**MASENO UNIVERSITY**

**BACHELOR OF SCIENCE IN COMPUTER SCIENCE**

**YEAR 4 SEMESTER 1 SEPTEMBER - DECEMBER 2017**

**CCS 407: DISTRIBUTED SYSTEMS**

**NOTES**

**Fundamental concepts**

**Definition and characterization of distributed systems**

**Definition distributed systems**

There are many definitions:

[Coulouris]

* A distributed system is one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages.

[Tanenbaum & van Steen]

* A distributed system is a collection of independent computers that appears to its users as a single coherent system.

[Lamport]

* A distributed system is a system that prevents you from doing any work when a computer you have never heard about, fails.

A distributed system consists of a collection of autonomous computers linked by a computer network and equipped with distributed system software. This software enables computers to coordinate their activities and to share the resources of the system hardware, software, and data.

The above definitions take different perspectives

* Operational perspective
* User perspective
* DS characteristics perspective

**Practical examples of distributed systems**

* Web search
* Index the entire contents of the Web.
* Massively multiplayer online games
* Very large number of users sharing a virtual world.
* Financial trading
* Real time access and processing of a wide range of information sources.

**Distributed Systems application domains connected with networking:**

|  |  |
| --- | --- |
| **Application domains** | **Examples of distributed systems** |
| Finance and commerce | eCommerce e.g. Amazon and eBay, PayPal, online banking  and trading |
| The information society | Web information and search engines, ebooks, Wikipedia;  social networking: Facebook and MySpace |
| Creative industries and  entertainment | online gaming, music and film in the home, user-generated  content, e.g. YouTube, Flickr |
| Healthcare | health informatics, on online patient records,  monitoring patients |
| Education | e-learning, virtual learning environments; distance learning |
| Transport and logistics | GPS in route finding systems, map services: Google Maps,  Google Earth |
| Science | The Grid as an enabling technology for collaboration between scientists |
| Environmental management | sensor technology to monitor earthquakes, floods or  tsunamis |

**Implications of distributed systems**

1. Concurrency

* Components execute in concurrent processes that read and update shared resources. Requires coordination

1. No global clock

* makes coordination difficult (ordering of events)

1. Independent failure of components

* “partial failure” and incomplete information

1. Unreliable communication

* Loss of connection and messages. Message bit errors

1. Unsecure communication

* Possibility of unauthorised recording and modification of messages

1. Expensive communication

* Communication between computers usually has less bandwidth, longer latency, and costs more, than between independent processes on the same computer

**Characterization of distributed systems**

How to characterize a distributed system

* concurrency of components
* lack of global clock
* independent failures of components

**Motivation for distributed systems**

1. resource sharing

* the possibility of using available resources any where

1. openness

* an open distributed system can be extended and improved incrementally
* requires publication of component interfaces and standards protocols for accessing interfaces

1. scalability

* the ability to serve more users, provide acceptable response times with increased amount of data

1. fault tolerance

* maintain availability even when individual components fail

1. allow heterogeneity

* network and hardware, operating system, programming languages, implementations by different developers

**Resource sharing**

* The opportunity to use available hardware, software or data anywhere in the system
* Resource managers control access, offer a scheme for naming, and control concurrency
* A service is a software module that manages a collection of related resources and presents their functionality to users.
* A resource sharing model describes how
* resources are made available
* resources can be used
* service provider and user interact with each other

**Models for resource sharing**

* Client-server resource model
* Server processes act as resource managers, and offer services (collection of procedures)
* Client processes send requests to servers
* (HTTP defines a client-server resource model)
* Object-based resource model
* Any entity in a process is modeled as an object with a message based interface that provides access to its operations
* Any shared resource is modeled as an object
* Object based middlewares (CORBA, Java RMI) defines object-based resource models

**Scalability**

* A system is scalable if it remains effective when there is a significant increase in the amount of resources (data) and number of users
* Internet: number of users and services has grown enormously
* Scalability denotes the ability of a system to handle an increasing future load
* Requirements of scalability often leads to a distributed system architecture (several computers)

**Scalability problems**

* Often caused by centralized solutions

|  |  |
| --- | --- |
| **Concept** | **example** |
| Centralized services | A single server for all users |
| Centralized Data | A single on-line telephone book |
| Centralized Algorithms | Doing routine based on complete information |

* Characteristics of decentralized algorithms:
* No machine has complete information about the system state.
* Machines make decisions based only on local information.
* Failure of one machine does not ruin the algorithm.
* There is no implicit assumption that a global clock exists.

**Scaling techniques**

* **Distribution**
* splitting a resource (such as data) into smaller parts, and spreading the parts across the system (case of DNS)
* **Replication**
* replicate resources (services, data) across the system
* increases availability, helps to balance load
* caching (special form of replication)
* Hiding communication latencies
* avoid waiting for responses to remote service requests (use asynchronous communication or design to reduce the amount of remote requests)

**Fault tolerance**

* Hardware, software and network fail.
* DS must maintain availability even in cases where hardware/software/network have low reliability
* Failures in distributed systems are partial
* makes error handling particularly difficult
* Many techniques for handling failures
* Detecting failures (checksum )
* Masking failures (retransmission in protocols)
* Tolerating failures (as in web-browsers)
* Recovery from failures (roll back)
* Redundancy (replicate servers in failure-independent ways)

**Distribution transparency**

* An important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers
* A distributed system that is able to present itself to its users and applications as if it were only a single computer system is said to be transparent

**Transparency in a distributed system**

|  |  |
| --- | --- |
| **Transparency** | **Description** |
| Access | Hide differences in data representation and how a resource is accessed |
| Location | Hide where a resource is located |
| Migration | Hide that a resource may move to another location |
| Relocation | Hide that a resource may be moved to another location while in use |
| Replication | Hide that a resource is replicated |
| Concurrency | Hide that a resource may be shared by several competitive users |
| Failure | Hide the failure and recovery of a resource |

**Pitfalls when Developing Distributed Systems**

* False assumptions made by first time developer:
* The network is reliable.
* The network is secure.
* The network is homogeneous.
* The topology does not change.
* Latency is zero.
* Bandwidth is infinite.
* Transport cost is zero.
* There is one administrator.

**Quality of Service (QoS)**

* Non-functional properties of the system:
* Reliability
* Security
* Performance (Responsiveness and throughput)
* Adaptability to meet changes is an important aspect of QoS

**Types of distributed system**

* Distributed Computing Systems
* Used for high performance computing tasks
* Cluster and Cloud computing systems
* Grid computing systems
* Distributed Information Systems
* Systems mainly for management and integration of business functions
* Transaction processing systems
* Enterprise Application Integration
* Distributed Pervasive (or Ubiquitous) Systems
* Mobile and embedded systems
* Home systems
* Sensor networks

**Cluster Computing Systems**

* Collection of similar PCs, closely connected, all run same OS

Management application

Local OS

Parallel libs

Component of parallel application

Local OS

**An example of a cluster computing system**

Component of parallel application

Local OS

Component of parallel application

Local OS

Compute mode

Compute mode

Compute mode mode

Master mode

Remote process Network

High-speed Network

Standard Network

....

**Grid Computing Systems**

* Federation of autonomous and heterogeneous computer systems (HW,OS,...), several adm domains

Applications

Collective layer

Connectivity layer

Resource layer

Fabric layer

A layered architecture for grid computing systems

**Enterprise Application Integration**

* Allowing existing applications to directly exchange information using communication middleware

Client application

Client application

Communication Middleware

Server-side application

Middleware as a communication facilitator in enterprise application integration

Server-side application

Server-side application

**Example communication middleware: CORBA**

* Clients may invoke methods of remote objects without worrying about:
* object location, programming language, operating system platform, communication protocols or hardware.

IDL

IDL

IDL

Object Request Broker (ORB)

Common object model

RMI over IIOP

Different programming languages (or object models)

**Distributed Pervasive Systems**

* Pervasive systems is about exploiting the increasing integration of services and (small/tiny) computing devices in our everyday physical world
* (Mobile) Devices in distributed pervasive systems that discover the environment (its services) and establishes themselves in this environment as best as possible.
* Requirements for pervasive applications
* Embrace contextual changes.
* Encourage ad hoc and dynamic composition.
* Recognize sharing as the default.

**Distributed Computing as a Utility**

* View: Distributed resources as a commodity or utility
* Resources are provided by service suppliers and effectively rented rather than owned by the end user.
* The term **cloud computing** capture the vision of computing as a utility

**Challenges of distributed Systems**

* heterogeneity of their components
* openness
* security
* scalability – the ability to work well when the load or the number of users increases
* failure handling
* concurrency of components
* transparency
* providing quality of service

1. **Heterogeneity**

Heterogeneity – variety and difference in:

* Networks
* computer hardware
* OS
* programming languages
* implementations by different developers

**Middleware**

* middleware – software layer providing:
* programming abstraction
* masking heterogeneity of:
* underlying networks
* hardware
* operating systems

**Heterogeneity and mobile code**

* Mobile code – programming code that can be transferred from one computer to another and run at the destination (Example: think of Java applets)
* Virtual machine approach – way of making code executable on a variety of host computers – the compiler for a particular language generates code for a virtual ma-chine instead of a particular hardware order code.

1. **Openness**

OPENNESS of a:

* **computer system** - can the system be extended and reimplemented in various ways?
* **distributed system** - can new resource-sharing services be added and made available for use by variety of client programs?

An open system – What is the most important property to start with? key interfaces need to be published!

An open distributed system has:

* uniform communication mechanism
* published interfaces to shared resources

Open DS - heterogeneous hardware and software, possibly from different vendors, but conformance of each component to published standard must be tested and verified for the system to work correctly

1. **Security**
2. Confidentiality – protection against disclosure to unauthorized individuals
3. Integrity – protection against alteration or corruption
4. Availability – protection against interference with the means to access the re-sources

Security challenges not yet fully met:

* denial of service attacks
* security of mobile code

1. **Scalability**

**Scalability** is the ability to work well when the system load or the number of users increases

Challanges with building scalable distributed systems:

* Controlling the cost of physical resources
* Controlling the performance loss
* Preventing software resources running out (like 32-bit internet addresses, which are being replaced by 128 bits)
* Avoiding performance bottlenecks

**Example:** some web-pages accessed very frequently – remedy: caching and replication

1. **failure handling**

Techniques for dealing with failures

* Detecting failures
* Masking failures

1. messages can be retransmitted
2. disks can be replicated in a synchronous action

* Tolerating failures
* Recovery from failures
* Redundancy - redundant components

1. at least two different routes
2. like in DNS every name table replicated in at least two different servers
3. database can be replicated in several servers

**Main goal:** High availability (What is the measure of availability? ) – measure of the proportion of time that it is available for use

1. **concurrency of components**

* Example: Several clients trying to access shared resource at the same time
* Any object with shared resources in a DS must be responsible that it operates correctly in a concurrent environment

1. **transparency**

**Transparency** in the context of Distributed Systems – concealment from the user and the application programmer of the separation of components in a Distributed System for the system to be perceived as a whole rather than a collection of independent components

* **Acess transparency** – access to local and remote resources identical
* **Location transparency** – resources accessed without knowing their physical or network location
* **Concurrency transparency** – concurrent operation of pro-cesses using shared resources without interference between them
* **Replication transparency** – multiple instances seem like one
* **Failure transparency** – fault concealment
* **Mobility transparency** – movement of resources/clients within a system without affecting the operation of users or programs

1. **providing quality of service**

Main nonfunctional properties of systems that affect Quality of Service (QoS):

* reliability
* security
* performance

Time-critical data transfers

Additional property to meet changing system configuration and resource availability:

* adaptability

**Trends in distributed systems**

* emergence of pervasive networking technology
* emergence of ubiquitous computing coupled with the desire to support user mobility
* multimedia services
* distributed systems as utility

1. **Pervasive networking and the modern Internet**

* Networking has become a pervasive resource and devices can be connected at any time and any place

1. **Mobile and ubiquitous computing**

* laptop computers
* handheld devices (mobile phones, smart phones, tablets, GPS-enabled devices, PDAs, video and digital cameras)
* Wearable devices (smart watches, glasses, etc.)
* Devices embedded in appliances (washing machines, refrigerators, cars, etc.)

1. **Distributed multimedia systems**

* live or pre-ordered television broadcasts
* video-on-demand
* music libraries
* audio and video conferencing

1. **Distributed computing as a utility**

* **Cluster computing**
* **Grid computing**
* **Cloud computing**

Internet

Client

Storage services

Application services

Client

Client

Computational services

**Distributed System models**

* Represent the ways to describe Distributed systems
* There are three basic ways to describe Distributed systems

1. Physical models – consider DS in terms of hardware – computers and devices that constitute a system and their interconnectivity, without details of specific technologies
2. Architectural models – describe a system in terms of the computational and communication tasks performed by its computational elements. Client-server and peer-to-peer most commonly used
3. Fundamental models – take an abstract perspective in order to describe solutions to individual issues faced by most distributed systems

* interaction models
* failure models
* security models

1. **Physical models**

• Baseline physical model – minimal physical model of a distributed system as an extensible set of computer nodes interconnected by a computer network for the required passing of messages.

Three generations of distributed systems

1. Early distributed systems

* 10 and 100 nodes interconnected by a local area network
* limited Internet connectivity
* supported a small range of services e.g.
* shared local printers
* file servers
* email
* file transfer across the Internet

1. Internet-scale distributed systems

* extensible set of nodes interconnected by a network of networks (the Internet)

1. Contemporary DS with hundreds of thousands nodes + emergence of:

* mobile computing
* laptops or smart phones may move from location to location – need for added capabilities (service discovery; support for spontaneous interoperation)
* ubiquitous computing
* computers are embedded everywhere
* cloud computing
* pools of nodes that together provide a given service
* Distributed systems of systems (ultra-large-scale (ULS) distributed systems)

Significant challenges associated with contemporary DS:

Generations of distributed systems

|  |  |  |  |
| --- | --- | --- | --- |
| **Distributed System:** | **Early** | **Internet-Scale** | **Contemporary** |
| **Scale** | Small | Large | Ultra-large |
| **heterogeneity** | Limited(Typically relatively homogenous configuration) | Significant in terms of platforms, languages and middleware | Added dimensions introduced including radically different styles of architecture |
| **Openness** | Not a priority | Significant priority with range of standards introduced | Major research challenge with existing standards not yet able to embrace complex systems |
| **Quality of Service** | In its infancy | Significant priority with range of services introduced | Major research challenge with existing services not yet able to embrace complex systems |

1. **Architectural Models**

* Major concerns: make the system reliable, manageable, adaptable and cost-effective

1. **Architectural elements**

* What are the entities that are communicating in the distributed system?
* How do they communicate, or, more specifically, what communication paradigm is used?
* What (potentially changing) roles and responsibilities do they have in the overall architecture?
* How are they mapped on to the physical distributed infrastructure (what is their placement)?

1. **Communicating entities**

* From system perspective: **processes -** in some cases we can say that:
* nodes (sensors)
* threads (endpoints of communication)
* From programming perspective

– **objects**

* computation consists of a number of interacting objects representing natural units of decomposition for the given problem domain
* Objects are accessed via interfaces, with an associated interface definition language (or IDL)

**– components** – emerged due to some weaknesses with distributed objects

* offer problem-oriented abstractions for building distributed systems
* accessed through interfaces
* + assumptions to components/interfaces that must be present (i.e.making all dependencies explicit and providing a more complete contract for system construction.)

**– web services**

* closely related to objects and components
* intrinsically integrated into the World Wide Web
* using web standards to represent and discover services

1. **Communication paradigms**

There are three communication paradigms:

* interprocess communication
* remote invocation
* indirect communication

**Interprocess communication**

* low-level support for communication between processes in distributed systems, including message-passing primitives, direct access to the API offered by Internet protocols (socket programming) and support for multicast communication

**Remote invocation**

* calling of a remote operation, procedure or method
* **Request-reply protocols** – a pattern with message-passing service to support client-server computing
* Remote invocation has two variations:
* Remote procedure call (RPC)
* Remote method invocation (RMI)

**Remote procedure call (RPC)**

* procedures in processes on remote computers can be called as if they are procedures in the local address space
* supports client-server computing with servers offering a set of operations through a service interface and clients calling these operations directly as if they were available locally
* RPC systems offer (at a minimum) access and location transparency

**Remote method invocation (RMI)**

* strongly resemble RPC but in a world of distributed objects
* tighter integration into object-orientation framework

In RPC and RMI – senders-receivers of messages

* coexist at the same time
* are aware of each other’s identities

**Indirect communication**

* Senders do not need to know who they are sending to (space uncoupling)
* Senders and receivers do not need to exist at the same time (time uncoupling)

**Key techniques in indirect communication:**

Group communication

* **Publish-subscribe systems:**
* (sometimes also called distributed event-based systems)
* publishers distribute information items of interest (events) to a similarly large number of consumers (or subscribers)

• **Message queues:**

* (publish-subscribe systems offer a one-to-many style of communication), message queues offer a point-to-point service
* producer processes can send messages to a specified queue
* consumer processes can
* receive messages from the queue or
* be notified

• **Tuple spaces (also known as generative communication):**

* processes can place arbitrary items of structured data, called tuples, in a persistent tuple space
* other processes can either read or remove such tuples from the tuple space by specifying patterns of interest
* readers and writers do not need to exist at the same time (Since the tuple space is persistent)

•**Distributed shared memory (DSM):**

* abstraction for sharing data between processes that do not share physical memory

Communication entities and communication paradigms

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Communication Entities(What is communicating)** | | **Communication paradigms (how they communicate)** | | |
| System-oriented entities | Problem-oriented entities | Interprocess communication | Remote invocation | Indirect communication |
| Node | Objects | Message passing | Request-reply | Group communication |
| Process | Components | Sockets | RPC | Publish-subscribe |
|  | Web services | Multicast | RMI | Message queues |
|  |  |  |  | DSM |

1. **Roles and responsibilities**

* Client-server: Clients invoke individual servers

Invocation

Invocation

Result

Result

Computer:

Process:

Key::

* Peer-to-peer architecture
* same set of interfaces to each other

**Peer 3**

**Peers 4...N**

**Sharable objects**

**Peer 2**

**Peer 1**

1. **Placement**

* crucial in terms of determining the DS properties:

– performance

– reliability

– security

**Possible placement strategies:**

* mapping of services to multiple servers
* mapping distributed objects between servers, or
* replicating copies on several hosts
* more closely coupled multiple-servers – cluster

Service

* **caching**
* A cache is a store of recently used data objects that is closer to one client or a particular set of clients than the objects themselves
* mobile code
* Web Applets are an example of mobile code

1. Client request results in the downloading of applet code

Applet code

1. Client interact with applet

* In another possibility – **push model**: server initiates interaction (e.g. on information updates on it)
* **mobile agents**
* Mobile agent : running program (including both code and data) that travels from one computer to another in a network carrying out a task on someone’s behalf (e.g. collecting information), and eventually re-turning with the results.
* Mobile agent could be used for
* software maintenance
* collecting information from different vendors’ databases of prices
* Possible security threats with mobile code and mobile agents

1. **Architectural patterns**

**Layering**

* Layered approach – complex system partitioned into a number of layers:
* vertical organisation of services
* given layer making use of the services offered by the layer below
* software abstraction
* higher layers unaware of implementation details, or any other layers beneath them

**Platform and Middleware**

Software and hardware service layers in distributed systems

Applications, services

Middleware

Operating System

Computer and Network hardware

Platform

* A platform for distributed systems and applications consists of the lowest-level hardware and software layers.
* Middleware – a layer of software whose purpose is to mask heterogeneity and to provide a convenient programming model to application programmers.

**Tiered architecture**

* Tiering is a technique to organize functionality of a given layer and place this functionality into appropriate servers and, as a secondary consideration, on to physical nodes
* **Example: two-tier and three-tier architecture**
* functional decomposition of a given application, as follows:
* presentation logic
* application logic
* data logic

**Two-tier architecture**

Tier 2

Tier 1

Personal computers or mobile devices

Server

* Three aspects partitioned into two processes
* Advantage low latency
* Disadvantage splitting application logic

**Three-tier architecture**

Tier 1

**Personal computers or** mobile devices

**Application server**

**Database server**

Tier 3

Tier 2

* Three aspects partitioned into three processes
* Advantage one-to-one map-ping from logical elements to physical servers
* Disadvantage added complexity, network traffic and latency

**Thin clients**

* enabling access to sophisticated networked services (e.g. cloud services) with few assumptions to client device
* software layer that supports a window-based user interface (local) for executing remote application programs or accessing services on remote computer
* Concept led to Virtual Network Computing (VNC) – VNC clients accessing VNC servers using VNC protocol

Network

Computer Server

Network computer or PC

**Proxy pattern**

* designed to support location transparency in RPC or RMI
* proxy created in local address space, with same interface as the remote object

**Brokerage in web services**

* supporting interoperability in potentially complex distributed infrastructures
* service provider, service requestor and service broker
* brokerage reflected e.g. in registry in Java RMI and naming service in CORBA
* The web service architectural pattern

**Reflection pattern**

* a means of supporting both:
* introspection (the dynamic discovery of properties of the system)
* intercession (the ability to dynamically modify structure or behaviour)
* used e.g. in Java RMI for generic dispatching
* ability to intercept incoming messages or invocations
* dynamically discover interface offered by a given object
* discover and adapt the underlying architecture of the system

1. **Fundamental models**

* Fundamental models are categorized into:
* Interaction model
* Failure model
* Security model

1. **Interaction model**

* processes interact by passing messages –
* communication (information flow) and
* coordination (synchronization and ordering of activities) between processes
* communication takes place with delays of considerable duration
* accuracy with which independent processes can be coordinated is limited by these delays
* and by difficulty of maintaining the same notion of time across all the computers in a distributed system

Behaviour and state of DS can be described by a distributed algorithm:

* steps to be taken by each interacting process
* transmission of messages between them

State belonging to each process is completely private

**Performance of communication channels**

* **latency** – delay between the start of message’s transmission from one process and the beginning of receipt by another
* **bandwidth of a computer network** – the total amount of information that can be transmitted over it in a given time
* **Jitter** – the variation in the time taken to deliver a series of messages

**Computer clocks and timing events**

* **clock drift rate** – rate at which a computer clock deviates from a perfect reference clock

**Two variants of the interaction model**

**Synchronous distributed systems:**

* The time to execute each step of a process has known lower and upperbounds
* Each message transmitted over a channel is received within a known bounded time
* Each process has a local clock whose drift rate from real time has a known bound

**Asynchronous distributed systems:**

* No bounds on:
* Process execution speeds
* Message transmission delays
* Clock drift rates

**Event ordering**

Figure of Real-time ordering of events

1

A

Z

Y

X

2

4

t2

t3

t1

receive

m2

m1

m2

m3

Physical time

m1

send

receive

receive

receive

receive

receive

receive

receive

receive

send

send

3

• Logical time – based on event ordering

1. **Failure model**

* faults occur in:
* any of the computers (including software faults)
* or in the network
* Failure model defines and classifies the faults

**Omission failures**

* process or communication channel fails to perform actions it is supposed to do

**Process omission failures**

* chief omission failure of a process is to crash
* crash is called fail-stop if other processes can detect certainly that the process has crashed

**Communication omission failures**

* communication channel does not transport a message from p’s outgoing message buffer to q’s incoming message buffer – known as dropping messages
* send-omission failures
* receive-omission failures
* channel-omission failures

Figure showing Processes and channels

Process p

Process q

Incoming message buffer

Outgoing message buffer

Communication channel

**Arbitrary failures**

* arbitrary or Byzantine failure is used to describe the worst possible failure se-mantics, in which any type of error may occur

Figure showing Omission and arbitrary failures

|  |  |  |
| --- | --- | --- |
| **Class of failure** | **Effects** | **Description** |
| Fail-stop | Process | Process halts and remains halted. Other processes may detect this state. |
| Crash | Process | Process halts and remains halted. Other processes may not be able to detect this state. |
| Omission | Process | A message inserted in an the outgoing message buffer never arrives at the end’s incoming message buffer. |
| Send-omission | Process | A process completes a send, but the message is not put in its outgoing message buffer. |
| Receive-omission | Process | A message is put in a process’s incoming message buffer but , that process does not receive it. |
| Arbitrary (Byzantine) | Process or channel | Process/Channel exhibits arbitrary behaviour: it may send/transmit arbitrary messages at arbitrary times, commits omissions; a process may stop or take an incorrect step. |

Timing failures

* applicable in synchronous distributed systems

Figure showing Timing failures

|  |  |  |
| --- | --- | --- |
| **Class of failure** | **Effects** | **Description** |
| Clock | Process | Process’s local clock exceeds the bounds on its rate of drift from real time. |
| Performance | Process | Process exceeds the bounds on the interval between two steps. |
| Performance | Channel | A message’s transmission takes longer than the stated bound. |

**Masking failures**

* knowledge of the failure can enable a new service to be designed to mask the failure of the components on which it depends

**Reliability of one-to-one communication**

* reliable communication:
* **Validity:** What is validity? Any message in the outgoing message buffer is eventually delivered to the incoming message buffer
* **Integrity:** What is integrity? The message received is identical to one sent, and no messages are delivered twice

1. **Security model**

* modular nature of distributed systems and their openness exposes them to attack by
* both external and internal agents
* Security model defines and classifies attack forms,
* providing a basis for the analysis of threats
* basis for design of systems that are able to resist them
* the security of a distributed system can be achieved by securing the processes and the channels used for their interactions and by protecting the objects that they encapsulate against unauthorized access.

**Protecting objects**

* Users with access rights
* association of each invocation and each result with the authority on which it is issued
* such an authority is called a principal
* principal may be a user or a process

Figure showing Objects and principals

Server

Access rights

Network

Principal (user)

Principal (server)

object

result

invocation

**Securing processes and their interactions**

* securing communications over open channels
* open service interfaces

**Communication channel**

**m**

**m’**

**Copy of m**

**Threats to processes**

* lack of knowledge of true source of a message
* problem both to server and client side
* example: spoofing a mail server

**Threats to communication channels**

* threat to the privacy and integrity of messages
* can be defeated using secure channels

**Defeating security threats**

**Cryptography and shared secrets**

* Cryptography is the science of keeping messages secure
* Encryption is the process of scrambling a message in such a way as to hide its contents

**Authentication**

* based on shared secrets authentication of messages – proving the identities sup-plied by their senders

**Secure channels**

Figure showing Secure channels

Principal A

Principal A

Secure channel

**Properties of a secure channel:**

* Each of the processes knows reliably the identity of the principal on whose behalf the other process is executing
* A secure channel ensures the privacy and integrity (protection against tampering) of the data transmitted across it
* Each message includes a physical or logical timestamp to prevent messages from being replayed or reordered

**Other possible threats from an enemy**

**Denial of service:**

* the enemy interferes with the activities of authorized users by making ex-cessive and pointless invocations on services or message transmissions in a network, resulting in overloading of physical resources (network band-width, server processing capacity)

**Mobile code:**

* execution of program code from elsewhere, such as the email attachment etc.

**The uses of security models**

Security analysis involves

* the construction of a threat model:
* listing all the forms of attack to which the system is exposed
* an evaluation of the risks and consequences of each

**Networking and internetworking and communication protocol**

* Distributed systems use local area networks, wide area networks and internetworks for communication
* Changes in user requirements have resulted in the emergence of wireless networks and of high-performance networks with QoS guarantees

1. **Introduction**

|  |  |  |
| --- | --- | --- |
| **Transmission media** | **Hardware devices** | **Software devices** |
| * Wire | * Routers | * Protocol stack |
| * Cable | * Switched | * Communication handlers |
| * Fibre | * Bridges | * Drivers |
| * Wireless channels | * Hubs |  |
|  | * Repeaters |  |

* **communication subsystem** – hardware and software components that provide the communication facilities for a distributed system
* **hosts** – computers and other devices that use the network for communication purposes
* **node** – any computer or switching device attached to a network.
* **subnet**
* unit of routing (delivering data from one part of the Internet to another);
* collection of nodes that can all be reached on the same physical network

**Networking issues for distributed systems**

**Performance**

* Message transmission time = latency + length ⁄ data transfer rate
* longer messages segmented – transmission time is the sum of transferring the segments
* total system bandwidth of a network is a measure of throughput – the total volume of traffic that can be transferred across the network in a given time
* in most wide area networks messages can be transferred on several different channels simultaneously
* The performance of networks deteriorates in conditions of overload
* time required to access shared resources on a local network remains about a 1000x greater than that required to access resources that are resident in local memory

**Scalability**

* Difficult to estimate the real size nowadays
* The potential future size of the Internet
* will be of order of the population of the planet
* several billion nodes and hundreds of millions of active hosts
* Many applications are able to recover from communication failures and hence do not require guaranteed error-free communication
* usually errors due to
* software errors in sender or receiver
* (for example, failure by the receiving computer to accept a packet)
* buffer overflow
* network errors (not that often though)

**Security**

* firewall technology
* cryptographic technology
* virtual private network (VPN) techniques

**Mobility**

* The Internet’s mechanisms have been adapted and extended to support mobility, but the expected future growth in the use of mobile devices will demand further development

**Quality of service**

* multimedia data transmission

**Multicasting**

* simultaneous transmission of messages to several recipients

**TYPES OF NETWORK**

* Types of networks are confusing because they seem to refer to the physical extent (local area, wide area), but they also identify physical transmission technologies and low-level protocols

Figure showing Network performance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Example** | **Range** | **Bandwidth(Mbps)** | **Latency(ms)** |
| **Wired** |  |  |  |  |
| LAN | Ethernet | 1 – 2km | 10 – 10,000 | 1 - 10 |
| WAN | IP routing | wide | 0.010 - 600 | 100 - 500 |
| MAN | ATM | 2 – 50km | 1 - 600 | 10 |
| Internetwork | Internet | Worldwide | 0.5 - 600 | 100 - 500 |
|  |  |  |  |  |
| **Wireless** |  |  |  |  |
| WPAN | Bluetooth(IEEE 802.15.1) | 10 – 30m | 0.5 - 2 | 5 - 20 |
| WLAN | WiFi(IEEE 802.11) | 0.15 – 1.5km | 11 - 108 | 5 - 20 |
| WMAN | WiMAX(IEEE 802.16) | 5 – 50km | 1.5 - 20 | 5- 20 |
| WWAN | 3G phone | Cell: 1 - 5 | 3.48 – 14.4 | 100 - 500 |

**Network principles**

Basis – packet switching technique (developed in 1960s)

* packets addressed to different destinations to share a single communications link
* Packets are queued in a buffer and transmitted when the link is available
* Communication is asynchronous –messages arrive with a delay depending upon properties and utilization of the network

**Packet transmission**

* messages – sequences of data items of arbitrary length
* subdivided into packets
* packets have a restricted length

**Data streaming**

Streaming is the transmission and display of audio and video in real time

* video stream requires
* 1.5 Mbps if data is compressed
* 120 Mbps if data is uncompressed
* channel from source to destination of a multimedia stream
* predefined route
* reserved set of resources
* buffering where appropriate for smoothness

**Switching schemes**

Broadcast

* involves no switching
* some LAN technologies (including Ethernet) based on broadcasting
* wireless networking with nodes grouped in cells

**Circuit switching**

* plain old telephone system (or POTS) – typical switching network

**Packet switching**

* store-and-forward network

**Frame relay**

* switching small packets called frames on the fly
* switching nodes route frames based on the examination of their first few bits
* frames as a whole are not stored at nodes but pass through them as short streams of bits

**Protocols**

A protocol is well-known set of rules and formats to be used for communication between processes in order to perform a given task. A protocol plays two important tasks:

1. a specification of the sequence of messages that must be exchanged
2. a specification of the format of the data in the messages

* The existence of well-known protocols enables the separate software components of distributed systems to be developed independently and implemented in different programming languages on computers that may have different order codes and data representations
* A protocol is implemented by a pair of software modules located in the sending and receiving computers.

**Protocol layers**

Network software arranged in

* hierarchy of layers
* each layer presents an interface (service) to the layer(s) above it

**Protocol suites**

* protocol suite (or protocol stack) is a complete set of protocol layers
* Seven-layer Reference Model for Open Systems Interconnection (OSI)

Figure showing Protocol layers in the ISO Open Systems Interconnection (OSI) model

Application

Presentation

Session

Transport

Network

Data link

physical

Recipient

Communication medium

Sender

Message received

Message sent

Layers

**Figure showing OSI protocol summary**

|  |  |  |
| --- | --- | --- |
| **Layer** | **Description** | **Examples** |
| Application | Protocols that are designed to meet communication requirements of specific applications, often defining the interface to a service | HTTP, FTP, SMTP, CORBA IIOP |
| Presentation | Protocols at this level transmit data in a network representation that is independent of the representations used in individual computers which may differ. Encryption is also performed in this layer if required | Secure sockets (SSL), CORBA data Representation |
| Session | At this level, reliability and adaptation are performed such as detection of failures and automatic recovery |  |
| Transport | This is the lowest level at which messages(rather than packets) are handled. Messages are addressed to communication ports attached to processes. Protocols in this layer may be connection-oriented or connectionless | TCP. UDP |
| Network | Transfers data packets between computers in a specific network. In a WAN or an internetwork, this involves the generation of a route passing through routers. In a single LAN, no routing is required | IP, ATM virtual circuits |
| Data link | Responsible for the transmission of packets between nodes that are directly connected by a physical link. In a WAN transmission is between pairs of routers and hosts. In a LAN it is between any pair of hosts | Ethernet MAC, ATM cell transfer, PPP |
| Physical | The circuits and hardware that drive the network. It transmits sequences of binary data by analog signaling using amplitude or frequency modulation of electrical signals(on cable circuits), light signals(on fibre optic circuits) or any other electromagnetic signals(on radio and microwave circuits). | Ethernet Base-band signaling, ISDN |