of the whole, then the importance of networks becomes apparent. The idea of a network is thus of more general importance than as a means of communication between computers and humans. Networks are *webs of influence*. If a system is tolerant to faults and security breaches, then we can look at it in one of two complementary ways:

- The access network that allows problems to propagate is poorly connected; i.e. connections (security breaches) between nodes (resources) are absent.
- The resource network is well connected and is resilient to removal of nodes (resources) and connections (supply channels).

The first of these viewpoints is useful for modeling intrusion or penetration by faults or intruders, while the latter is useful for securing a system against lack of access to critical resources.

A *tolerant network* is robust to node removal and connection removal. Node removal is usually more serious (see figure 2).

One type of network of special importance is the *random network*. A random network is formed by making random connections between nodes within a set. Randomness is a good strategy for covering a large number of possibilities without making exhaustive use of resources. In the absence of precise knowledge about a system, random 'shots in the dark' are an efficient way of hitting an unpredictable or moving target, such as a random fault. Conversely, random links lead to a high probability of connecting together all of the elements in a set of nodes, provided their density is sufficient. While this double dose of unpredictability

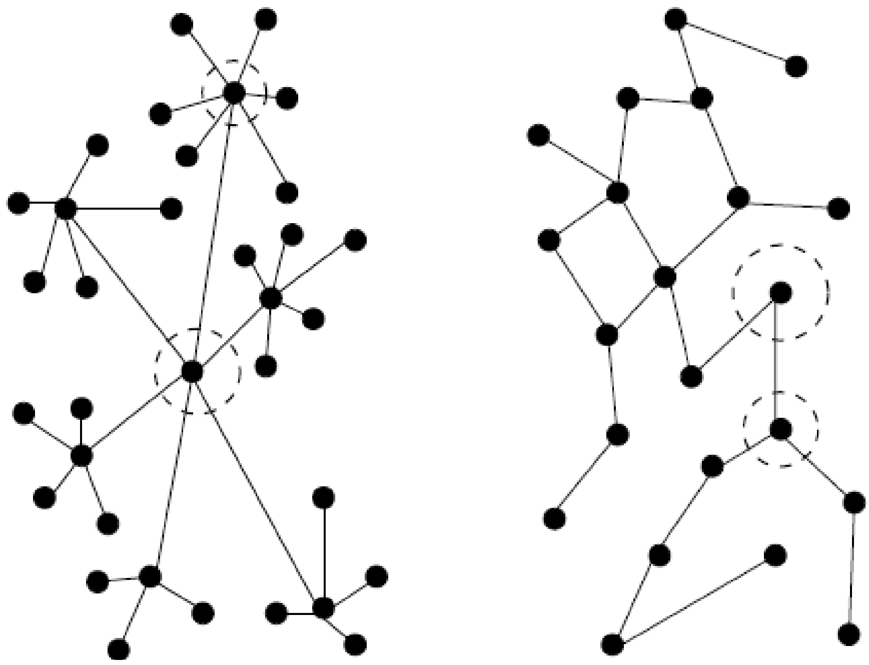


Figure 2: Network tolerance to node removal: nodes are more important than connectors.

Sounds like an unlikely combination for success; it works and leads to highly robust networks.

Humans do not build technology at random so, apart from a robustness to failure, why should random networks be of interest to the study of human–computer systems? The answer lies in so-called *small-world* networks that approximate random ones.

3.2 Faults

The IEEE classification of software anomalies is,

- Operating system crash
- Program hang-up
- Program crash
- Input problem
- Output problem
- Failed required performance
- Perceived total failure
- System error message
- Service degraded
- Wrong output
- No output.

This classification touches on a variety of themes, all of which might plague the interaction between users and an operating system. Some of these issues encroach on the area of performance tuning, e.g. service degraded. Performance tuning is certainly related to the issue

of availability of network services and thus this is a part of system administration. However, performance tuning is of only peripheral importance compared with the matter of possible complete failure.
Many of the problems associated with system administration can be attributed to input problems (incorrect or inappropriate configuration) and failed performance through loss of resources. Unlike many software situations these are not problems which can be eliminated by re-evaluating individual software components.

Another source of error is found at the human edge of the system:

- Management error
- Miscommunication
- Forgetfulness
- Misunderstanding/miscommunication
- Misidentification
- Confusion/stress/intoxication
- Ignorance
- Carelessness
- Slowness of response
- Random procedural errors
- Systematic procedural errors
- Inability to deal with complexity
- Inability to cooperate with others.

In system administration the problems are partly social and partly due to the cooperative nature of the many interacting software components. The unpredictability of operating systems is dominated by these issues.

3.2.1 How are faults corrected?

Faults occur for a plethora of reasons, too complex to present in any summarial fashion. Sometimes diagnosing a fault can take days or even weeks. In spite of this, a working solution to the fault is often extremely simple. It might be as simple as restarting a process, killing a process, editing a file, changing the access rights (permissions) to a file object, and so on. The complexity of fault diagnosis originates from the same place as the complexity of the system: i.e. that operating systems are cooperative systems with intricate causal relationships. It is usually these causal relationships which are difficult to diagnose, not the measurable effects which they have on the system. Such causal relationships make useful studies to publish in journals, since they document important experience.

The root cause of a fault is often not important to the running of the system in practice. One may complain about buggy software, but system administrators are not always in a position to fix the software, nor is it rational for them to do so. While everyone agrees that the fault needs to be fixed at source, the system must continue to function in lieu of that time. Once a fault has been successfully diagnosed it is usually a straightforward matter to find a recipe for preventing the problem, or for curing it, if it should occur again. Problem diagnosis is way beyond the abilities of current software systems except in the simplest cases, so the best one could do would be to capture the experience of a human administrator using a knowledge-based expert system. In artificial intelligence studies expert systems are not the only approach to diagnosis. Another approach, for instance, is the use of genetic algorithms. Such algorithms can be fruitful when looking for trends in statistical data, but statistically meaningful data are seldom available in system administration. The nature of most problems is direct cause and effect, perhaps with a cascade or domino effect. That is not to say that statistical data cannot be used in the future. However, at present no such data exist and no one knows what such data are capable of revealing about system behavior.

Suppose we abstract an operating system by considering it as the sum of its interfaces and resources. There is only a handful of operations which can be performed on this collection of objects and so this set of basic primitives is the complete toolbox of a system administrator. One can provide helpful user interfaces to execute these primitives more easily but no greater functionality is possible. The basic primitives are:

- Examining files
- Creating files
- Aliasing files
- Replacing files
- Renaming files
- Removing files
- Editing files
- Changing access rights on files
- Starting and stopping processes or threads
- Signaling processes or threads
- Examining and configuring hardware devices.

From these primitives one may build more complex operations such as frequently required tasks for sharing resources. Note that the difference between a thread and a process is not usually relevant for the system administrator, so we shall speak mainly of processes and ignore the concept of a thread. The reason for this is that kernel-level threads are usually transparent or invisible to processes and user-level threads cannot normally be killed or restarted without restarting an entire process.

3.2.2 Fault report and diagnosis

When problems arise, one needs to develop a systematic approach to diagnosing the error and to getting the system on its feet again. As in the field of medicine, there is only a limited number of symptoms which a body or computer system can express (sore throat, headache, fever, system runs sluggishly, hangs etc). What makes diagnosis difficult is that virtually all ailments therefore lead to the same symptoms. Without further tests, it is thus virtually impossible to determine the cause of symptoms.

A distressing habit acquired from the home computer revolution is the tendency to give up before even attempting a diagnosis and simply reboot the computer. This might bypass an immediate problem but we learn nothing about why the problem arose. It is like killing a patient and replacing him with another. The act of rebooting a computer can have unforeseen effects on what other users are doing, disrupting their work and perhaps placing the security of data in jeopardy. Rather we need to carefully examine the evidence on a process by process and file by file basis.

3.2.3 Error reporting

Reporting a health problem is the first step to recognizing its importance and solving it. Users tend to fall into the categories of active and passive users. Active users do not need encouraging to report problems. They will usually report even the smallest of problems; sometimes they will even determine the cause and report a fix. While they can often be wearisome in a stressful situation, active users of this type are our friends and go a long way to spreading the burden of problem solving. Remember the community principle of delegation: if we cannot make good use of resources, then the community is not working.
Active users are sometimes more enthusiastic than they are experienced, however, so the system administrator's job is not simply to accept on trust what they say. Their claims need to be verified and perhaps improved upon. Sometimes, users' proposed solutions cannot be implemented because they are in conflict with the system policy, or because the solution would break something else. Only the system administrator has that kind of bird's-eye view of the system to make the judgment.

In contrast to active users, passive users normally have to be encouraged to report errors. They will fumble around trying to make something work, without understanding that there is necessarily a problem. Help desk systems such as Rust, Gnats, Nearnet, Netlog, PTS, QueueMH and REQ can help in this way, but they can also encourage reports of problems which are only misunderstandings.

Suggestion (FAQs). *Providing users with a road-map for solving problems, starting with Frequently Asked Questions and ending with an error report, can help to rationalize error reporting.*

3.2.4 A diagnostic principle

Once an error has been reported, we must determine its cause. A good principle of diagnostics comes from an old medical adage: When you hear the sound of distant hooves, think horses not zebras, i.e.

Principle (Diagnostics). *Always eliminate the obvious first.*

What this means is that we should always look for the most likely explanation before toying with exotic ideas. It is embarrassing to admit how many times apparently impossible problems have resulted from a cable coming out, or forgetting to put in a plug after being distracted in the middle of a job. If the screen is dark, is it plugged in, is the brightness turned up, is the picture centered? Power failures, loose connections, and accidentally touching an important switch can all confuse us. Since these kinds of accident are common, it is logical to begin here.

Nothing is too simple or menial to check. A systematic approach, starting with simple things and progressing through the numbers often makes light work of many problems. The urge to panic is often strong in novices, when there is no apparent explanation; with experience, however, we can quell the desire to run for help. A few tests will almost always reveal a problem. Experience allows us to expand our repertoire and recognize clues, but there is no reason why cold logic should not bring us home in every case.

Having eliminated the obvious avenues of error, we are led into the murkier waters of fault diagnosis. When a situation is confusing, it is of paramount importance to keep a clear head. Writing down a log of what we try and the effect it has on the problem prevents a forgetful mind from losing its way. Drawing a conceptual map of the problem, as a picture, is also a powerful way of persuading the human mind to do its magic.

Once of the most powerful features of the human mind (the thing which makes it, by far, the most powerful pattern-recognition agent in existence) is its ability to associate information input with conceptual models from previous experience. Even the most tenuous of resemblances can lead us to be amused at a likeness of a person or object, seen in an unusual context. We recognize human faces in clouds and old cars; we recognize a song from just a few notes. The ability to make connections leads us in circles of thought which sooner or later lead to 'inspiration'. As most professionals know, however, inspiration is seldom worth waiting for. A competent person knows how to work through these mental contortions systematically to come up with the same answer. While this might be a less romantic notion than waiting for inspired enlightenment, it is usually more efficient.

3.2.5 Establishing cause and effect

If a problem has arisen, then something in the system is different than it was before the error occurred. Our task then is to determine the source of that change and identify a chain of events which resulted in the unfortunate effect. The hope is that this will tell us whether or not we can prevent the problem from recurring and perhaps also whether or not we can fix it. It is not merely so that we can fill out a report in triplicate that we need to debug errors.

Problem diagnosis is one of the hardest problems in any field, be it system administration, medicine or anything else. Once a cause has been found, a cure can be simple, but finding the problem itself often requires experience, a large knowledge base and an active imagination. There is a three-stage process:

- Gather evidence from users and from other tests.
- Make an informed guess as to the probable cause.
- Try to reproduce (or perhaps just fix) the error.

It is only when we have shown that a particular change can switch the error on or off that we can say with certainty what the cause of the error was.

Sometimes it is not possible to directly identify the causal chain which led to an error with certainty. Trying to reproduce a problem on an unimportant host is one way of verifying a theory, but this will not always work. Computers are complex systems which are affected by the behavior of users, interactions between subsystems, network traffic, and any combination of these things. Any one of these factors can have changed in the meantime. Sometimes it can be a chance event which creates a unique set of conditions for an error to occur. Usually this is not the case though; most problems are reproducible with sufficient time and imagination.

Trying to establish probable cause in such a web of intrigue as a computer system is enough to challenge the best detective. To employ a tried and tested strategy, in the spirit of Sherlock Holmes, we can gradually eliminate possibilities and therefore isolate the problem, little by little. This requires a certain inspiration for hypothesizing causes which can be found from any number of sources.

• One should pay attention to all the facts available about the problem. If users have reported it, then one should take seriously what they have to say, but always attempt to verify the facts before taking too much on trust.
• Reading documentation can sometimes reveal simple misunderstandings in configuration which would lead to the problem.
• Talking to others who might have seen the problem before can provide a short cut to the truth. They might have done the hard work of diagnosis before.
Again, their solutions need to be verified before taking them on trust.
• Reading old bug and problem reports can provide important clues.
• Examining system log files will sometimes provide answers.
• Performing simple tests and experiments, based on a best-guess scenario, sharpens the perception of the problem and can even allow the cause to be pinpointed.
• If the system is merely running slower than it should, then some part of it is struggling to allocate resources. Is the disk nearing full, or the memory, or even the process table? Entertain the idea that it is choking in garbage. For instance, deleted files take up space on systems like Novell, since the files are stored in such a way that they can be undeleted. One needs to purge the filesystem every so often to remove these, or the system will spend much longer than it should looking for free blocks. Unix systems thrash when processes build up to unreasonable levels. Garbage collection is a powerful tool in system maintenance. Imagine how human health would suffer if we could never relieve ourselves of dead cells or the byproducts of a healthy consumption. All machines need to do this.

Ideally, one would have a control measurement (‘baseline’) of the system, so that one has a set of measurements when the system is working normally for comparison.

3.2.6 Gathering evidence

From best guess to verification of fault can be a puzzling time in which one grapples with the possible explanations and seeks tests which can confirm or deny their plausibility. One could easily write a whole book exemplifying techniques for troubleshooting, but that would take us beyond the limits set for this book. Let us just provide two simplified examples of real cases which help to illustrate how the process of detection can proceed.

Example (Network services become unavailable). *A common scenario is the sudden disappearance of a network service, like, say, the WWW from a site. If a network service fails to respond it can only be due to a few possibilities:*

- *The service has died on the server host.*
- *The line of communication has been broken.*
- *The latency of the connection is so long that the service has timed-out.*

A natural first step is to try to send a 'ping' to the server-host:

ping www.domain.country *to see whether it is alive. A ping signal will normally return with an answer within a couple of seconds, even for a machine halfway across the planet. If the request responds with*

```
www.domain.country is alive
```

then we know immediately that there is an active line of communication between our host and the server hosts and we can eliminate the second possibility. If the ping request does not return, then there are two further possibilities:

- *The line of communication is broken.*
- *The DNS lookup service is not responding.*

The DNS service can hang a request for a long period of time if a DNS server is not responding. A simple way to check whether the DNS server is at fault or not is to bypass it, by typing the IP address of the WWW server directly:

```
ping -n 128.39.74.4
```

If this fails to respond then we know that the fault was not primarily due to the name service. It tends to suggest a broken line of communication. The traceroute *command on Unix-like operating systems, or* tracert *on Windows can be used to follow a net connection through various routers to its destination. This often allows us to narrow down the point of failure to a particular group of cables in the network. If a network break has persisted for more than a few minutes, a ping or traceroute will normally respond with the message*

```
ICMP error: No route to host
```

and this tells us immediately that there is a network connectivity problem.

But what if there is no DNS problem and the ping tells us that the host is alive? Then the natural next step is to verify that the WWW service is actually running on the server host. On a Unix-like OS we can simply log onto the server host (assuming it is ours) and check the process table for the httpd *daemon which mediates the WWW service*

```
ps waux | grep httpd
ps –elf   | grep httpd
```

for BSD and Sys V Unices respectively. On a Windows machine, we would have to go to the host physically and check its status. If the WWW service is not running, then we would like to know why it stopped working. Checking log files to see what the server was doing when it stopped working can provide clues or even an answer. Sometimes a server will die because of a bug in the program. It is a simple matter to start the service again. If it starts and seems to work normally afterwards, then the problem was probably a bug in the program. If the service fails to start, then it will log an error message of some kind which will tell us more. One possibility is that someone has changed something in the WWW service's configuration file and has left an error behind. The server can no longer make sense of its configuration and it gives up. The error can be rectified and the server can be restarted.

What if the server process has not died? What if we cannot even log onto the server host? The latter would be a clear indication that there was something more fundamentally wrong with the server host. Resisting the temptation to simply reboot it, we could then try to test other services on the server host to see if they respond. We already know that the ping echo service is responding, so the host is not completely dead (it has power, at least). There are therefore several things which could be wrong:

- *The host is unable to respond (e.g. it is overloaded).*
- *The host is unwilling to respond (e.g. a security check denying access to our host).*

We can check that the host is overloaded by looking at the process table to see what is running. If there is nothing to see there, the host might be undergoing a denial of service attack. A look at netstat *will show how many external connections are directed towards the host and their nature. This might show something that would confirm or deny the attack theory. An effective attack would be difficult to prevent, so this could be the end of the line for this particular investigation and the start of a new one, to determine the attacker. If there is no attack, we could check that the DNS name service is working on the server-host. This could cause the server to hang for long periods of time. Finally, there are lots of reasons why the kernel itself might prevent the server from working correctly: the TCP connection close time in the kernel might be too long, leading to blocked connections; the kernel itself might have gone amok; a full disk might be causing errors which have a knock-on effect (the log files from the server might have filled up the disk), in which case the disk problem will have to be solved first. Notice how the DNS and disk problems are problems of* dependency*: a problem in one service having a knock-on effect in another.*

Example (Disks suddenly become full). *A second example, with a slightly surprising conclusion, begins with an error message from a program telling us that the system disk of a particular host has become full. The nature of this particular problem is not particularly ambiguous. A full disk is a disk with no space left on it. Our aim is to try to clear enough space to get the system working again, at least until a more permanent solution can be found. In order to do this, we need to know why the disk became full. Was it for legitimate reasons, or because of a lack of preventative garbage collection, or in this case a completely different reason? There are many reasons why a disk partition might become full. Here are some obvious ones:*

- *A user disk partition can become full if users download huge amounts of data from the Internet, or if they generate large numbers of temporary files. User disks can become full both for valid reasons and for mischievous reasons.*
- *The contents of the system disk only change for one of two reasons: log files which record system activity can grow and fill up a disk; temporary files written to public directories can grow and fill a disk.*

If a user disk becomes full, it is usually possible to find some unnecessary files which can be deleted in order to make space temporarily. The files we deem as unnecessary have to be defined as such as a matter of policy. *It would be questionable ethically to make a habit of deleting files which users did not know could be removed, in advance. Some administrators follow the practice of keeping a large file on every disk partition which can be removed to make space. Of course, if we have done our preventative maintenance, then there should not be any junk files taking up space on the system. In the end, all user disk usage grows monotonically and new disks have to be bought, users can be moved to new disks to spread the load, and so on.*

If a system disk becomes full, there are three main things to look for:

- Core *files (Unix)*
- *Temporary files*
- *Log files.*

Core files are image files which are dumped when programs crash. They are meant to be used for debugging purposes; in practice they cause more problems than they solve. Core files are very large and one or two can easily fill a tight partition, though disk sizes are always growing and giving us more playing room. Preventative maintenance should delete such files regularly. Temporary files /tmp *and* /var/tmp *in Unix-like systems, or* C:\Temp *on Windows are publicly writable directories which usually take up space on the system disk. Temporary files can also be written elsewhere. These can be filled up either accidentally or maliciously. Again, these should be cleared regularly. The final source of trouble is log files. Log files need to be* rotated *on a regular basis so that they do not grow too large. Rotation means starting a new log and saving a small number of old log files. This means that old log data eventually get thrown away, rather than keeping it forever.*

In all of the above cases, we can identify the recent change in a filesystem by searching for files which have changed in the last 24 hours. On a Unix-like system, this *is easily done by running a command to look at all subdirectories of the current directory:*

```
Find . –mtime -1 -print -xdev
```

On other systems it is harder and requires special software. A GNU version of the Unix find *utility is available for Windows.*
A third reason why a filesystem can become full is corruption. *In one instance a Unix disk continued to grow, despite verifying that no new files had been created and after removing all old log files. The Unix* df *disk utility eventually reported that the filesystem was 130% full (an*

impossibility) and it continued to grow. The eventual cause of this problem was identified as a fault in the filesystem structure, or inode corruption. This was brought about by the host concerned overheating and causing memory errors (system log errors confirmed memory write errors). The problem recurred twice before the host was moved to a cooler environment, after which time it righted itself (though the filesystem had to be repaired with fsck *on each occasion).*

There are many tips for tracing the activity of programs. For instance, to trace what files are read by a program, use strace or truss to watch for file descriptors

```
 Truss  -t  open,close  program
```

This runs the program concerned in a monitor which prints out all the listed system calls. This can be a good way of finding out which libraries a program uses (or tries and fails to use) or which configuration files it opens.

Complete your own list of troubleshooting tips. This is a list you will be building for the rest of your life.

SELF ASSESSMENT EXERCISES

1. Describe some typical strategies for finding faults.
2. Describe some typical strategies for correcting faults.
3. Establish cause and effect of faults
4. What kind of faults can occur in a human–computer system?

4.0 CONCLUSION

From best guess to verification of fault can be a puzzling time in which one grapples with the possible explanations and seeks tests which can confirm or deny them. One could easily write a whole book exemplifying techniques for troubleshooting. However, we provide simplified examples of real cases which help to illustrate how the process of detection can proceed and be corrected.

5.0 SUMMARY

All complex systems behave unexpectedly some of the time. How do errors penetrate a system? Faults travel from part to part as if in a network of interconnections. If errors can propagate freely, then a small error in one part of a system can have consequences for another part. By studying different kinds of network, we can learn about the likelihood of error propagation.

Faults occur for a plethora of reasons, too complex to present in any summarial fashion. Sometimes diagnosing a fault can take days or even weeks. In spite of this, a working solution to the fault is often extremely simple. It might be as simple as restarting a process, killing a process, editing a file, changing the access rights (permissions) to a file object, and so on.

When problems arise, one needs to develop a systematic approach to diagnosing the error and to getting the system on its feet again. As in the field of medicine, there is only a limited number of symptoms which a body or computer system can express (sore throat, headache, fever, system runs sluggishly, hangs etc). What makes diagnosis difficult is that virtually all ailments therefore lead to the same symptoms. Without further tests, it is thus virtually impossible to determine the cause of symptoms.

If a problem has arisen, then something in the system is different than it was before the error occurred. Our task then is to determine the source of that change and identify a chain of events which resulted in the unfortunate effect. The hope is that this will tell us whether or not we can prevent the problem from recurring and perhaps also whether or not we can fix it. It is not merely so that we can fill out a report in triplicate that we need to debug errors.

6.0 TUTOR-MARKED ASSIGNMENTS

1. Describe some typical strategies for finding faults.

2. Describe some typical strategies for correcting faults.

3. Describe the process you would use to troubleshoot a slowly running host.
   Formalize this process as an algorithm.

4. Today CPU power is cheap; previously it was common for organizations to have to load users and services onto a single host with limited CPU. Describe as many strategies as you can that you might use to prevent users from hogging CPU-intensive services.

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Burke, J. R.(2004). Network Management Concepts and Practice: a Hands-on Approach. Pearson.

3. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

4. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley.

5. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

6. Subramanian, M. (2000). Network Management: Principles and Practice, Addison-Wesley.

UNIT 4: MONITORING AND SYSTEM PERFORMANCE TUNING

1.0 INTRODUCTION

Having set policy and implemented it to some degree, it is important to verify the success of this programme by measuring the state of the system. In this unit we explore this, as well as, discuss some key performance bottlenecks.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- Know about monitoring tools.
- State the fundamental principle of any performance analysis and its corollary.
- Know about software tuning and kernel configuration
- Know how to do performance tuning

3.0 MAIN CONTENT

3.1 MONITORING

Various monitoring tools exist, depending upon the level at which we wish to evaluate a system:

- Machine performance level
- Abstract policy level.

While these two levels are never unrelated, they pose somewhat different questions.

A very interesting idea which might be used both in fault diagnosis and security intrusion detection is the idea of *anomaly detection*. In anomaly detection we are looking for anything abnormal. That could come from abnormal traffic, patterns of kernel activity, or changes in the statistical profiles of usage. An anomaly can be responded to as a punishable offence, or as a correctable transgression that leads to regulation of behavior, depending on its nature and the policy of the system administrator (see figure 1).

Today much effort is aimed at detecting anomalies for security related intrusion detection rather than for general maintenance, or capacity planning. This has focused attention on mainly short-term changes; however, long-term changes can also be of interest in connection with maintenance of host state and its adaptability to changing demand.

SNMP tools such as MRTG, RRDtool and Cricket specialize in collecting data from SNMP devices like routers and switches. Cfengine’s environment daemon adopts a less deterministic approach to anomaly detection over longer time scales, that can be used to trigger automated policy countermeasures. For many, monitoring means feeding a graphical representation of the system to a human in order to provide an executive summary of its state.

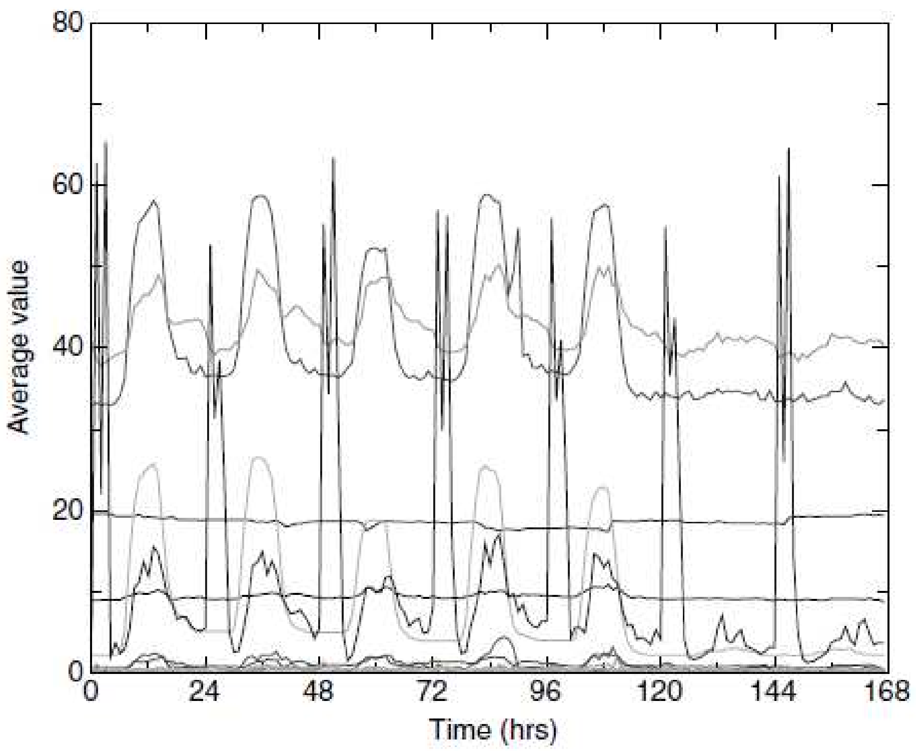


Figure 1: An average summary of system activity over the course of a week, as generated by cfengine’s environment daemon.

3.2 System performance tuning

When is a fault not a fault? When it is an inefficiency. Sooner or later, user perception of system performance passes a threshold. Beyond that threshold we deem the performance of a computer to be unacceptably slow and we become irritated. Long before that happens, the system itself recognizes the symptoms of a lack of resources and takes action to try to counter the problem, but not always in the way we would like.

Efficiency and users’ perception of efficiency are usually two separate things. The host operating system itself can be timesharing perfectly and performing real work at a break-neck pace, while one user sits and waits for minutes for something as simple as a window to refresh. For anyone who has been in this situation, it is painfully obvious that system performance is a highly subjective issue. If we aim to please one type of user, another will be disappointed. To extract maximal performance from a host, we must focus on specific issues and make particular compromises. Note that the system itself is already well adjusted to share resources: that is what a kernel is designed to do. The point of performance tuning is that what is good for one task is not necessarily good for another. Generic kernel configurations try to walk the line of being adequate for everyone, and in doing so they are not great at doing any of them in particular. The only way we can truly achieve maximal performance is to specialize. Ideally, we would have one host per task and optimize each host for that one task. Of course this is a huge waste of resources, which is why multitasking operating systems exist. The inevitability of sharing resources between many tasks is to strike compromise. This is the paradox of multitasking.

Our modest aim in this material is, as usual, to extract the essence of the topic, pointing fingers at the key performance bottlenecks. If we are to tune a system, we need to identify what it is we wish to optimize, i.e. what is most important to us. We cannot make everything optimal, so we must pick out a few things which are most important to us, and work on those.
System performance tuning is a complex subject, in which no part of the system is sacrosanct. Although it is quite easy to pin-point general performance problems, it is harder to make general recommendations to fix these. Most details are unique to each operating system. A few generic pointers can nonetheless offer the greatest and most obvious gains, while the tweaking of system-dependent parameters will put the icing on the cake.

In order to identify a problem, we must first measure the performance. Again there are the two issues: *user perception* of performance (interactive response time) and *system throughput* and we have to choose the criterion we wish to meet. When the system is running slowly, it is natural to look at what resources are being tested, i.e.

- What processes are running
- How much available memory the system has
- Whether disks are being used excessively
- Whether the network is being used heavily
- What software dependencies the system has (e.g. DNS, NFS).

The last point is easy to overlook. If we make one host dependent on another then the dependant host will always be limited by the host on which it depends. This is particularly true of file-servers (e.g. NFS, DFS, Netware distributed filesystems) and of the DNS service.

Principle (Symptoms and cause). *Always try to fix problems at the root, rather than patching symptoms.*

3.2.2 Hardware

The fundamental principle of any performance analysis is:

Principle (Weakest link). *The performance of any system is limited by the weakest link amongst its components. System optimization should begin with the* source. *If performance is weak at the source, nothing which follows can make it better.*

Obviously, any effect which is introduced after the source will only reduce the performance in a chain of data handling. A later component cannot 'suck' the data out of the source faster than the source wants to deliver it. This tells us that the logical place to begin is with the system hardware. A corollary to this principle follows from a straightforward observation about hardware. As Scotty said, we cannot change the laws of physics:

Corollary to principle (Performance). *A system is limited by its slowest moving parts. Resources with slowly moving parts, like disks, CD-ROMs and tapes, transfer data slowly and delay the system. Resources which work purely with electronics, like RAM memory and CPU calculation, are quick. However, electronic motion/communication over long distances takes much longer than communication over short distances (internally within a host) because of impedances and switching.*

Already, these principles tell us that RAM is one of the best investments we can make. Why? In order to avoid mechanical devices like disks as much as possible, we store things in RAM; in order to avoid sending unnecessary traffic over networks, we cache data in RAM. Hence RAM is the primary workhorse of any computer system. After we have exhausted the possibilities of RAM usage, we can go on to look at disk and network infrastructure.

- *Disks:* When assigning partitions to new disks, it pays to use the fastest disks for the data which are accessed most often, e.g. for user home directories. To improve disk performance, we can do two things. One is to buy faster disks and the other is to use *parallelism* to overcome the time it takes for physical motions to be executed. The mechanical problem which is inherent in disk drives is that the heads which read and write data have to move as a unit. If we need to collect two files concurrently which lie spread all over the disk, this has to be done *serially*. *Disk striping* is a technique whereby filesystems are spread over several disks. By spreading files over several disks, we have several sets of disk heads which can seek independently of one another, and work in parallel. This does not necessarily increase the transfer rate, but it does lower seek times, and thus performance improvement can approach as much as *N* times with *N* disks. RAID technologies employ striping techniques and are widely available commercially. GNU/Linux also has RAID support. Spreading disks and files across multiple disk controllers will also increase parallelism.
- *Network:* To improve network performance, we need fast interfaces. All interfaces, whether they be Ethernet or some other technology, vary in quality and speed. This is particularly true in the PC world, where the number of competing products is huge. Network interfaces should not be trusted to give the performance they advertise. Some interfaces which are sold as 100Mbits/sec, Fast Ethernet, manage little more than 40Mbits/sec. Some network interfaces have intelligent behavior and try to detect the best available transmission rate. For instance, newer Sun machines use the *hme* fast Ethernet interface. This has the ability to detect the best transmission protocol for the line a host is connected to. The best transmission type is 100Mbits/sec, full duplex (simultaneous send and receive), but the interface will switch down to 10Mbits/sec, half duplex (send or receive, one direction at a time) if it detects a problem. This can have a huge performance effect. One problem with auto-detection is that, if both ends of the connection have auto-detection, it can become an unpredictable matter which speed we end up with. Sometimes it helps to try setting the rate explicitly, assuming that the network hardware supports that rate. There are other optimizations also, for TCP/IP tuning.

The sharing of resources between many users and processes is what networking is about. The competition for resources between several tasks leads to another performance issue.

Principle (Contention/competition). *When two processes compete for a resource, performance can be dramatically reduced as the processes fight over the right to use the resource. This is called contention. The benefits of sharing have to be weighed against the pitfalls.*

Contention could almost be called a strategy, in some situations, since there exist technologies for avoiding contention altogether. For example, Ethernet technology allows contention to take place, whereas Token Ring technology avoids it. We shall not go into the arguments for and against contention. Suffice it to say that many widely used technologies experience this problem.

- *Ethernet collisions:* Ethernet communication is like a television panel of politicians: many parties shouting at random, without waiting for others to finish. The Ethernet cable is a shared bus. When a host wishes to communicate with another host, it simply tries. If another host happens to be using the bus at that time, there is a collision and the host must try again at random until it is heard. This method naturally leads to contention for bandwidth. The system works quite well when traffic is low, but as the number of hosts competing for bandwidth increases, the probability of a collision increases in step. Contention can only be reduced by reducing the amount of traffic on the network segment. The illusion of many collisions can also be caused by incorrect wiring, or incorrectly terminated cable, which leads to reflections. If collision rates are high, a wiring check might also be in order.
- *Disk thrashing:* Thrashing is a problem which occurs because of the slowness of disk head movements, compared with the speed of kernel time-sharing algorithms. If two processes attempt to take control of a resource simultaneously, the kernel and its device drivers attempt to minimize the motion of the heads by queuing requested blocks in a special order. The algorithms really try to make the disks traverse the disk platter uniformly, but the requests do not always come in a predictable or congenial order. The result is that the disk heads can be forced back and forth across the disk, driven by different processes and slowing the system to a virtual standstill. The time for disk heads to move is an eternity to the kernel, some hundreds of times slower than context switching times.

  An even worse situation can arise with the virtual memory system. If a host begins paging to disk because it is low on memory, then there can be simultaneous contention both for memory and for disk. Imagine, for instance, that there are many processes, each loading files into memory, when there is no free RAM. In order to use RAM, some has to be freed by paging to disk; but the disk is already busy seeking files. In order to load a file, memory has to be freed, but memory can't be freed until the disk is free to page, this drags the heads to another partition, then back again ... and so on. This nightmare brings the system to a virtual standstill as it fights both over free RAM and disk head placement. The system spends more time juggling its resources than it does performing real work, i.e. the overhead to work ratio blows up. The only cure for thrashing is to increase memory, or reduce the number of processes contending for resources.

A final point to mention in connection with disks is to do with standards. Disk transfer rates are limited by the protocols and hardware of the disk interfaces. This applies to the interfaces in the computer and to the interfaces in the disks. Most serious performance systems will use SCSI disks. However, there are many versions of the SCSI disk design. If we mix version numbers, the faster disks will be delayed by the slower disks while the bus is busy, i.e. the average transfer rate is limited by the weakest link or the slowest disk. If one needs to support legacy disks together with new disks, then it pays to collect like disks with a special host for each type, or alternatively buy a second disk controller rather than to mix disks on the same controller.

3.2.3 Software tuning and kernel configuration

It is true that software is constrained by the hardware on which it runs, but it is equally true that hardware can only follow the instructions it has received from software. If software asks hardware to be inefficient, hardware will be inefficient. Software introduces many inefficiencies of its own. Hardware and software tuning are inextricably intertwined.

Software performance tuning is a more complex problem than hardware performance tuning, simply because the options we have for tuning software depend on what the software is, how it is written and whether or not the designer made it easy for us to tune its performance. Some software is designed to be stable rather than efficient. Efficiency is not a fundamental requirement; there are other priorities, such as simplicity and robustness.

In software the potential number of variables is much greater than in hardware tuning. Some software systems can be tuned individually. For instance, high availability server software such as WWW servers and SMTP (E-mail) servers can be tuned to handle traffic optimally for heavy loads.

More often than not, performance tuning is related to the availability or sharing of system resources. This requires tuning the system kernel. The most configurable piece of software on the system is the kernel. All Unix-like systems kernel parameters can be altered and tuned. The most elegant approach to this is taken by Unix SVR4, and Solaris. Here, many kernel parameters can be set at run time using the kernel module configuration command ndd. Others can be configured in a single file /etc/system. The parameters in this file can be set with a reboot of the kernel, using the reconfigure flag

```
reboot - -  -r
```

For instance, on a heavily loaded system which allows many users to run external logins, terminals, or X-terminal software, we need to increase many of the default system parameters. The maxusers parameter (actually in most Unix-like systems) is used as a guide to estimating the size of many tables and limits on resources. Its default value is based on the amount of available RAM, so one should be careful about changing its value in Solaris, though other OSs are less intelligent. Solaris also has a separate parameter pt_cnt for extending the number of virtual terminals (pty's). It is possible to run out if many users are logged in to the same host simultaneously. Many graphics-intensive programs use shared memory in large blocks. The default limit for shared memory segments is only a megabyte, so it can be increased to optimize for intensive graphics use, but should not be increased on heavily loaded file-servers, where memory for caching is more important. The file /etc/system, then looks like this:

```
set maxusers=100
set shmsys:shminfo_shmmax = 0x10000000
set pt_cnt=128
```

After a reboot, these parameters will be set. Some caution is needed in editing this file. If it is non-existent or unparsable, the host will not be able to boot (a questionable design feature). The ndd command in Solaris can be chosen to optimize its over-safe defaults set on TCP/IP connections.

For busy servers which handle many TCP connections, the time it takes an operating system to open and close connections is important. There is a limit on the number of available connections and open sockets; if finished socket connections are not purged quickly from the kernel tables, new connections cannot be opened in their place. On non-tuned hosts, used sockets can hang around for five minutes or longer on a Solaris host. On a heavily loaded server, this is unacceptable. The close time on sockets can be shortened to half a minute so as to allow newer sockets to be opened sooner (though note that this contravenes RFC 793). The parameters can be set when the system boots, or patched at any later time. The times are measured in milliseconds.

```
/usr/sbin/ndd  -set  /dev/tcp  tcp_keepalive_interval 900000
/usr/sbin/ndd  -set  /dev/tcp  tcp_time_wait_interval 30000
```

Prior to Solaris 2.7 (SunOS 5.7) the latter line would have read:

```
/usr/sbin/ndd  -set /dev/tcp  tcp_close_wait_interval 30000
```

which illustrates the futility of documenting these fickle parameters in a static medium like a book. Note that setting these parameters to ultra-short values could cause file transmissions to be terminated incorrectly. This might lead to corruption of data. On a web server, this is a nuisance for the client, but it is not mission-critical data. For security, longer close times are desirable, to ensure correct closure of sockets. After setting these values, the network interface needs to be restarted, by taking it down and up with ifconfig. Alternatively, the values can be configured in a startup script which is executed before the interface is brought up at boot time.

Suggestion. *Do not change operating system defaults unless you have good cause, and really know what you are doing. Deviations from expert defaults must be on a case-by-case basis.*

Most Unix-like operating systems do not permit run-time configuration. New kernels have to be compiled and the values hard-coded into the kernel. This requires not just a reboot, but a recompilation of the kernel in order to make a change. This is not an optimal way to experiment with parameters. Modularity in kernel design can save us memory, since it means that static code does not have to take up valuable memory space. However, the downside of this is that modules take time to load from disk, on demand. Thus a modular kernel can be slower than a statically compiled kernel. For frequently used hardware, static compilation is a must, since it eliminates the load-time for the module, at the expense of extra memory consumption.

The GNU/Linux system kernel is a modular kernel, which can load drivers for special hardware at run time, in order to remain small in the memory. When we build a kernel, we have the option to compile in modules statically. . Tips for Linux kernel configuration can readily be found by searching the Internet, so we shall not reproduce these tips here, where they would quickly become stale.

Windows performance tuning can be undertaken by perusing the multitudinous screens in the graphical performance monitor and editing the values. For once, this useful tool is a standard part of the Windows system.

SELF ASSESSMENT EXERCISES

1. State the fundamental principle of any performance analysis.
2. Explain the weakest link principle in performance analysis.
3. How do you do Software tuning and kernel configuration?

4.0 CONCLUSION

System performance tuning is a complex subject, in which no part of the system is sacrosanct. Although it is quite easy to pin-point general performance problems, it is harder to make general recommendations to fix these. Most details are unique to each operating system. A few generic pointers can nonetheless offer the greatest and most obvious gains, while the tweaking of system-dependent parameters will put the icing on the cake.

5.0 SUMMARY

Various monitoring tools exist, depending upon the level at which we wish to evaluate a system: Machine performance level and Abstract policy level. While these two levels are never unrelated, they pose somewhat different questions.

If we are to tune a system, we need to identify what it is we wish to optimize, i.e. what is most important to us. We cannot make optimal, so we must pick out a few things which are most important to us, and work on those.

In order to identify a problem, we must first measure the performance. Again there are the two issues: *user perception* of performance (interactive response time) and *system throughput* and we have to choose the criterion we wish to meet.

The fundamental principle of any performance analysis is: *The performance of any system is limited by the weakest link amongst its components. System optimization should begin with the* source. *If performance is weak at the source, nothing which follows can make it better.* A corollary to this principle follows from a straightforward observation about hardware. *A system is limited by its slowest moving parts.*

It is true that software is constrained by the hardware on which it runs, but it is equally true that hardware can only follow the instructions it has received from software. If software asks hardware to be inefficient, hardware will be inefficient. Software introduces many inefficiencies of its own. Hardware and software tuning are inextricably intertwined.

6.0 TUTOR-MARKED ASSIGNMENTS

1. Suppose you are performance tuning, trying to find out why one host is slower than another. Write a program which tests the efficiency of *CPU-intensive work only*. Write programs which test the speed of *memory-intensive work* and *disk-intensive* work. Would comparing the time it takes to compile a program on the hosts be a good way of comparing them?

2. What role does monitoring the system play in a rational decision-making process?

3. Explain *Ethernet collisions and Disk thrashing.*

4. Discuss the monitoring tools in your organization.

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

3. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley

4. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

MODULE 4: NETWORK SIMULATION, DOCUMENTATION AND SECURITY

UNIT 1: NETWORK SIMULATION AND DOCUMENTATION

UNIT 2: NETWORK SECURITY

UNIT 3: OUTLOOK AND THE FUTURE OF NETWORK ADMINISTRATION

UNIT 1: NETWORK SIMULATION AND DOCUMENTATION

1.0 INTRODUCTION

In this unit we discuss network simulation as a management tool in conducting network administration. Also, we discuss network documentation as a means of keeping records of where things are, explaining how to do things, and making useful information available to customers.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- Understand what network simulation is
- Give examples of network simulator
- Explain what network documentation is about
- Map out both the physical and logical networks layout

3.0 MAIN POINT

3.1 What is network simulation?

In communication and computer network research, **network simulation** is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities (hosts/routers, data links, packets, etc) using mathematical formulas, or actually capturing and playing back observations from a production network. The behavior of the network and the various applications and services it supports can then be observed in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network would behave under different conditions. When a simulation program is used in conjunction with live applications and services in order to observe end-to-end performance to the user desktop, this technique is also referred to as network emulation.

3.2 Network simulator

A network simulator is a software program that imitates the working of a computer network. In simulators, the computer network is typically modelled with devices, traffic etc and the performance is analysed. Typically, users can then customize the simulator to fulfill their specific analysis needs. Simulators typically come with support for the most popular protocols in use today, such as WLAN, Wi-Max, UDP, and TCP.

3.3 Simulations

Most of the commercial simulators are GUI driven, while some network simulators require input scripts or commands (network parameters). The network parameters describe the state of the network (node placement, existing links) and the events (data transmissions, link failures, etc). Important outputs of simulations are the trace files. Trace files can document every event that occurred in the simulation and are used for analysis. Certain simulators have added functionality of capturing this type of data directly from a functioning production environment, at various times of the day, week, or month, in order to reflect average, worst-case, and best-case conditions. Network simulators can also provide other tools to facilitate visual analysis of trends and potential trouble spots.

Most network simulators use discrete event simulation, in which a list of pending "events" is stored, and those events are processed in order, with some events triggering future events -- such as the event of the arrival of a packet at one node triggering the event of the arrival of that packet at a downstream node.

Some network simulation problems, notably those relying on queueing theory, are well suited to Markov chain simulations, in which no list of future events is maintained and the simulation consists of transiting between different system "states" in a memoryless fashion. Markov chain simulation is typically faster but less accurate and flexible than detailed discrete event simulation. Some simulations are cyclic based simulations and these are faster as compared to event based simulations.

Simulation of networks can be a difficult task. For example, if congestion is high, then estimation of the average occupancy is challenging because of high variance. To estimate the likelihood of a buffer overflow in a network, the time required for an accurate answer can be extremely large. Specialized techniques such as "control variates" and "importance sampling" have been developed to speed simulation.[

Examples of network simulators

Examples of notable network simulation software are, ordered after how often they are mentioned in research papers:

1. ns2/ns3
2. OPNET
3. NetSim

3.4 Documentation

Network documentation takes on many forms, the most fundamental of which is labeling. The need for documentation and the forms it should take are not likely to change with time.

Maps of both the physical and logical networks should be part of the network documentation. The physical-network map should show where the wires go and the end points or ranges of wireless links. If redundancy was part of the physical-network design, it should clearly indicate and document the physically diverse paths. The amount and type of connectivity available for each link should be indicated. For example, if 200 pairs of copper wires and m20 pairs of fiber-optic cables connect a pair of buildings, the documentation should specify how both sets are rated and terminated and the distances between the termination points.

The logical-network map should show the logical-network topology, with network numbers, names, and speeds. This map should also show any routing protocols and administrative domains that vary across the network. Both the physical- and logical-network maps should reach to the perimeter of the organization's network and identify its outer boundaries.

Labeling is the single most important component of the network documentation. Clear, consistent labeling on patch panels and long-distance connections is particularly important. A patch panel should clearly indicate the physical location of the corresponding patch panel or jacks, and each of the connections on the patch panel should be clearly and consistently labeled at both ends. Long-distance connections should clearly indicate where the circuit goes, whom to report problems to, and what information will be required when reporting a problem, such as the circuit ID and where it terminates. Placing this label immediately beside the unit's fault-indicator light can be helpful. Doing so eliminates the need to trace cables to find the necessary information when a fault occurs. For example, one might otherwise have to trace cables from a channel service unit/data service unit to the punch-down block at the telephone company's demarcation point or to a jack on the wall. Less permanent connections, such as the network connection for each host on the network, also should be labeled. Labeling on each wire is easier to maintain in a relatively static environment and more difficult to maintain in a highly dynamic one. You should attempt to do this level of labeling only if you can maintain it. Incorrect labels are worse than none at all.

A compromise between no labels and full cable labeling is to purchase cables with a unique serial number shown at each end. With a serial number, you can quite quickly trace exactly where a cable goes, if you have an approximate idea of the location of the other end. The serial number label can also indicate length and the way that the cable is wired. For example, the first two digits can indicate straight-through, crossover, twisted-pair, FDDI, or other wiring arrangements, followed by a dash, three digits indicating the cable length, another dash, and the serial number. Colored covers on the connectors can also be used to indicate cable type.

Network cables are often difficult to label. One of the most effective ways we have seen is to use a cable tie with a protruding flat tab to which standard sticky labels can be affixed. It is securely attached and can be easily altered.

The other key location for documentation is online, as part of the configuration of the network devices themselves. Wherever possible, comment fields and device names should be used to provide documentation for the network administrators. Naming standards for devices can go a long way toward making network administration easier and more intuitive.

Case Study: Naming Conventions

A midsize multinational software company used a multistar topology for its wide-area connectivity. One of the star centers was in Mountain View, California. The router at each remote site that connected to Mountain View was called location2mtview: for example, denver2mtview or atlanta2mtview. The router at the Mountain View end of the connection was called location-router: for example, denverrouter or atlanta-router, in addition to any other names that it might have.

When a remote site, suffered connectivity problems, everyone could immediately identify which routers served that site, without resorting to network maps or tracing cables. This standardization vastly improved the level of support that remote sites could expect from the average system administrator (SA). All those capable of performing basic network debugging were given read-only access to the network equipment and were able to perform basic diagnostics before handing the problem to the network team. Routers usually permit a text comment to be recorded with each interface. For WAN connections, this comment should include all the information a technician needs in an emergency involving the link going down: the name of the vendor providing the link, the vendor’s phone number, the circuit identifier, and the maintenance contract number that the vendor needs to provide service. For LAN connections, include the name of the subnet and the contact information for the owner of the subnet, if it is not the main SA team. If your LAN equipment has a comment field for each port, use it to indicate the room number and jack at the other end of the cable.

SELF ASSESSMENT EXERCISES

1. What do you understand by network simulation?
2. Give three examples of network simulator
3. Explain what network documentation is about

4.0 CONCLUSION

Network simulation enables us to observe the behavior of a network and the various applications and services it supports in a test lab; various attributes of the environment can also be modified in a controlled manner to assess how the network would behave under different conditions. Network documentation takes on many forms, the most fundamental of which is labeling.

5.0 SUMMARY

In communication and computer network research, **network simulation** is a technique where a program models the behavior of a network either by calculating the interaction between the different network entities (hosts/routers, data links, packets, etc) using mathematical formulas, or actually capturing and playing back observations from a production network. A network simulator is a software program that imitates the working of a computer network.

Maps of both the physical and logical networks should be part of the network documentation. The physical-network map should show where the wires go and the end points or ranges of wireless links. The logical-network map should show the logical-network topology, with network numbers, names, and speeds. This map should also show any routing protocols and administrative domains that vary across the network. Both the physical- and logical-network maps should reach to the perimeter of the organization's network and identify its outer boundaries.

6.0 TUTOR-MARKED ASSIGNMENTS

1. Draw a physical-network map for your organization.
2. Draw a logical-network map for your organization.
3. Use the example case study naming conventions and do same your own organization.
4. Use any available network simulator and produce the listing.

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Burke, J. R.(2004). Network Management Concepts and Practice: a Hands-on Approach. Pearson.

3. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

4. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley.

5. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

6. Subramanian, M. (2000). Network Management: Principles and Practice, Addison-Wesley.

UNIT 2: NETWORK SECURITY

1.0 INTRODUCTION

Security management cannot be separated from network and system administration because security requires a fully systemic approach. However, it is important to identify some principles of security management in isolation, in order to better understand them and underline their importance. In this unit, we dissect security into its constituent parts.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- Understand the principles of security
- Know how to analyze network security
- Know about recovery plan
- Know about authentication methods

3.0 MAIN CONTENT

3.1 Principles of security

Security is about protecting things of value to an organization, in relation to the possible risks. This includes material and intellectual assets; it includes the very assumptions that are the foundation of an organization or human–computer system. Anything that can cause a failure of those assumptions can result in loss, and must therefore be considered a threat. In system administration terms this often means a loss of data or availability in the computing system, but that is really just the tip of the iceberg. The number of ways in which this can occur is vast – making security a difficult problem.

In order to have security, we must sacrifice a certain level of convenience for a measure of discipline. This promotes systems with predictable behavior, where one can arrange to safeguard the system from unpleasant occurrences. To develop computer security by assuring predictability, we have to understand the interrelationships between all of the hosts and services on our networks as well as the ways in which those hosts can be accessed. A system can be compromised by:

- *Physical threats*: weather, natural disaster, bombs, power failures etc.
- *Human threats*: cracking, stealing, trickery, bribery, spying, sabotage, accidents.
- *Software threats*: viruses, Trojan horses, logic bombs, denial of service.

Protecting against these issues requires both pro-active (preventative) measures and damage control after breaches. Our task is roughly as follows:

• Identify what we are trying to protect.
• Evaluate the main sources of risk and where trust is placed.
• Work out possible or cost-effective counter-measures to attacks.

Security is an increasingly important problem. In recent years the number of attacks and break-ins to computer systems has risen to millions of cases a year. Crackers or hackers have found their way inside the computers of the Pentagon, the world's security services, warships, fighter plane command computers, banks and major services such as electrical power grids. With this kind of access the potential for causing damage is great. Computer warfare is the next major battlefield to subdue; it is going on now, as you read these words: it is here, like it or not. It is estimated that the banks lose millions of dollars a year to computer crime.

Security embraces other issues such as reliability. For instance, many computers are used in mission-critical systems, such as aircraft controls and machinery, where human lives are at stake. Thus reliability and safety are also concerns. *Real-time systems* are computer systems which are guaranteed to respond within a well-defined time limit when a request is made of them. This is a kind of quality of service. That means that a real-time system must always be fast enough to cope with any demand which is made of it. Real-time systems are required in cases where human lives and huge sums of money are involved. For instance, in a flight control system it would be unacceptable to give a command

'Oh my goodness, we're going to crash, flaps NOW!'

and have the computer reply with 'Processing, please wait...'.

Security is a huge subject, because modern computer systems are complex and the connectivity of the Internet means that millions of people can try to break into networked systems.
.

3.1.1 Four independent issues

For many, security is regrettably perceived as being synonymous with network privacy or network intrusion. Privacy and intrusion are two particular aspects of security, but the network is not our particular enemy. Many breaches of security happen from within, or by accident. If we focus exclusively on network connectivity we ignore the threats from internal employees (e.g. the janitor who is a computer expert and has an axe to grind, or the mischievous son of the director who was left waiting to play in mom's office, or perhaps the unthinkable: a disgruntled employee who feels as though his/her talents go unappreciated).

Software security is a vast subject, because modern computer systems are complex. It is only exacerbated by the connectivity of the Internet which allows millions of people to have a go at breaking into networked systems. What this points to is the fact that a secure environment requires the control of all parts of a system, not merely at specific access points like login terminals or firewalls.

Principle (Security is a property of systems). *Security is a property of entire systems, not an appendage that can be added in any one place, or be applied at any one time. It relies on the constant appraisal and re-appraisal (the integrity) of our assumptions about a system. There are usually many routes through a system that permit theft or destruction. If we try to 'add security' in one place, an attacker or random chance will simply take a different route.*

If we stretch our powers of abstraction even to include loss by natural disaster, then system security can be summarized by a basic principle.

Principle (Access and privilege). *A fundamental prerequisite for security is the ability to restrict access to data. This leads directly to a notion of privilege for certain users.*

The word privilege does not apply to loss by accident or natural disaster, but the word access does. If accidental actions or natural disasters do not have access to data, then they cannot cause them any harm. Any attempt to run a secure system where restriction of access is not possible is fundamentally flawed.

There are four basic elements in security:

- *Privacy or confidentiality*: restriction of access.
- *Authentication*: verification of presumed identity.
- *Integrity*: protection against corruption or loss (redundancy).
- *Trust*: underlies every assumption.

Some authors include the following as independent points:

- *Availability*: preventing disruption of a service.
- *Non-repudiation*: preventing deniability of actions.

They can also be considered simply as issues of *integrity* of a service (availability) and the imperviousness of accountability logs (non-repudiation).

The most important issue to understand about security is a basic tenet that is widely unappreciated:

Principle (Security is about trust). *Every security problem boils down to a question of whom or what do we trust?*

Once we have understood this, the topic of security is reduced to a litany of examples of how trust may be exploited and how it may be improved using certain technological aids. Failure to understand this point can lead to embarrassing mistakes being made.

We introduce the somewhat ill-defined notion of 'security' to describe protecting ourselves against parties whom we do not trust. But how do we solve this problem? Usually, we introduce some kind of technology to move trust from a risky place to a safer place. For example, if we do not trust our neighbors not to steal our possessions, we might put a lock on our door. We no longer have to trust our neighbors, but we have to trust that the lock will do its job in the way we expect. This is easier to trust, because a simple mechanical device is more *predictable* than complicated human beings, but it can still fail.

If we don't entirely trust the lock, we could install an alarm system which rings the police if someone breaks in. Now we are trusting the lock a little, the alarm system and the police. After all, who says that the police will not be the ones to steal your possessions? In some parts of the world, this idea is not so absurd.

Trust is based on assumption. It can be bolstered with evidence but, just as science can never prove something is true, we can never trust something with absolute certainty. We only know when trust is broken. This is the real insight of security – not the technologies that help us to build trust.

Example. *One of the big problems with security mechanisms is that they hinder people sometimes from taking part in legitimate activities. They are then frequently turned off out of misunderstanding or annoyance, leaving a system unprotected. It is therefore important to educate the managers of security systems about procedures and practices surrounding a secured system. If there is a way to proceed, it should be by an approved channel; if a pathway is blocked, then it should be for a good reason that is understood by all parties.*

3.1.3 Physical security

For a computer to be secure it must be physically secure. If we can get our hands on a host then we are never more than a screwdriver away from all of its assets. Disks can be removed. Sophisticated users can tap network lines and listen to traffic. The radiation from monitor screens can be captured and recorded, showing an exact image of what a user is looking at on his/her screen. Or one can simply look over the shoulder of a colleague while he or she types a password. The level of physical security one requires depends on the sophistication of the potential intruder, and therefore in the value of the assets which one is protecting.

Cleaning staff frequently dust monitors and keyboards, switch off monitors and computers by accident and even pull plugs on computers to plug in their machinery. If a computer serves a valuable purpose, or is vulnerable to accidental input, it is not only attackers we have to protect against. Cleaning staff have keys to the building, so locking an office door will not help here.

Assuming that hosts are physically secure, we still have to deal with the issues of software security which is a much more difficult topic. Software security is about access control and software reliability. No single tool can make computer systems secure. Major blunders have been made out of the belief that a single product (e.g. a 'firewall') would solve the security problem. The bottom line is that there is no such thing as a secure operating system. What is required is a persistent mixture of vigilance and adaptability.

3.1.5 Security policy and definition of security

Security only has meaning when we have defined a frame of reference. It is intrinsically connected to our own appreciation of risks. It must be based on a thorough *risk analysis*.

Principle (Risk). *There is always a non-zero level of risk associated with any system.*

Clearly, there is a finite chance that a dinosaur-killer asteroid will strike your computer and destroy it. This risk is not very high, but it is real. Few organizations would care to try to protect themselves against this eventuality and would consider themselves secure if this were the only remaining threat, but this parodic example makes an important point. Security is intrinsically related to a threat assessment. An acceptable definition of a secure system is:

Definition (Secure system). *A secure system is one in which every possible threat has been analyzed and where all the risks have been assessed and accepted as policy.*

Clearly this is a tall order, and it is probably impossible to realize, but we should be clear about what this means: it is a definition that tells us that security is determined by policy – as *acceptable risk*.

Defining what *we*, a local community, mean by security is essential. Only then will we know when security has been breached, and what to do about it. Some sites, which contain sensitive data, require strict security and spend a lot of time enforcing it, others do not particularly care about their data and would rather not waste their time on pointless measures to protect them. Security must be balanced against convenience. How secure must we be?

- From outside the organization?
- From inside the organization (different host)?
- From inside the organization (same host)?
- Against the interruption of services?
- From user error?

Finally, how much inconvenience are the users of the system willing to endure in order to uphold this level of security? This point should not be underestimated: if users consider security to be a nuisance, they try to circumvent it.

Suggestion (Work defensively). *Expect the worst, do your best, preferably* in advance *of a problem.*
Visible security can be a problem in itself. Systems which do not implement high level security tend to attract only low-level crackers – and those who manage to break in tend to use the systems only as a springboard to go to other places. The more security one implements, and the more visible it is, the more of a challenge it is for a cracker. So spending a lot of time on security might only have the effect of asking for trouble.

Suggestion (Network security). *Extremely sensitive data should not be placed on a computer which is attached in any way to a public network.*

What resources are we trying to protect?

- *Secrets:* Some sites have secrets they wish to protect. They might be government or trade secrets or the solutions to a college exam.
- *Personnel data:* In your country there are probably rules about what you must do to safeguard sensitive personal information. This goes for any information about employees, patients, customers or anyone else we deal with. Information about people is private.
- *CPU usage/System downtime:* We might not have any data that we are afraid will fall into the wrong hands. It might simply be that the system is so important to us that we cannot afford the loss of time incurred by having someone screw it up. If the system is down, everything stops.
- *Abuse of the system:* It might simply be that we do not want anyone using our system to do something for which they are not authorized, like breaking into other systems.

Who are we trying to protect them from?

- *Competitors*, who might gain an advantage by learning your secrets.
- *Malicious intruders.* Note that people with malicious intent might come from inside or outside our organization. It is wrong to think that the enemy is simply everyone outside of our domain. Too many organizations think 'inside/outside' instead of dealing with proper access control. If one *always ensures that systems and data are protected* on a need-to-know basis, then there is no reason to discriminate between inside or outside of an organization.
- *Old employees with a grudge* against the organization.

Next: what will happen if the system is compromised?

- Loss of money
- Threat of legal action against you
- Missed deadlines
- Loss of reputation.

How much work will we need to put into protecting the system? Who are the people trying to break in?

- Sophisticated spies
- Tourists, just poking around
- Braggers, trying to impress.

Finally: *what risk is acceptable?* If we have a secret which is worth 4 lira, would we be interested in spending 5 lira to secure it? Where does one draw the line?
How much is security worth?

The social term in the security equation should never be forgotten. One can spend a hundred thousand dollars on the top of the range firewall to protect data from network intrusion, but someone could walk into the building and look over an unsuspecting shoulder to obtain it instead, or use a receiver to collect the stray radiation from your monitors. Are employees leaving sensitive printouts lying around? Are we willing to place our entire building in a Faraday cage to avoid remote detection of the radiation expelled by monitors? In the final instance, someone could just point a gun at someone's head and ask nicely for their secrets.

3.1.6 System failure modes

Since the explosion of interest in the Internet, the possibility of hosts being attacked from outside sources has become a significant problem. With literally millions of users on the net, the tiny percentage of malicious users becomes a large number. The number of ways in which a system can fail are thus many and varied.

3.1.7 Preventing and minimizing failure modes

Prevention of loss is usually cheaper than recovery after the fact. Any reasonable preventative measures we can take are worth the investment.

3.1.8 Some well-known attacks

There are many ways to attack a networked computer in order to gain access to it, or simply disable it. Some well-known examples are listed below. The actual attack mechanisms used by attackers are often intricate and ingenious, but the common theme in all of them is to exploit naive limitations in the way network services are implemented. Time and again one sees crackers make use of software systems which were written in good faith, by forcing them into unnatural situations where the software fails through inadequate checking.

3.1.8.1 Ping attacks

The RFC 791 specifies that Internet datagrams shall not exceed 64kB. Some implementations of the protocol can send packets which are larger than this, but not all implementations can receive them ping -s 65510 target host.

Some older network interfaces can be made to crash certain operating systems by sending them a 'ping' request like this with a very large packet size. Most modern operating systems are now immune to this problem (e.g. NT 3.51 is vulnerable, but NT 4 is not). If not, it can be combated with a packet filtering router. See http://www.sophist.demon.co.uk/ping/.

3.1.8.2 Denial of service (DoS) attacks

Another type of attack is to overload a system with so many service requests that it grinds to a halt. One example is mail spamming, in which an attacker sends large numbers of repetitive E-mail messages, filling up the server's disk and causing the sendmail daemon to spawn rapidly and slow the system to a standstill.

Denial of service attacks are almost impossible to protect against. It is the responsibility of local administrators to prevent their users from initiating such attacks wherever possible.

3.1.8.3 TCP/IP spoofing

Most network resources are protected on the basis of the host IP addresses of those resources. Access is granted by a server to a client if the IP address is contained in an access control list (ACL). Since the operating system kernel itself declares its own identity when packets are sent, it has not been common to verify whether packets actually do arrive from the hosts which they claim to arrive from. Ordinary users have not traditionally had access to privileges which allow them to alter network protocols. Today everyone can run a PC with privileged access to the networking hardware.

Normally an IP datagram passing from host A to host B has a destination address ‘host B’ and source address ‘host A’. IP spoofing is the act of forging IP datagrams in such a way that they appear to come from a third party host, i.e. an attacker at host A creates a packet with destination address ‘host B’ and source address ‘host C’. The reasons for this are various. Sometimes an attacker wants to appear to be host C in order to gain access to a special resource which host C has privileged access to. Another reason might be to attack host C, as part of a more elaborate attack. Usually it is not quite this simple however, since the forgery is quickly detected. The TCP handshake is such that host A sends a packet to host B and then replies to the source address with a sequence number which has to match the next number of an agreed sequence. If another packet is not received with an agreed sequence number the connection will be reset and abandoned. Indeed, if host C received the confirmation reply for a message which it never sent, it would send a reset signal back immediately, saying effectively ‘I know nothing about this’. To prevent this from happening it is common to take out host C first by attacking it with some kind of Denial of Service method, or simply choosing an address which is not used by any host. This prevents it from sending a reset message. The advantage of choosing a real host C is that the blame for the attack is placed on host C.

SELF ASSESSMENT EXERCISES

1. Describe the nature of possible threats to the security of a human–computer system.
2. What is meant by ‘security is a property of systems’?
3. What are the four main themes in computer security?
4. What role does trust play in setting the ground rules for security?
5. Explain how security relates to risk assessment.
6. What are the main threats to human–computer security?

3.2 Security implementation

We looked at the meaning of security in the context of a computer system. Now we apply the basic principles and consider what practical steps can be taken to provide a basic level of security.

3.2.1 System design and normalization

Security is a property of systems; to address security, we must speak of the system as a whole:

- Identify what assets we are trying to protect.
- Evaluate the main sources of risk and where trust is placed.
- Work out possible counter-measures to attacks.

Counter-measures can be both preventative and reactive. They consist of:

- Rules
- Codified responses.

The foundation of security is policy. We must agree on what is valuable and acceptable in the system. Without such an assessment, we cannot speak of the risk to those assets, and determine what level of risk is acceptable. Policy is decided by social groups.

A system consists of an assembly of parts that exhibit three main activities:

- Input
- Rules
- Output.

Each of these must be addressed and poses different threats. Input exposes a part of the system to its outside environment. Output exposes the environment to the system part. The rules determine whether the part fulfills its role in the system. Security of a system requires the safe and predictable functioning of all parts within a system, and the safe and predictable functioning of the sum of those parts.

Protecting ourselves against threat also involves a limited number of themes:

- Applying safeguards (shields)
- Access control (selective shields)
- Protocols (specification of and limitation to safe behavior)
- Feedback regulation (continuous assessment)
- Redundancy (parallelism instead of serialism) detection and correction
- Monitoring the system
- Regulation.

We need to apply these to environments which utilize computer systems.
Normalization of a system is a concept from the theory of databases.

- Avoid unnecessary dependencies and inconsistencies.
- Validate assumptions.

3.2.2 The recovery plan

When devising a security scheme, think of the post-disaster scenario. When disaster strikes, how will the recovery proceed? How long is this likely to take? How much money or time will be lost as a result?

The network is a jigsaw puzzle in which every piece has its place and plays its part. Recall the principle of redundancy: the more dependent we are on one particular piece of the puzzle, the more fragile the set up. Recovery will occur more quickly if we have backups of all key hardware, software and data.

In formulating a recovery plan, then, we need a scheme for replacing key components either temporarily or permanently, and we should also bear in mind that we do rely on many things which are outside of our immediate control. What happens, for instance, if a digger (back-hoe) goes through the net cable, our only link to the outside world? Whom should we call? Less fundamental but more insidious, what if the network managers above us decide to decouple us from the network without informing us in advance? In a large organization, different people have responsibility for different maintenance tasks. It has happened on more than one occasion that the power has been shut down without warning – a potentially lethal act for a computer.

3.2.3 Data integrity and protection

As part of any infrastructure plan, we need to apply the principles of redundancy and protection to the system's data. Although backup copies will not protect us against loss, they do provide minimal insurance against accidents, intentional damage and natural disasters, and make the business of *recovery* less painful. There are several general strategies:

Encryption	Prevention of access on theft or tampering
Integrity checksums	Detection of error or tampering
Redundancy	Recovery from loss

3.2.3.2 Backup schemes

We can lose information in many ways: by accident, technical failure, natural disaster or even sabotage. We must make sure that there are several copies of the data so that everything may be recovered from a secure backup. Backups are one of the favorite topics of the system administration community. Everyone has their own local tricks. Many schemes for backup have been described; most of them resemble one another apart from cosmetic differences. Descriptions of backup schemes are manifold.

Backup applies to individual changes, to system setup and to user data alike. In backing up data according to a regular pattern, we are assuming that no major changes occur in the structure of data. If major changes occur, we need to start backups afresh. The network has completely changed the way we have to think about backup. Transmitting copies of files to secondary locations is now much simpler. The basics of backup are these:

- *Physical location:* A backup should be kept at a different physical location than the original. If data were lost because of fire or natural disaster, then copies will also be lost if they are stored nearby. On the other hand, they should not be too far away, or restoration time will suffer.
- *How often?:* How often do the data change significantly, i.e. how often do we need to make a backup? Every day? Do you need to archive several different versions of files, or just the latest version? The cost of making a backup is a relevant factor here.

- *Relevant and irrelevant files:* There is no longer much point in making a backup of parts of the operating system distribution itself. Today it is usually just as quick to reinstall the operating system from source, using the original CD-ROM. If we have followed the principle of separating local modifications from the system files, then it should be trivial to backup only the files which cannot be recovered from the CD-ROM, without having to backup everything.
- *Backup policy:* Some sites might have rules for defining what is regarded as valid information, i.e. what it is worth making a backup of. Files like prog.tar.gz might not need to be kept on backup media since they can be recovered from the network just as easily. Also one might not want to make backups of teen 'artwork' which certain users collect from the network, nor temporary data, such as browser cache files.

A backup schedule

How often we need to make backups depends on two competing rates of change:

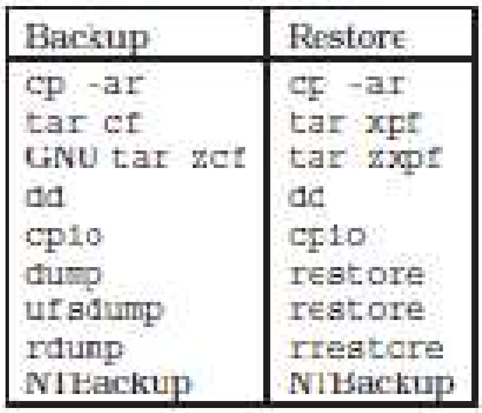
- The rate at which new data are produced.
- The expected rate of loss or failure.

For most sites, a daily backup is sufficient. In a war-zone, where risk of bombing is a threat at any moment, it might be necessary to back up more often. Most organizations do not produce huge amounts of data every day; there are limits to human creativity. However, other organizations, such as research laboratories collect data automatically from instruments which would be impractically expensive to re-acquire. In that case, the importance of backup would be even greater. Of course, there are limits to how often it is possible to make a backup. Backup is a resource-intensive process.

Suggestion (Static data). *When new data are acquired and do not change, they should be backed up to permanent write-once media at once. CD-ROM is an excellent medium for storing permanent data.*

For a single, un-networked host used only occasionally, the need for backup might be as little as once per week or less.

The options we have for creating backup schemes depend on the tools we have available for the job. On Windows we have NTBackup. On Unix-like systems there is a variety of tools which can be used to copy files and filesystems.



Backup	Restore
cp -ar	cp -ar
tar cf	tar xpf
GNU tar zcf	tar zxpf
dd	dd
cpio	cpio
dump	restore
ufsdump	restore
rdump	rrestore
NTBackup	NTBackup

Of course, commercial backup solutions exist for all operating systems, but they are often costly.

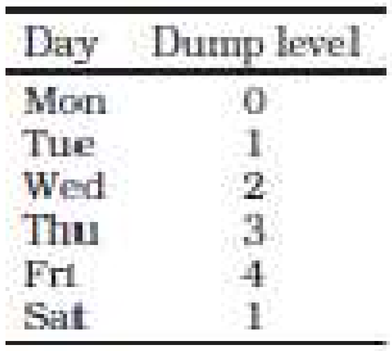
On both Unix and Windows, it is possible to backup filesystems either *fully* or *differentially*, also called *incrementally*. A full dump is a copy of every file. An incremental backup is a copy of only those files which have changed since the last backup was taken. Incremental backups rely on dump timestamps and a consistent and reliable system clock to avoid files being missed. For instance, the Unix dump utility records the dates of its dumps in a file /etc/dumpdates. Incremental dumps work on a scheme of levels, as we shall see in the examples below.

There are many schemes for performing system dumps:

• *Mirroring:* By far the simplest backup scheme is to mirror data on a daily basis. A tool like cfengine or rsync (Unix) can be used for this, copying only the files which have changed since the previous backup. Cfengine is capable of retaining the last two versions of a file, if disk space permits. A disadvantage with this approach is that it places the onus of keeping old versions of files on the user. Old versions will be mercilessly overwritten by new ones.

• *Simple tape backup:* Tape backups are made at different *levels*. A level 0 dump is a complete dump of a filesystem. A level 1 dump is a dump of only those files which have changed since the last level 0 dump; a level 2 dump backs up files which have changed since the last level 1 dump and so on, *incrementally*. There are commonly nine levels of dumps using the Unix dump commands. NTBackup also allows incremental dumps. The point of making incremental backups is that they allow us to capture changes in rapidly changing files without having to copy an entire filesystem every time. The vast majority of files on a filesystem do not change appreciably over the space of a few weeks, but the few files which we are working on specifically do change often. By pin-pointing these for special treatment we save both time and tapes.

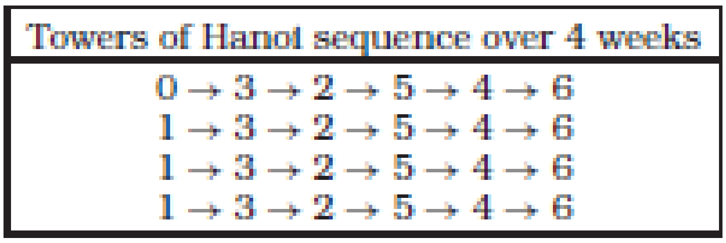
So how do we choose a backup scheme? There are many approaches, but the key principle to have in mind is that of *redundancy*. The more copies of a file we have, the less likely we are to lose the file. A dump sequence should always begin with a level 0 dump, i.e. the whole filesystem. This initializes the sequence of incremental dumps. Monday evening, Tuesday morning or Saturday are good days to make a level 0 dump, since that will capture most large changes to the filesystem that occur during the week or weekend, in the level 0 dump rather than in the subsequent incremental ones. Studies show that users download large amounts of data on Mondays (after the weekend break) and it stands to reason that after a week of work, large changes will have taken place by Saturday. So we can take our pick. Here is a simple backup sequence for user home-directories, then, assuming that the backups are taken at the end of each day:



Day	Dump level
Mon	0
Tue	1
Wed	2
Thu	3
Fri	4
Sat	1

Notice how this sequence works. We start with a full dump on Monday evening, collecting all files on the filesystem. Then on subsequent days we add only those files which have changed since the previous day. Finally on Saturday we go back to a level 1 dump which captures all the changes from the whole week (since the Monday dump) in one go. By doing this, we have two backups of the changes, not just one. If we do not expect much to happen over the weekend, we might want to drop the dump on Saturday.

A variation on this scheme, which captures several copies of every file over multiple tapes, is the so-called *Towers of Hanoi* sequence. The idea here is to switch the order of the dump levels every other day. This has the effect of capturing not only the files which have changed since the last dump, but also all of the files from the previous dump as well. Here is a sample for Monday to Saturday:



Towers of Hanoi sequence over 4 weeks
$0 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 6$
$1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 6$
$1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 6$
$1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \rightarrow 4 \rightarrow 6$

There are several things to notice here. First of all, we begin with a level 0 dump at the beginning of the month. This captures primarily all of the static files. Next we begin our first week with a level 3 dump which captures all changes since the level 0 dump. Then, instead of stepping up, we step down and capture all of the changes since the level 0 dump again (since 3 is higher than 2). This means that we get everything from the level 3 dump and all the changes since then too. On day 4 we go for a level 5 dump which captures everything since the last level 3, and so on. Each backup captures not only new changes, but the entire previous backup also.

This provides double the amount of redundancy as would be gained by a simple incremental sequence. When it comes to Monday again, we begin with a level 1 backup which grabs the changes from the whole of the previous week. Then once a month, a level 0 backup grabs the whole thing again.

The Towers of Hanoi sequence is clever and very secure, in the sense that it provides a high level of redundancy, but it is also expensive since it requires time and attention. Robotic automation can help here.

The level of redundancy which is appropriate for a given site has to be a question of economics based on four factors:

1. The cost of the backup (in time and media).
2. The expected rate of loss.
3. The rate of data production.
4. Media reliability.

These factors vary for different kinds of data, so the calculation needs to be thought out for each filesystem independently. The final point can hardly be emphasized enough. It helps us nothing to make ten copies of a file, if none of those copies are readable when we need them.

Suggestion (Tape backup). *Tapes are notoriously unreliable media, and tape streamers are mechanical nightmares, with complex moving parts which frequently go wrong. Verify the integrity of each substantial backup tape backup once you have made it. Never trust a tape. If the tape streamer gets serviced or repaired, check old tapes again afterwards. Head alignment changes can make old tapes unreadable.*

Needless to say, backups should be made when the system is virtually quiescent: at night, usually. The most obvious reason for this is that, if files are being changed while the backup is progressing, then data can be corrupted or backed up incorrectly. The other reason is one of load: traversing a filesystem is a highly disk-intensive operation. If the disk is being used extensively for other purposes at the same time, both backup and system will proceed at a snail's pace.

File separation

The principle of keeping independent files separate was not merely to satisfy any high-flying academic aesthetic, it also has a concrete practical advantage, particularly when it comes to backing up the system. There is little sense in backing up the static operating system distribution. It can be reinstalled just as quickly from the original CD-ROM (a non-perishable medium). However, operating system files that change often such as /etc/passwd or /etc/shadow which need to be at special locations within largely quiescent filesystems, can be copied to another filesystem which is backed up often. This follows automatically from our principle of keeping local changes separate from the OS files.

The same thing applies to other files like /etc/fstab or /etc/group and crontab which have been modified since the operating system was installed. However, here one can reverse the policy for the sake of a rational approach. While the password and shadow files have to be at a fixed place, so that they will be correctly modified when users change their passwords, none of the other files have to be kept in their operating system recommended locations.

Suggestion (OS configuration files). *Keep master versions of all configuration files like* /etc/fstab*,* /etc/group *or* crontabs/ *in a directory under site-dependent files, and us*[illegible] *l which synchronizes the contents of the master files with the operating system fi*[illegible]. *cfengine). This also allows the files to be distributed easily to other hosts which share a common configuration, and provides us with* one place *to make modifications, rather than having to hunt around the system for long-forgotten modifications. Site-dependent files should be on a partition which is backed up. Do not use symbolic links for synchronizing master files with the OS: only the root filesystem is mounted when the system boots, and cross-partition links will be invalid. You might render the system unbootable.*

3.2.3.3 Recovery from loss

The ability to recover from loss presupposes that we have enough pieces of the system from which to reconstruct it, should disaster strike. This is where the principle of redundancy comes in. If we have done an adequate job of backing up the system, including special information about its hardware configuration, then we will not lose data, but we can still lose valuable time.

Recovery plans can be useful provided they are not merely bureaucratic exercises. Usually a checklist is sufficient, provided the system administration team is all familiar with the details of the local configuration. A common mistake in a large organization, which is guaranteed to lead to friction, is to make unwarranted assumptions about a local department. Delegation can be a valuable strategy in the fight against time. If there are sufficient local system administrators who know the details of each part of the network, then it will take such people less time to make the appropriate decisions and implement the recovery plan. However, delegation also opens us up to the possibility of inconsistency – we must make sure that those we delegate to are well trained. (Remember to set the write-protect tab on tapes and have someone check this afterwards.)

When loss occurs, we have to recover files from the backups. One of the great advantages of a disk mirroring scheme is that users can find backups of their own files without having to involve an administrator. For larger file recoveries, it is more efficient for a system administrator to deal with the task. Restoring from tape backup is a much more involved task. Unfortunately, it is not merely a matter of donkey work. First of all we have to locate the correct tape (or tapes) which contain the appropriate versions of backed up files. This involves having a system for storage, reading labels and understanding any incremental sequence which was used to perform the dump. It is a time-consuming business. One of the awkwardnesses of incremental backups is that backing up files can involve changing several tapes to gather all of the files. Also, imagine what would happen if the tapes were not properly labeled, or if they are overwritten by accident.

Suggestion (URL filesystem names). *Use a global URL naming scheme for all filesystems, so that the filename contains the true location of the file, and you will never lose a file on a tape, even if the label falls off. Each file will be sufficiently labeled by its time-stamp and its name.*

We have two choices in recovery: reconstruction from backup or from source. Recovery from source is not an attractive option for local data. It would involve typing in every document from scratch. For software which is imported from external sources (CD-ROMs or ftp repositories), it is possible to reconstruct software repositories like /usr/local or Windows' software directories. Whether or not this is a realistic option depends on how much money one has to spend. 197 particularly impoverished department, reconstruction from source is a cheap option.

ACLs present an awkward problem for Windows filesystems. Whereas Unix's root account always has permission to change the ownership and access rights of a file, Windows's Administrator account does not. On Windows systems, it is important not to reinstate files with permissions intact if there is a risk of them belonging to a foreign domain. If we did that, the files would be unreadable to everyone, with no possibility of changing their permissions.

Data directory loss is one thing, but what if the system disk becomes corrupted? Then it might not even be possible to start the system. In that case it is necessary to boot from floppy disk, CD-ROM or network. For instance, a PC with GNU/Linux can be booted from a 'rescue disk' or boot disk, in single-user mode (see section 4.3.1), just by inserting a disk into the floppy drive. This will allow full access to the system disk by mounting it on a spare directory:

```
 mount /dev/hda1 /mnt
```

Recovery also involves some soul searching. We have to consider the reason for the loss of the data. Could the loss of data have been prevented? Could it be prevented at a later time? If the loss was due to a security breach or some other form of vandalism, then it is prudent to consider other security measures at the same time as we reconstruct the system from the pieces.

3.2.4 Authentication methods

Authentication methods are techniques for re-identifying users. They are based on matching attributes that uniquely identify individuals. Traditionally authentication has been based on shared secrets used in conjunction with cryptographic

There are two main approaches to the use of encryption: the use of symmetric encryption algorithms and the use of public key algorithms. Recently, related techniques such as smart cards (used in mobile phones) and biometrics (fingerprints and iris scans) have been experimented with.

3.2.4.1 Symmetric and asymmetric key methods

A shared secret identifies two parties to one another. With a symmetric key algorithm both parties must have explicit knowledge of the same secret key; one then has the problem of agreeing secrets with all of the individuals we want to talk to. If *N* parties need to communicate privately with a unique key, then one needs $N(N-1)/2$ secrets in total. Trust is established between each pair of individuals during the mutual agreement of the key. This is a simple and effective model, but its great overhead is the work required to establish and remember all of the keys.

With a public (or asymmetric) key algorithm, each party has two keys: a public key and a private key; thus there are $2N$ keys in total. The key-pair belonging to a given party consists of two related keys. A message that is encrypted with one of them can only be decrypted with the other. Each user can now keep one key completely secret and make the other key known to everyone. To send a secret message to the owner of the private key, someone only needs to encrypt a message with their public key. Only the owner of the matching private key can decrypt the message again (not even the person who encrypted it). This makes the problem of key distribution very straightforward. However, it has a price: since it obviates the need for a trusted meeting between the parties to agree on a secret, it makes the issue of trusting keys much harder. If you find a key, supposedly belonging to *X* on a particular web-site, you have only the word of the web-site owner that the key really is the key belonging to *X*. If you send a secret message to *X* using this key, it will only be readable by the owner of the private key that matches this key, but that could be anyone. Thus one has no idea, in general, whether or not to trust the identity associated with a public key. This issue is explored further below.

Public key algorithms are now widely used in authentication for their great convenience and flexibility.

3.2.5 Analyzing network security

In order to assess the potential risks to a site, we must gain some kind of overview of how the site works. We have to place ourselves in the role of an outsider: how would someone approach the network from outside? Then we have to consider the system from the viewpoint of an insider: how do local users approach the system? To begin the analysis, we form a list:

- What hosts exist on our site?
- What OS types are used?
- What services are running?
- What bug patches are installed?
- Run special tools, nmap, SATAN, SAINT, TITAN to automate the examination procedure and find obvious holes.
- Examine trust relationships between hosts.

This list is hardly a trivial undertaking. Simply building the list can be a lesson to many administrators. It is so easy to lose control over a computer network, so difficult to keep track of changes and the work of others in a team that one can easily find oneself surprised by the results of such a survey. Having made the list, it should become clear as to where potential security weaknesses lie. Network services are a common target for exploitation. FTP servers and Windows's commercial WWW servers have had a particularly hard time with bugs which have been exploited by attackers.

Correct host configuration is one of the prerequisites for network security. Even if we have a firewall shielding us from outside intrusion, an incorrectly configured host is a security risk. Firewalls do not protect us from the contents of data which are relayed to a host. If a bug can be exploited by sending a hidden message, then it will get through a firewall. Some form of automated configuration checking should be installed on hosts. Manual checking of hosts is impractical even with a single host; a site which has hundreds requires an automated procedure for integrity checking. On Unix and Windows one has cfengine and Perl for these tasks.

Trust relationships are amongst the hardest issues to debug. A trust relationship is an implicit *dependency*. Any host which relies on a network service implicitly trusts that service to be reliable and correct. This can be the cause of many stumbling blocks. The complexity of interactions between host services makes many trust relationships opaque. Trust relationships occur in any instance in which there is an external source of information: remote copying, hostname lookup, directory services etc. The most important trust relationship of all is the Domain Name Service (DNS). Many access control systems rely on an accurate identification of the host name. If the DNS service is compromised, hosts can be persuaded to do almost anything. For instance, access controls which assign special privileges to a name can be spoofed if the DNS lookups are corrupted or intercepted. DNS servers are therefore a very important pit-stop in a security analysis.

Access control is the fundamental requirement for security. Without access controls there can be no security. Access controls apply to files on a filesystem and to services provided by remote servers. Access should be provided on a need to- know basis. If we are too lax in our treatment of access rights, we can fall foul of intrusion. For example: a common error in the configuration of Unix file-servers is to grant arbitrary hosts the right to mount filesystems which contain the personal files of users. If one export filesystems which contain users' personal data to Unix-like hosts, it should be done on a host-by-host basis, with strict controls. If a user, who is root on their own host (e.g. a portable PC running GNU/Linux), can mount a user filesystem (with files belonging to a non-root user), that person owns the data there. The privileged account can read any file on a mounted filesystem by changing its user ID to whatever it likes. That means that anyone with a laptop could read any user's mail or change any user's files. This is a huge security problem. Hosts which are allowed to mount NFS filesystems containing users' private data should be secured and should be active at all times to prevent IP spoofing; otherwise it is trivial to gain access to a user's files.

There are many tools written for Unix-like operating systems which can check the security of a site, literally by trying every conceivable security exploit. Tools like SPY, COPS, SATAN, SAINT, TITAN, Nessus are aimed at Unix-like hosts. Port scanners such as nmap will detect services on any host with any operating system. These tools can be instrumental in finding problems. Recent and frightening statistics from the Computer Emergency Response Team indicated that only a pitiful number of sites actually upgrade or install patches and review their security, even after successful network intrusions.

Having mapped out an overview of a network site, and used the opportunity both to learn more about the specifics of the system, as well as fix any obvious flaws, we can turn our attention to more specific issues at the level of hosts.

3.2.5.1 Password security

Perhaps the most important issue for network security, beyond the realm of accidents, is the consistent use of strong passwords. Unix-like operating systems which allow remote logins from the network are particularly vulnerable to password attacks. The .rhosts and hosts.equiv files which allowed login without password challenge via rsh and rlogin were acceptable risks in bygone times, but these days one cannot afford to be lax about security. The problem with this mechanism is that .rhosts and hosts.equiv use hostnames as effective passwords. This mechanism trusts DNS name service lookups which can be spoofed in elaborate attacks. Moreover, if a cracker gets into one host, he/she will then be able to log in on every host in these files without a password. This greatly broadens the possibilities for effective attack. Typing a password is not such a hardship for users and there are alternative ways of performing remote execution for administrators, without giving up password protection (e.g. use of cfengine).

Password security is the first line of defence against intruders. Once a malicious user has gained access to an account, it is very much easier to exploit other weaknesses in security. Experience, indeed empirical evidence, shows that many users have little or no idea about the importance of using a good password. Consider some examples from a survey of passwords at a university. About 40 physicists had the password ‘Einstein’, around 10 had ‘Newton’ and several had ‘Kepler’. Hundreds of users used their login-name as their password; some of them really went to town and added ‘123’ to the end. Many girls chose ‘horse’ as their passwords. Even after extensive campaigns encouraging good passwords, users have a shocking tendency to trivialize this matter. User education is clearly an important weapon against weak passwords.

Some sites use schemes such as password aging in order to force users to change passwords regularly. This helps to combat password familiarity gained over time by local peer users, but it has an unfortunate side-effect. Users who tend to set poor passwords will not appreciate having to change their passwords repeatedly and will tend to rebel by setting trivial passwords if they can. Once a user has a good password, it is often advantageous to leave it alone. The problems of password aging are insignificant compared with the problem of weak passwords. Finding the correct balance of changing and leaving alone is a challenge.

Passwords are not visible to ordinary users, but their encrypted form is often visible. Even on Windows systems, where a binary file format is used, a freely available program like PwDump can be used to decode the binary format into ASCII. There are many publicly available programs which can guess passwords and compare them with the encrypted forms, e.g. crack, which is available both for Unix and for Windows. No one with an easy password is safe. Passwords should never be any word in a dictionary or a simple variation of such a word or name. It takes just a few seconds to guess these.

Modern operating systems have *shadow password files* or databases that are not readable by normal users. For instance, the Unix password file contains an 'x' instead of a password, and the encrypted password is kept in an unreadable file. This makes it much harder to scan the password file for weak passwords.

Tools for password cracking (e.g. Alec Muffet's crack program) can help administrators find weak passwords before crackers do. Other tools can be obtained from security sites to prevent users from typing in weak passwords.
.

3.2.5.2 Password sniffing

Many communication protocols (telnet, ftp etc.) were introduced before security was a concern amongst those on the Internet, so many of these protocols are very insecure. Passwords are often sent over the network as plain text. This means that a sophisticated cracker could find out passwords simply by listening to everything happening on the network and waiting for passwords to go by. If a cracker has privileged access to at least one machine with a network interface on the same network he/she can use tcpdump to capture all network traffic. Normal users do not have this privilege for precisely this reason. These days however, anyone with a laptop, an Ethernet card and a GNU/Linux installation could do this. Switched networks used to be immune to this problem since traffic is routed directly from host to host. However, now there exist tools that can poison the ARP cache and cause packets to be rerouted; thus switching is now only a low-level hindrance to password sniffing. In principle, any mildly determined user could do this.

Programs which dump all network traffic include tcpdump, etherfind, snoop and ethereal. Here is a sample of the output from Solaris' snoop program showing the Ethernet traffic from a segment of cable. Snoop recognizes common high-level protocols (SMTP/FTP/ARP etc.) and lists them explicitly. Unknown protocol types (in this case IPX) are simply listed as ETHER. In the right-hand column is the information which an intruder would try to use to sniff passwords.

One way to avoid the problem of password sniffing is to use fully encrypted links such as ssh and SSL (Secure Socket Layer) enabled services which replace the standard services. Another is to use a system of one-time passwords. One-time passwords are designed to eliminate the need for users to send their passwords over the network at all. Instead of typing an actual password, one types the remote password for a host into a program on a local machine, in order to generate a sequence of throw-away passwords which can be used in place of the actual remote password. The passwords are used only once so, even if someone gets to overhear them, it will already be too late: the password will have expired. Also the system is ingeniously designed so that the actual remote password (which is used to generate the one time passwords) never gets sent over the network at all. S/KEY is such a system. Here is an example of how it works:

1. We want to make a connection from host A to host B.
2. We have earlier set a password on host B.
3. We telnet to host B from host A.

4. Host B prompts us with a code string: 659 ta55095 and asks for our username. We type the username and host B asks for the one-time password.
5. We now need to find the one-time password by running a local program on host A with the code string as an argument:

```
Key 659 ta55095
passwd: *******
```

The key program prompts us for the secret password on host B. When we type this it does not go across the network. The key program returns a clear text, one-time password valid for one session: ‘EASE FREY WRY NUN ANTE POT’.
6. We type ‘EASE FREY WRY NUN ANTE POT’on host B (sent over the network) and the password is accepted.
7. Next time we follow the same procedure and get a different password.

3.2.5.4 Protecting against attacks

• Look out for users with weak or non-existent passwords. This is the easiest
for an attacker to enter the system.
• Train all staff in basic security procedures, and pay special attention to those who are highly privileged.
• Do not give trusted access to other hosts unless absolutely necessary.
Make sure there are no NIS wildcards + in /etc/hosts.equiv. Avoid using
.rhosts files altogether, and replace all of the old Berkeley ‘r’-commands (rlogin, rsh etc.) with a version of secure shell (ssh).4
• Attempts at initiating ping attack have been identified by large numbers of persistent ping processes.
• Disable unused services, e.g. in /etc/inetd.conf, which might contain security leaks, like UUCP, TFTP.
• Make sure that each active service runs in its own sandbox, with nonoverlapping privileges.
• Make sure the router filters all unnecessary traffic. Usually there is no reason to permit RPC or SNMP, NetBEUI, or NFS traffic outside of the local domain for instance. Anti-spoof filtering of IP addresses is also a must: e.g. a packet with a source address from a network on the other side of the planet cannot originate from inside the local network, so filter it.
• Make sure that the latest security patches are installed on all systems.
• Monitor connections using netstat -a to show all listening connections.
Use tcpd logging.
• Monitor processes running on the system. How many copies of important processes are running? How many should be running? Often it is possible to see that one is under attack by looking at what processes are running and who is running them. For instance an attempt at port sniffing or spamming might be seen with a bunch of processes like this:

```
Nobody  ....  /usr/sbin/inetd
```

```
nobody .... /usr/sbin/inetd
nobody .... /usr/sbin/inetd
nobody .... /usr/sbin/inetd
nobody .... /usr/sbin/inetd
```

inetd is a multiplexer which starts Internet services on many ports. Normally it is only root who runs this. The above indicates that a user is trying to use
the well-known account nobody to start services, or to overload the system with requests.

• Check filesystems for suspicious looking hidden files, i.e. files with names like
.. . These are often used to hide dangerous programs or shells which users can use to gain root privileges. Cfengine performs this task automatically when it examines filesystems.

• Check file integrity of static files and program code using MD5, SHA-1 or other checksums.

• Make sure that is not in root's path. It is possible to inadvertently execute a Trojan horse program.

• Make sure that log and audit files like /var/adm/utmp are not world writable, if possible, hence allowing crackers to cover their tracks. Modern Unices do not have this problem.

3.2.6 WWW security

The concept of World Wide Web (WWW) security sounds like a contradiction in terms. The WWW is designed to publish information to the masses. Security has to do with restricting access. What has the WWW got to do with security?

Web security has to do with:

• Protecting the published data from corruption.

• Granting access only to those files we wish to publish.

• Preventing users from tricking the WWW server into executing unauthorized commands on the server host.

• Protecting against a protocol that opens the system to all manner of attacks through abuses of the protocol.

Although there have been many security problems with the feature over-laden
Internet Information Server for Windows [300], there is nothing principally insecure about the WWW service. Any file-server can, in principle, compromise the security of a host by making information about that host available to others. If a server provides access to unauthorized files, this will clearly be the case. All we need to do is to ensure that proper access controls are maintained.

The Free Apache WWW server has all of the features one requires to operate a secure web service. It can be run without special privilege, and it has quite sophisticated mechanisms for restricting access to data. It is nevertheless possible to configure the server in an insecure fashion, so one needs to be cautious. There are three distinct categories for web use:

- External web service for organization.
- Internal web service for organization.
- Private users' web pages.

The last of these is arguably the greatest potential security risk for the Web: we usually trust the files and programs which we write ourselves in the name of our organization, but we have no reason to trust the integrity of private or guest users. There are two areas where a security breach can occur:

- File ownership and access rights.
- CGI-scripts.

CGI-scripts can be used to execute commands on the server-host with the user privileges of the WWW user. Although the WWW user is introduced precisely to isolate the powers of the WWW service, we can still do quite a bit of damage – not to the host directly, but to other users and to the web server access controls. It is an inevitable consequence of running a public service with a private ID that any file which gets written by a CGI-script can also be overwritten by another CGI-script, regardless of which user is responsible for that script. Thus users could wage waron one another with CGI-scripts such as guest-books, corrupting or even deleting one another's data freely. This is a fundamental weakness in the WWW service: if we allow the existence of arbitrary CGI-scripts on the system, then we can carry out arbitrary operations with the privileges of the WWW user. Users can:

- Send anonymous, untraceable mail which appears to come from the WWW user at the organization hosting the CGI program.
- Circumvent .htaccess access controls to certain files on most types of operating system, by executing the command /bin/cat filename as part of a CGI-script.

The first principle of server security is thus:

Principle (Service corruption). *If a server runs with the privileges of a non-privileged user, then none of the data or configuration files of the system should be owned by, or be writable by the* www *user, otherwise it is often trivial to alter the contents of the data using the service.*

3.2.7 Firewalls

A firewall is a network configuration which isolates some machines from the rest of the network. It is a gate-keeper which limits access to and from a network. Our human bodies are relatively immune to attack by bacteria and viruses because we have a barrier: skin. The skin contains layers of various fatty acids in which bacteria and viruses cannot normally survive. If we lose the skin from a part of the body, wounds become quickly infected; indeed, prior to antibiotics, many people died from infected wounds. A firewall is like a skin for a local area network.

The idea is this: if we could make a barrier between our local network and the Internet which is impenetrable, then we would be safe from network attacks. But if there is an impenetrable barrier so that no one can get into the network, then no one can get out either. Why pay for a firewall when we could just pull out the network cable? Think of the body again: we have to put food and air into our bodies and we have to let stuff out, so we need a hole in the skin (preferably several). We do not usually die of the food we eat because the body has filters which screen out and break down dangerous organisms (stomach acid and layers of mucus etc.). These then hand us the 'input' by proxy. We do the same thing with computer networks. A firewall is not an impenetrable barrier: it has holes in it with passport checks. We demand that only network data with appropriate credentials should be allowed to pass.

3.2.7.3 Intrusion detection and forensics

In the last few years the reality of network intrusion has led to several attempts to build systems which can detect break-ins, either while they are in progress or afterwards.

There are several ways in which we can gather evidence about intrusions. Evidence can be direct and indirect. Direct evidence might come from audits and log files, smoking guns, user observations, records of actions conducted by intruders, and so on. Checksums of important files can detect unauthorized changes, for instance. Indirect evidence can be obtained by looking at system activity and trying to infer unusual activity. Changes in the behavior of programs can signal changes in the patterns of usage of a system, perhaps flagging the exploit of vulnerability in software.

Intrusion detection by process monitoring is a relatively new idea. The idea is to gather a profile of what is normal and compare it with software behavior over time. This idea is a little like the idea of an immune system which tolerates 'self' and reacts against 'non-self'. Forrest et al. have pioneered system call profiling, inspired by vertebrate immune systems in order to detect hostile patterns of activity in special software processes. They build a database of short patterns of system call usage and then perform direct pattern search on subsequent data to detect anomalous patterns. The rationale for this approach is that intrusions are often caused by exploits of system calls which do not follow intended patterns. The beauty of this approach is its natural simplicity; its disadvantage is that it incurs a high overhead in resources to implement pattern searching in real-time; also the system needs to be taught what is normal in advance. Unfortunately 'normal' is a rather fickle concept [54], so in spite of its appealing simplicity, this is unlikely to be a complete, workable solution to the problem.

Another approach is to go to the network level and examine the totality of traffic arriving at a host. In order to detect an intrusion in progress, programs like *Network Flight Recorder* (NFR) and *Big Brother* (Bro) attempt to examine every packet on the network in order to look for tell-tale signatures of network break-in activity. This is an extremely resource-consuming task and it is beset with a number of problems. Few organizations have the resources to actually analyze the volumes of data they collect.

Network monitors look for packets containing data which might represent an attack, as they arrive. Network monitoring has its problems, however. One problem is that of *fragmentation*. Fragmentation is something which occurs to IP datagrams which pass between networks with different transmission rates. Larger packets can be broken up into smaller packets in order to optimize transmission. These fragments are reassembled at the final destination. This presents

a problem for intrusion detection systems because the fragmented packets might not contain enough data to identify them as hostile. This would allow them to get past the detection system. An intruder might be able to generate packets which were fragmented in such as way as to confound the attempts at detection. Another problem is that switches and routers limit the spread of traffic to specific cables. An intrusion detection system needs to see all packets in order to cover every attack. In spite of the difficulties, network intrusion detection is a hot research topic. A number of conferences on intrusion detection methods have sprung up to explore this problem in depth.
Network forensics is what one does after an intrusion. The idea is to examine logs and system audits in order to name the intruder and determine the damage. Network forensics is perhaps most important for the purpose of possible legal action against intruders. The cost of keeping the necessary logs and audits is very great and the work required after a break-in is far from trivial.

SELF ASSESSMENT EXERCISES

1. What elements should you have in a security and recovery plan?
2. Suggest some simple safeguards to protect inexperienced users from themselves.
3. What is meant by a file integrity check?
4. Explain the purpose of a firewall.
5. Explain the limitations of firewalls.
5. Explain the purpose of intrusion detection.
7. How would you deal with a host that you knew to be compromised by crackers?

4.0 CONCLUSION

We have discussed the various constituent parts of security, the basic principles and considered what practical steps to be taken to provide a basic level of security.

5.0 SUMMARY

Security is about protecting things of value to an organization, in relation to the possible risks. This includes material and intellectual assets; it includes the very assumptions that are the foundation of an organization or human–computer system. Anything that can cause a failure of those assumptions can result in loss, and must therefore be considered a threat.

Privacy and intrusion are two particular aspects of security, but the network is not our particular enemy. Many breaches of security happen from within, or by accident. If we focus exclusively on network connectivity we ignore the threats from internal employees (e.g. the janitor who is a computer expert and has an axe to grind, or the mischievous son of the director who was left waiting to play in mom’s office, or perhaps the unthinkable: a disgruntled employee who feels as though his/her talents go unappreciated).

There are many ways to attack a networked computer in order to gain access to it, or simply disable it. Some well-known examples are listed. The actual attack mechanisms used by attackers are often intricate and ingenious, but the common theme in all of them is to exploit naive limitations in the way network services are implemented.

Security is a property of systems; to address security, we must speak of the system as a whole: The foundation of security is policy. We must agree on what is valuable and acceptable in the system. Without such an assessment, we cannot speak of the risk to those assets, and determine what level of risk is acceptable. Policy is decided by social groups.

A firewall is a network configuration which isolates some machines from the rest of the network. It is a gate-keeper which limits access to and from a network. In the last few years the reality of network intrusion has led to several attempts to build systems which can detect break-ins, either while they are in progress or afterwards. There are several ways in which we can gather evidence about intrusions. Evidence can be direct and indirect.

Network forensics is what one does after an intrusion. The idea is to examine logs and system audits in order to name the intruder and determine the damage. Network forensics is perhaps most important for the purpose of possible legal action against intruders. The cost of keeping the necessary logs and audits is very great and the work required after a break-in is far from trivial.

6.0 TUTOR-MARKED ASSIGNMENTS

1. What are the basic requirements for computer security? Look around your network. Which hosts satisfy these basic requirements?

2. Devise a checklist for securing a PC attached to a network in your organization.
How would you secure a PC in a bank? Are there any differences in security requirement between your organization and a bank? If so, what are they and how do you justify them?

3. Determine what password format is used on your own system. Are shadow password files used?

4. Assume that passwords may consist of only the 26 letters of the alphabet. How many different passwords can be constructed if the number of characters in the password is 1, 2, 3, 4, 5, 6, 7 or 8 characters?

7.0 REFERENCES/FURTHER READING

1. Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2. Burke, J. R.(2004). Network Management Concepts and Practice: a Hands-on Approach. Pearson.

3. Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

4. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley.

5. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.

6. Subramanian, M. (2000). Network Management: Principles and Practice, Addison-Wesley.

UNIT 3: OUTLOOK AND THE FUTURE OF NETWORK ADMINISTRATION

1.0 INTRODUCTION

The aim of this course has been to present an overview of the field of network and system administration. For a long time, system administration has been passed on by word of mouth and has resisted formalization. Only in recent times has the need for a formalization of the field been acknowledged, through courses and certifications: determined, if not always ideal, attempts to crystallize something definite from the fluid and fickle body of knowledge which system administrators operate.

2.0 OBJECTIVES

At the end of this unit, you should be able to:

- Discuss on the outlook of computing
- Have an idea on the future of network and system administration

3.0 MAIN CONTENT

3.1 Outlook

System administration is about putting together a network of computers (workstations, PCs and supercomputers), getting them running and then *keeping* them running in spite of the activities of *users* who tend to cause the systems to fail. The failure of an operating system can be caused by one of several things. Most operating systems do not fail by themselves: it is users perturbing the system which causes problems to occur. Even in the cases where a problem can be attributed to a bug in a software component, it normally takes a user to provoke the bug. The fact that users play an important role in the behavior of computer systems is far from doubt. At universities, students rush to the terminal rooms to surf on the Web during lunch breaks. This can result in the sudden caching of hundreds of megabytes of temporary files which can prevent legitimate work from being carried out. In offices, the workers probably run from their desks giving the opposite pattern of behavior. The *time scale* involved here is just a matter of minutes, perhaps an hour. In that short space of time, user behavior (Web surfing) can cause a general failure of the system for all users (disk full). System administration is therefore a mixture of technical expertise and sociology. Patterns of user behavior need to be taken into account in any serious discussion of this problem. As a consequence, it is necessary to monitor the state of the system and its resources and react swiftly (on the time scale of human behavior) to correct problems.

3.2 Pervasive computing

In only a few years' time, computers will be everywhere in our lives. Embedded computers are already built into cars, kitchen appliances, media and telecommunications equipment; soon we will have smart walls with built in audiovisual equipment, smart buildings that can respond to those within it, cities wired with sensors that regulate traffic, and many other computing applications. Today many of these devices are running specialized operating systems, but increasingly Windows and Linux based kernels are being used. Such high-level operating systems have the advantage of being well known and highly adaptable, but they present the specter of complexity to tasks that are relatively simple. This presents a significant management challenge that will have to be addressed. Today, very few organizations face a challenge of this magnitude and very few technologies exist to cope with this level of complexity.

One clue about how this might be tackled lies in the Internet and the network of routers. Originally it was thought that routers might be controlled by a central authority; it did not take long to realize that the scale of the Internet (which rapidly outgrew expectations) was far greater than a centralized model could cope
with. Bottleneck designs, like central SNMP management, are quickly falling from favor. Routing succeeded because routers were made able to communicate and cooperate in building up their routing paths automatically, while satisfying broad criteria set by policy. This type of collaborative ordering is sometimes referred to as *swarm intelligence*, after the phenomenon observed in collaborating insects. This gives us a clue as to how the future of system administration is likely to evolve.

3.3 The future of system administration

We are approaching a new generation of operating systems, with the capacity for self-analysis and self-correction. It is no longer a question of whether they will arrive, but of when they will arrive. When it happens, the nature of system administration will change.

The day to day tasks of system administration change constantly and we pay these changes little attention. However, improvements in technology always lead to changing work practices, as humans are replaced by machinery in those jobs which are menial and repetitive. The core principles of system administration will remain the same, but the job description of the system manager will be rather different. In many ways, the day to day business of system administration consists of just a few recipes which slowly evolve over time. However, underneath the veneer of cookery, there is a depth of understanding about computer systems which has a more permanent value. Even when software systems take over many of the tasks which are now performed manually, there will be new challenges to meet.

For understandable reasons, the imaginations and attentions of our college generations have been captured, not by the intrigue of learning machines and intelligent systems, but by the glamour of multimedia. The computer has matured from a mere machine to a creative palette. It is difficult to articulate just why the administration of computer communities is an exciting challenge, but if we are to succeed in pushing through programmes of research which will bring about the level of automation we require, then it will be necessary to attract willing researchers. Fortunately, today there is a high proportion of system administrators with scientific backgrounds with the will and training to undertake such work. However, only the surface has been scratched. The tendency has been to produce tools rather than to investigate concepts, and while the tools are necessary, they must not become an end in themselves. A clearer understanding of the problems we face, looking forward, will only be achieved with more analytical work.

It is on this canvas that we attempt to congeal the discipline of system administration. We began this course material by asking whether system administration was indeed a discipline. I hope that it is now clear that it is – for a long time a diffuse one, but nevertheless real. In many ways system administration is like biology. Animals are machines, just billions of times more complex than our own creations, but the gap is closing and will continue to close as we enter into an era of quantum and biological computing techniques. The essence of experimental observation and of the complex phenomena and interrelationships between hosts is directly analogous to what one does in biology. We may have created computers, but that does not mean that we understand them implicitly. In our field, we are still watching the animals do their thing, trying to learn.

SELF ASSESSMENT EXERCISE

1. Now that we are done, compare your impressions of system administration with those you had at the end of module 1.

4.0 CONCLUSION

The day to day tasks of system administration change constantly and we pay these changes little attention. However, improvements in technology always lead to changing work practices, as humans are replaced by machinery in those jobs which are menial and repetitive. The core principles of system administration will remain the same, but the job description of the system manager will be rather different.

5.0 SUMMARY

The aim of this course has been to present an overview of the field of network and system administration. For a long time, system administration has been passed on by word of mouth and has resisted formalization.

System administration is about putting together a network of computers (workstations, PCs and supercomputers), getting them running and then *keeping* them running in spite of the activities of *users* who tend to cause the systems to fail.

We are approaching a new generation of operating systems, with the capacity for self-analysis and self-correction. It is no longer a question of whether they will arrive, but of when they will arrive. When it happens, the nature of system administration will change.

The day to day tasks of system administration change constantly and we pay these changes little attention. However, improvements in technology always lead to changing work practices, as humans are replaced by machinery in those jobs which are menial and repetitive. The core principles of system administration will remain the same, but the job description of the system manager will be rather different. In many ways, the day to day business of system administration consists of just a few recipes which slowly evolve over time.

It is on this canvas that we attempt to congeal the discipline of system administration. We began this course material by asking whether system administration was indeed a discipline. I hope that it is now clear that it is – for a long time a diffuse one, but nevertheless real.

6.0 TUTOR-MARKED ASSIGNMENTS

1. Discuss the future of network and system administration.
2. Write on what you see as the outlook for computing.

7.0 REFERENCES/FURTHER READING

1.Burgess, M. (2004). Principles of Network and System Administration. (2nd Ed.). Chichester, West Sussex , England: Wiley.

2.Forouzan, B.A, & Fegan, S.C. (2007). Data communications and Networking (4th Ed). Mc Graw Hill.

3. Limoncelli, T. A.,Hogan, C. J. & Chalup, S. R (2007}. The Practice of System and Network Administration. (2nd Ed.). Upper Saddle River, NJ: Addison-Wesley

4. Stallings, W. (2009). Data and computer communications (8th ed.). Upper saddle River, NJ.: Pearson Education Inc.