Notes for exam 2021

by Sondre Lindaas Gjesdal

Introduction

Distributed systems

Definition 1: Collection of autonomous computing elements that appears to its users as a single coherent system

Definition 2: One in which hardware or software components located at networked computers communicate and coordinate their actions only by passing images

Characteristics of distributed systems

- **Collection of autonomous nodes**
  - Nodes can act independently of each other, but programmed to achieve common goals through exchange of messages
    - No concept of global clock (need to resolve synchronization and coordination issues)
  - Group membership
    - Open or closed group
    - Admission control
    - Managing authentication of nodes
  - Organization of nodes (e.g. P2P structures)
    - Structured overlay network
      - Tree or ring structures
    - Unstructured overlay network
      - Node has a number of references to randomly selected other nodes

BitTorrent Peer-to-Peer distributed file sharing

Cluster vs. Grid computing

Idea: build a supercomputer from a collection of single computers for compute intensive and parallel computations

Cluster computing

- Homogeneous: Same OS, near/identical hardware
- Usually connected via LAN
- Single managing code

Grid computing

- Heterogeneous Network and Nodes
- Dispersed across several organizations
- Can easily span a wide-area network

Cloud computing

Cloud computing is a form of utility computing where resources are provisioned and paid on a need basis

Distributed information systems

Enterprise application

e.g. Distributed Database application

- Banking systems
- Hospital information
- Airline management systems

Distributed transactions. Transactions are executed in a "all" or "none" fashion.

Characteristics of distributed systems

Single coherent systems

The collection of nodes as a whole operates the same, no matter where, when, and how interaction between a user and the system takes place.

Examples:

- And end user cannot tell where a computation is taking place
- Where data is exactly stored should be irrelevant to an application
- If or not data has been replicated is completely hidden

Keyword is distribution transparency

Downside: Partial failures It is inevitable that at any time only a part of the distributed system fails. Hiding partial failures and their recovery is often very difficult and in general impossible to hide

Middleware: The OS of distributed systems

- Intended to hide heterogeinity and provide useful abstractions for application programmers
- Contains commonly used components and functions that need not be implemented by application seperately
- Provides services such as:
  - Resource management
  - Provides facility for interapplication communication
    - Remote Procedure Call(RPC)
    - Messages Oriented MiddleWare(MOM)
  - Masking of and recovery from failures
  - Security services

Goals of distributed systems

- Resource sharing
- Transparency

- Openness
- Scalability

Design goals

Support resource sharing

Goal:

- Access and share remote resources(e.g.)
  - Computing resources - e.g. Storage
  - File-sharing
  - Groupware: Collaborative editing and teleconferencing, discussion forums, etc.

Making distribution transparent

- How to achieve the single-system image, i.e., How to make a collection of computers appear as a single computer
  - Access transparency
    - Hide differences in data representation and how an object is accessed
  - Location transparency
    - Hide where an object is located
  - Relocation transparency
    - Hide that object may be moved to another location while in use
  - Migration transparency
    - Hide movement of object to another location
  - Replication transparency
    - Hide replication of object
  - Concurrency transparency
    - Hide the sharing of object by several independent users
  - Failure transparency
    - Hide the failure & recovery of an object

Openness

- Interoperability
  - Extent by which two implementations of systems or components from different manufacturers can co-exist and work together using a specified common standard
- Portability
  - Extent an application developed for distributed system A can be executed, without modification, on a different distributed system B that implements the same interfaces as A.

Scalability

- Size scalability

  - Adding size and resources without noticeable performance loss

- Geographical scalability

- Users and resources may lie far apart, communication delay is not noticed by users

- Administrative scalability

    - Easily managed even if it spans many independent administrative organizations

- Bottlenecks

    - Computational capacity, limited by the CPUs
    - Storage capacity, including the I/O transfer rate
    - Network between the user and the centralized service

- A surge in network traffic to the database server

- Performance can degrade as a result (slow network)

- Availability can be affected (systen can hang due to overload)

- Single point of failure - the database machine can crash

Solutions for bottlenecks:

Solution 1 - Upgrade the database hardware (Vertical scaling). *Limitation:* Performance can only increase up to the latest hardware capabilities

Solution 2 - Distribute the database server (Horizontal scaling/Replication). *Fault-tolerance:* If one node fails, users are not affected as other running nodes automatically handle the requests. *Low latency:* Users are directed to geographically closer node thereby reducing the packets round-trip time.

Topics

- **Communication paradigms in DS**
- **Coordination and Synchronization**
- **Naming**
- **Replication and Consistency**
- **Fault-tolerance**
- **Security**

Metrics layering

Internet - Network model

- A network of networks supporting applications exchanging messages
    - Internal structure is invisible to the applications
- Hosts - end systems
    - PCs, servers, mobile devices, internet-things and so on
    - Runs networked applications/software
- Packet switches
    - Routers and link-layer switches
    - Forward packets between hosts and switches
- Communication Links

- Copper, optical fiber, radiowaves
- Connects hosts and packet switches

Packet switching

- Internet is based on packet switching
- Packet switching operation
  - Receive packets on incoming communication links
  - Consult forwarding table to determine outgoing link
  - Buffer (queue) packets for transmission on outgoing link
  - Transmit packet on the outgoing communication link
- The store-and-forward operation results in
  - **Processing delays:** checking packet integrity and forwarding table lookup
  - **Queuing delays:** packets waiting for transmission on the communication link
  - **Packet drop/loss:** Due to buffer overflow

Transmission and propagation delay

- A communication link can be characterised by **transmission rate** R and **propagation speed** S
- Transmission rate R
  - bits per second (bps) that can be put on the link
  - Typically ranges from 10 Mbps - 100 Gbps
  - Determines the **Transmission delay Dtrans = L/R** for a packet of L bits
- Propagation speed S
  - The speed with which a bit/signal moves along the link
  - Typically ranges between 2 * 10^8 m/s to 3 * 10^8 m/s
  - Determines the propagation delay Dprop = d/S for a link distance d

Node and end-to-end delay

- Nodal delay is the total delay experienced by a data packet at a node (packet switch or host)
- Nodal delay Dnodal at a packet switch: Dnodal = Dproc + Dqueue + Dtrans + Dprop
- End to end delay is the total delay experienced by a packet from source to destination

Traffic intensity and delay

- Traffic intensity I relates the capacity of outgoing links with the arrival rate of incoming packets
  - Tranmission rate **R bits/second** on outgoing link
  - packet size of **L bits**
  - Average arrival rate of **a packets per second**
- Intensity = (packet size * average arrival rate) / Transmission rate
- I < 1
  - Queue size will be bounded
- I > 1
  - Queue size grows without bound
- Network congestion arises as traffic intensity approaches 1

Throughput

- Throughput measures the rate at which data is being received at the destination
- **Instantaneous throughput:** current amount of data received measured in bits/second
- **Average throughput:** average amount of data received per time unit for entire data transfer

Throughput and bottleneck links

- The transmission rate of links between sender and receiver gives an upper bound on the throughput
- The **bottleneck link** is the link that prevents an increase in throughput

Simultaneous transmission

- Senders and receivers are competing for the available bandwidth in the internet
  - Multiple data traffic flow competing for bandwidth/capacity
  - Throughput and delay affected by the data traffic between other senders and receivers
  - Throughput and delay may also be affected by multiple application on hosts
  - Bottleneck link is usually in the access network

Protocol layering and stacks

- Huge gap between
  - **Physical transmission of bits** over a single **communication link** connecting a host to a packet switch (router) or connecting two packet switches; to
  - **Exchange of messages** between **application running on hosts** connected to a network of networks connected by packet switches
- Protocols, software and hardware in the Internet is oganised in a **stack** with five layers
  - **Application**
    - User space
  - **Transport:** Reliable and unreliable end-to-end transmission of *segments*
    - OS - kernel space
  - **Network:** Transmission of *datagrams* between nodes over multiple networks
    - OS - kernel space
  - **Link:** Transmission of *frames* over a communication link/network
    - Hardware - network interface card
  - **Physical:** Transmission of *bits* over a communication link/network
    - Hardware - network interface card

Transmission on links

- An *unreliable* stream of bits between sender and receiver
- **Transmission errors** (0->1,1->0) cause the bits received to be different from the bits sent

Distributed applicaiton

- Rely on the *exchange of messages* and the use of high-level programming primitives and APIs
- Example is simple HTTP client application

Layering

- Divides complexity into smaller and manageable problems

- Each layer provides a well-defined *service* to the upper layer
- A service provided by a layer is implemented using *protocols* relying on the service provided by the lower layer
- Implementation of a layer can be modified without affecting the upper layers - as long the same service is provided

Summary for metrics layer

- Internet is based on packet switching which may give rise to queuing delays and packet loss
- Sources of delay at a node - hosts or packet switches
    - Processing time and size of packet queues at a node
    - Transmission delay: depends on the packet length and the transmission rate of outgoing communication link
    - Link propagation delay: depends on signal propagation speed and the length of the communication link
- Size of queues start to increase as the traffic intensity on outgoing links approach 1
- Instantaneous and average throughput measures the rate at which data is being received at the destination
    - Bottleneck links are links that prevent an increase in throughput
- Queing visualisation: https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/queuing-loss-applet/index.html
- Link delay visualisation: https://media.pearsoncmg.com/aw/ecs_kurose_compnetwork_7/cw/content/interactiveanimations/transmission-vs-propogation-delay/transmission-propagation-delay-ch1/index.html
- Protocols, hardware, and software are organised into layers to bridge the large gap between:
    - Application requirements: exchange of messages(data)
    - The unreliable transmission of bits on the physical links
- The TCP/IP protocol stack comprises five layers
    - Physical(1), link(2), network(3), transport(4), and application(5)
    - Hosts implement 1-5
    - Routers implements 1-3
- Each layer provides a service (the what) to the upper layer and is implemented by a protocols (the how) at that layer
- This separates the network interconnection service from the complexity of applications and provides flexibility

Layering and applications

TCP/IP protocol stack - layering

- Protocols, hardware, and software organised into five layers
    - Each layer provides a well-defined service for the upper layer
    - A service provided by a layer is implemented using protocols relying on the service provided by the lower layer

Physical layer

- Transmission of bits across a single communication link
- **protocol:** map bits into signals on the physical medium
- **Service provided:** transmission of an unreliable bit-stream
    - Transmission errors may cause bits to be flipped

Link layer

- Transmission of frames on a single communication link
- May involve medium access control protocols and addressing
- Service, frame format, and addressing of sender and receiver depend on the concrete link layer technology

Network layer

- Main purpose is to create a network of networks
    - Unreliable end-to-end(host-to-host) transmission of datagrams(loss, duplication, and overtaking is possible)
    - Multihop routing of datagrams based on local routing/forwarding table
- Service implemented by the Internet Protocol(IPv4/v6)
    - Addressing using
        - 32-bit IPv4 addresses
            - 158.37.70.86
        - 128-bit IPv6 addresses
            - `fe80::c80:dda1:5a68:1693::60f8:1dcb:131e`

Transport layer

- Service provided: end-to-end transport of segments between applications(processes)
- Addressing based on IP addresses and a 16-bit port number

Transmission Control Protocol (TCP)

- Reliable transport service
    - Connection-oriented
    - Bidirectional reliable byte-stream between applications

User Datagram Protocol(UDP)

- Unreliable transport service
    - Connection-less
    - Unreliable transfer of segments
    - Loss, Overtaking, and duplication

Transport services

- The choice of transport service depends upon the particular application and its requirements
    - **Reliability:** can occasional data loss be tolerated?
    - **Throughput:** sensitivity to variations in throughput?
    - **Timing:** requirement to the end-to-end delay?

- **Security:** is the data confidential?

Process communication

- Processes access the transport service using a socket API provided via the OS
  - Processes creates and closes sockets via the API
  - Processes send and receive messages via sockets

Application layer

- Exchange of messages between processes running on hosts - any software that runs on top of the transport services
- Important application layer protocols
  - Remote login to hosts - telnet/SSH
  - File Transport protocol(FTP)
  - Electronic mail transport protocols(SMTP)
  - Domain Name System(DNS)
  - Simple network Management protocol(SMTP)
  - Hypertext Transport protocol(HTTP)

Headers and payload

- The packets exchanges at a given layer consists of two parts
  - A header for control information - often divided into multiple fields
  - A payload part for carrying data - not all packets may have payload

Encapsulation - decapsulation

- Encapsulation/decapsulation performed at layer boundaries
  - Payload data to be sent is obtained from the upper layer
  - Encapsulation on the sending side: header(control) information is added to the payload data
  - Decapsulation on the receiving side: Header(control) information is removed to extract the payload data
  - Payload data is then provided to the upper layer

Definitions

- **Encapsulation:** å legge ein melding/pakke type fra et overliggende nivå inn i ein melding/pakketype for the underliggende nivå
- **decapsulation:** Å ta en data ut av ein melding/pakketype før det gis videre til det overliggende nivå?
- **Multiplexing:** Å legge til kontrollinformasjon i ein pakke/melding som identifiserer en komponent på mottakersiden på tilsvarende nivå som skal motta/håndtere payload i pakken/meldingen
- **demultiplexing:** Å gi ein pakke/melding videre basert på kontroll(header) informasjon til ein komponent som skal håndtere pakken/meldingen videre

Multiplexing - demultiplexing

- Multiplexing/demultiplexing may happen internally at layers
  - Entities at a given layer often have logical connection

- Multiplexing at the sending side: the data to be sent is mapped onto one underlying connection
- Demultiplexing at the receiving side: The data received is mapped back to the logical connection
- The mapping is done via control information in the headers

Frame integrity and checksums

- Frames may be lost but if they are received they have the same content as when they were sent
- How to check fram integrity?

Network applications

- A set of processes running on different hosts communicating using the transport services
  - The programs may be implementing different programming languages
  - The programs may run on different platforms (OS or hardware)
  - The programs running on hosts need not be identical
  - Relies on protocls executed between the application layer entities

Web and web services

- Network applications that allow clients to retrieve and manipulate objects stored on servers
  - Objects are addressed and referenced by URL(Uniform Resource Locators)
- Hypertext Transport protocol(HTTP)
  - Transfer protocol following a request-response pattern
  - Defines how resources can be retrieved and manipulated
  - Provides the foundation of the world-wide web and web-based applications
  - Serve also as a basis for REST-based web services and the WebSocket protocol

Basic HTTP operation

1. Client opens TCP connection to a server on a server port (typically 80/8080/443)
2. Client sends an HTTP request message containing the operation to be performed on the object
3. Server performs the requested operation and sends an HTTP response message
4. Server and client closes the TCP connection

HTTP request message

- Sent by the client to access an object on the server
- HTTP methods
  - GET
  - PUT
  - POST
  - DELETE
  - ...

HTTP response message

- Sent by the server in response to a client request

Intra layer protocol communication

- The sending of payload data at a given layer may involve multiple exchanges of packets and control signalling
- Example:
  - Opening and closing TCP connection before and after sending HTTP requests and receiving responses
  - Splitting a large payload from the upper layer into smaller parts which are then sent individually
  - Sending acknowledgement on received packets at the link and at the transport layer
  - ...

HTTP and REST-based web services

- HTTP is also more generally used for REST-based web services/RESTful APIs
  - GET: Safe and idempotent operation to retrieve a resource
  - POST: unsage and non-idempotent operation to create a resource
  - PUT: unsafe and idempotent operation to update a resource
  - DELETE: unsafe and idempotent operation to delete a resource

idempotent: Describing an action which, when performed multiple times, has no further effect on its subject after the first time it's performed

Network and protocol security

- Security is an important consideration from the link-layer to the application layer
- **link layer:** capture all frames on a wireless LAN
  - can be done by putting wireshark in promiscious mode
- **Application layer:** reading content of HTTP Messages
- Secure HTTP (HTTPS)
  - Relies on the Transport layer security (TLS) service provided by transport layer
  - Provides confidentiality via encryption of the HTTP messaging
  - Uses by convention TCP port 443 on the side server
  - Protocol is selected using https in front of URL instead of just HTTP

Remote procedure call

RPC (Remote procedure call)

- Remote procedure call is used to communicate between remote processes
- The main idea behind remote procedure call:
  - Allow client process to call (execute) the procedure of a remote process

Issues in RPC

Since distributed machines are heterogenous (platform, language, machine architectures) - Interoperability is a major challenge

- Heterogeneous hosts
  - Differences in address spaces on different hosts
    - Parameter passing by reference
  - Differences in memory addressing on hosts

- Big Question is:
  - How do we exchange messages between different hosts without ambiguity? (i.e. both the sender and the receiver interpret the message the same way)
- Client and server operate in different address spaces in memory
  - Parameters and results have to be passed
    - By value or reference?
- Issue if the machines are not identical
  - Big-endian vs. little endian memory addressing
    - Where we start reading address in memory from. start or end

Solution for RPC issues

- Design a middleware
  - Responsible for implementing the RPC protocol
    - Client stub
    - Server stub

RPC stub generation

- For RPC to work, both the caller and the callee must agree on the format of the messages they exchange (i.e. must follow the same RPC protocol)
- Message format definition is one aspect of an RPC protocol, but it is not sufficient
- To remove ambiguity/misunderstanding: client and the server need to agree on representation of simple data structures, such as ints, chars, bools
- Example:
  - The protocol could prescribe that ints are represented in two's complement, chars in 16-bit unicode, and floats in IEEE standard #754 format, with everything stored in little endian.
- Lastly, caller and callee agree on the actual exchange of messages e.g. using TCP or UDP connection.
- The client and server stubs need to be implemented
- Stub is a piece of code that converts parameters passed between client and server during a RPC
  - Hide the complexity of performing a RPC.
- One of the roles of the stub code is to marshall the necessary data into a format that can be sent as a byte stream to a remote server
- Client stub is responsible for conversion(marshalling) of parameters used in a function call and deconversion (unmarshalling) of results passed from the server after the execution of the function
- Server skeleton (server stub) is responsible for unmarshalling of parameters passed by the client and conversion(marshalling) of the results after the execution of the function
- Stubs can be generated in two ways:
  - **Manually:** The RPC implementer provides a set of translation functions from which a user can construct his or her own stubs
  - **Automatically:** uses an interface description language(IDL) to define the interface between client and server.(embedded in programming language environment)

Application socket Programming

Application layer

- Exchanges of messages between processes running on hosts - any software that runs on top of the transport services
- Important application layer protocols
    - Remote login to hosts - Telnet/SSH
    - File Transport Protocol (FTP)
    - Electronic mail Transport Protocol (SMTP)
    - Domain Name System (DNS)
    - Simple Network Management protocol (SNMP)
    - HyperText Transport Protocol(HTTP)

Internet Directory service and the Domain Name System(DNS)

DNS

- Internet hosts are identified by hostnames in applications
    - www.hvl.no
    - www.google.com
    - www.oh.no
- Hosts are identified by 32/128-bit IP addresses at the network and transport layers
- Applications require a translation service from hostnames to IP addresses
- Example: An HTTP client needs the IP address of the HTTP server to open the underlying TCP connection used
- The domain name system consists of
    - Hierarchically organised collection of servers providing a distributed database storing resource record
    - An application layer request-reply protocol defined in RFC1034 and RFC 1035
        - https://datatracker.ietf.org/doc/html/rfc1034
        - https://datatracker.ietf.org/doc/html/rfc1035
    - DNS is based on the UDP transport service (usually port 53) for querying the distributed database for resource record
- Based on a client-server architecture providing
    - Hostname to IP address translation
    - Host and server aliasing: a host may a canonical hostname and several alias hostnames(e.g www.hvl.no, hvl.no)
    - Load distribution: a set of IP addresses can be returned for a hostname(e.g. for replicated web servers/ www.google.com)
    - The nslookup-utility is a command-line client available on most platforms

DNS System architecture

- DNS database is hierarchically organised and distributed on multiple servers
    - Root DNS servers: contains IP address of DNS servers for top-level domains(.com, .org, .edu, .no)
    - TLD DNS servers: Contains IP addresses of authorative DNS servers for organisations
    - Authoratative DNS servers: Contains the mapping from hostnames to IP address for the organisations' publically accessible hosts

Local DNS servers

- Organisations and ISPs typically have local DNS servers(name servers)
- DNS queries are sent to a local DNS server which acts as a proxy and performs further DNS queries/lookups if required
- Example:
  - Host `cis.poly.edu` finding the IP address (128.119.245.12) of `gaia.cs.umass.edu`

DNS queries

- The DNS servers acts as clients when contacting other servers to resolve a hostname
- Two types of DNS queries that can be used in combination
  - **Iterative queries:** DNS client obtains a reply containing the next server to contact.
  - **Recursive queries:** DNS client requests the DNS server to resolve the hostname
- A DNS query chain may also involve intermediate DNS servers storing IP addresses of authoratative servers
- Caching is applied to reduce DNS messages
  - DNS servers cache replies which are being received
  - Caching of direct mappings and addresses of TLD, Authoratative, and intermediate DNS servers.

DNS messages

- DNS query and reply messages have the same format

Sockets and network programming

Network applications

- A set of processes running on different hosts communicating using transport services
  - may be implemented using differend programming languages
  - may run on different platforms(OS and hardware)
  - running on hosts need not be identical
  - Relies on protocols executed between the application layer entities

basic TCP/IP transport services

- Transport Control Protocol (TCP)
  - **Connection oriented:** a TCP connnection between sockets is established prior to exchange of data
  - **Reliable service:** The stream of bytes sent will be equal to the stream of bytes that are received(no loss and the order of bytes is preserved)
  - Provides **flow-** and **congestion control:** adaptive transmission rate based on receiver speed and in case network congestion is detected
- Universal Datagram Protocol (UDP)
  - **Connection-less:** No establishment of connections between sockets
  - **Unreliable service:** datagrams may be lost, duplicated or received out of order(but integrity of datagrams is guaranteed)
- Transport endpoints are identified by an Internet Protocol IP address and a port number

Process communication and sockets

- Processes access the transport service using socket API provided via OS
  - Processes creates and closes sockets via API
  - Processes send and receive messages via sockets

Java Socket API

- Provided by `java.net` package
  - `DatagramSocket`: represents a socket for sending and receiving datagram packets(UDP)
  - `DatagramPacket`: represents datagram packet(UDP)
  - `ServerSocket`: Implements server socket (TCP)
  - `Socket`: implements client socket(TCP)

Summary

- UDP and TCP transport services are available via a socket API
  - Sockets are created by applications/processes and represents communication endpoints
  - Sockets are controlled by the OS and can be used to both send and receive data
- The Socket API enables communication between
  - Applications implemented in different programming languages
  - Applications/processes running on different platforms
- Initial testing of networked applications can be performed using the lookback interface and network(localhost)

Example application

- An application based on the client-server architecture implementing a "convert-to-uppercase" network service
- Operation of networked application
  1. The Client reads a line of text from the user
  2. client sends the text to the server in a request message
  3. server reads the line receives from the client
  4. server converts the line to uppercase
  5. servers sends the modified text to the client in response message
  6. Client receives the server response
  7. client extracts the received text and shows it to the user

testing network applications

- testing of networked applications acan be challenging
- Implementations of the TCP/IP protocol stack supports a loopback intefrace and network
  - Virtual network and connection provided by the OS
  - Datagrams sent on the loopback interface is immediately passed back as if received from a real network
  - Usually named `lo0` and `localhost` with IP address `127.0.0.1`(IPv4)
- The communication processes can be executed on the same computer using the loopback network as the Internet

Processes

Topics

- Threads and processes in DS
- Multithreading in client and server designs
- Scaling in DS(LAN and WAN)
- Virtualization

Main goal: How processes/threads are used to provide distribution transparency in client and server design for distributed systems

Process and threads

- A process has at least one thread
- Each process runs in its own address space
- Thread run within the address space of its process

Threads in DS

- Multithreaded clients and servers

Multithreaded Clients

- **Example:** A web browser does a number of things concurrently : e.g. Fetch text content, fetch images, fetch other data and so on. Designed as multithreaded. Each thread setup a different connection to the server and pulls in the data
- **Benefits:**
    - Hides communication latency as much as possible(data keeps flowing to the page until everything is delivered)
    - Several connections can be opened simultaneously. Web servers can be replicated across many machines allowing multithreaded clients to open concurrent connections to replicas and transfer data in parallel

Multithreaded servers

- Main use is found at the server side
- Multithreading simplifies server code
    - When multiple services should be granted to multiple clients
- Better exploitations of multiprocessor architectures(parallelism)
- Better performance

Server-design for Distribution transparency

- General Server design issues
- Issues:
    - Concurrent or iterative server
        - Iterative server: The server itself handles the request and, if necessary, returns a response to the requesting client.
        - Concurrent server: it does the request itself, but passes it to a seperate thread or another process, after which it immediately waits for the next incoming request. e.g. Multithreaded

server
    - Contacting a server via endpoint. How do we identify a server on a host transparently?
        - Clients request to an end point via port on the machine where the server runs HTTP(80), HTTPS(443), FTP(21), SSH(22), - Internet Assigned Numbers Authority(IANA)
        - Not every service requires a pre-assigned end point. Use dynamic port assigned by the local OS
    - Stateful or stateless servers
        - Stateless does not keep information on the state of its clients, and can change its own state without having to inform any clients, e.g. A Web Server
        - A stateful server does maintain persistent information on its clients, e.g. a file server that allows a client to keep a local copy of a file
    - How can a server be interrupted transparently?

Clusters in LAN

- Important goal: access and replication transparency
    - Hide the fact that clients may be interacting with multiple servers with replicated data
- Three different tiers
    - Switch: forms the entry point
    - The first tier is generally responsible for passing clients requests to approapriate servers
        - This is known as request dispatching

Clusters in WAN

- Main issue: Redirection policy - Which server should handle the client request? Locations transparency in addition to access and replication
    - How to redirect clients to the nearest server location
    - Estimate the latency between the client and several servers
    - Solution: DNS(Domain Name System)
- Drawback:
    - DNS server acting as a proxy usually sends its own address during the clients request to another proxy
        - DNS IP is returned for the client(Locality awareness lost)

WAN: Wide area network

Virtualization

The partitioning of the resourcces of a physical system

- Heterogeneity in networked and distributed systems
- Ease of portability and code migration
- Isolation of failing or attacked components(reliability)

Ways toi virtualize

- Process VM
    - Seperate set of instructions, an interpreter/emulator, running atop an OS e.g. JVM
- Native VMM

  - Low-level instructions, along with bare-bones minimal operating system e.g. Microsoft Hyper-V
- Hosted VMM
  - Low-level instructions, but delefating most work to a full fledged OS. e.g. Oracle VirtualBox

Application of VM to DS

Cloud computing is one of the major application of VM technologies in distributed systems.

IaaS: instead of renting out a physical machine, a cloud provider will rent out a VM or VMM that may possibly be sharing a physical machine with other customers => almost complete isolation between customers(high performance isolation may not be reached)

Benefits of Virtualization in DS

- **Porting of legacy systems:** Legacy software comes from the relatively low change rate of high-level software, while hardware and low-level systems change quite fast - to keep high-level software working, virtualisation is of help
- **Portability:** Heterogeneous computing platforms are interconnected and should make diverse applicatoins run - Which could bring their own environment with them
- **Replication:** A server could be completely replicated whenever and wherever needed - e.g. edge servers

Communication

Topics

- Asynchronous RPC
- Message-oriented communication
  - MQTT examples

Traditional RPC

- In traditional RPC, process blocks until the operation is complete
  - i.e. the processes are synchronised

Asynchronous RPC

- Asynchronous RPC aims to get rid of the strict request-reply behavior, but let the client continue without waiting for an answer from the server

Combining Asynchronous RPCs

- Deferred Synchronous RPC combines two asynchronous RPC to provide an ad hoc form of synchronicity
- The first asynchronous call selects the procedure to be executed and provides for the parameters
- The second asynchronous call returnss the result to the client.
- In between, the caller may be on computing
- This is realized by using a callback function to the client

Multicast RPC

- Essence: Sending a RPC request to multiple RPC servers
- Example: Replicated server processing different parts of a taks(parallel tasks for compute-intensive jobs)
    - Password cracking
    - Breaking encryption algorithm by exhaustive search
    - Search for files with certain text pattern
    - Finding prime numbers(RSA crypto key algorithm)

Message oriented middleware

- Two major weaknesses with RPC:
    - Both the sender and the receiver have to be running during message exchanges(coupled in time)
    - Traditional RPC is synchronous - blocking mechanism
- Message oriented persistent communication
    - Message-oriented Middleware(MOM) or Message-queueing systems

MOM model

- Sender and receiver are loosely coupled in time
- Producer/sender
    - a process that sends message
- Queue
    - A buffer that stores messages
- Consumer/Receiver
    - A process that receives

Persistent message-oriented-middleware

- Essence:
    - Asynchronous persistent communication through support of middleware-leve queues
    - Application communicate by inserting messages in specific queues
    - Queues corresponds to buffers at communication servers
    - Allow us to have a loosely coupled architecture
        - Processes are loosely coupled in time

Queue operations

- put
    - Append a message to a specified queue
- get
    - Block until the specifie queue is nonempty, and remove the first message
- poll
    - Check a specific queue for messages, and remove the first. Never block
- notify
    - Install a handler to be called when a message is put into the specified queue

Messages have to be properly addressed - provide a systemwide unique name of the destination queue.

MQTT - example

- Message queue Telemetry transport(MQTT)
- A lightweight message queueing and transport protocol
- Suited for M2M, WSN, IoT scenarios
- Example: building control applications
- Asynchronous communication model
- Publish/subscribe model
- Decoupling of data producer(publisher) and data consumer(subscriber) through topics (message queues)

MQTT model

- Client
  - =publisher, subscriber
- server
  - =broker
- Topics
- Session
- subscriptions

MQTT message format

Message types

- Connect
- Connack
- publish
- puback
- pubrec
- pubrel
- pubcomp
- subscribe
- suback
- unsubscribe
- unsuback
- disconnect
- pingreq
- pingresp

Naming

Naming in DS

- Importance of naming in DS
  - Share resources
  - Uniquely identify entities
  - Refer to location
- Implementation
  - Distributed across several machines

- Core issues
  - Resolution of names to the entities they refer to
  - Scalability and efficiency of naming system

Name, Entity, Address, Access point, identifier

- **names:** Bits or characters used to refer to an entity
- **Entity:** Can represents virtually anything
  - resources (hosts, printers, disks, files), processes, messages, users
- **Access point:** where resources/processes are contained or can be found
- **Address:** we need the address of the access points to be able to find or locate resources
- **Identifier:** A name that uniquely identifies an entity

Identifier

A true identifier is a name that has the following properties

- An identifier refers to at most one entity
- Each entity is referred to by at most one identifier
- An identifier always refers to the same entity(i.e. it is never reused)

Key points

- Entities can be operated on: e.g. Printer interface with operations for printing a document or a server that services requests
- An entity can offer more than one access point
- An entity may change its access points
- A name for an entity that is independent from its addresses is ofte much easier and more flexible to use. Such a name is called **location independent**
- If an address can be assigned to a different entity, we cannot use such address as an identifier
- Identifiers can be used to unambiguously refer to an entity

Classes of naming systems

- Flat naming
- Structured naming
- Attribute-based naming

Flat naming

Characteristics

- Entities are refered to by an identifier/name with no meaning
- It bears no structure
- e.g. MAC or memory addresses

Core issue

- Needs special mechanisms to trace the location of refered entities

Approaches for searching/resolving a name/identifier of an entity to its address

- Broadcasting
- Chains of forwarding links
- Home-based approach
- Distributed Hash tables
- Hierarchical location services

Broadcasting

- Broadcast the ID, requesting the entity to return its current address
- Can never scale beyond local-area networks (LANs)
- Requires all processes to listen to incoming location requests

Address Resolution Protocol (ARP)

- To find out which MAC address is associated with an IP address, broadcast the query "who is this IP address"

MAC: media access control, used as a network address in communications within a network segment.

Multicasting

- Broadcasting has the problems that it:
  - Wastes bandwidth because of request messages
  - Requires all processes to listen to incoming location requests(not meant for them)
- Multicasting associate a host to a multicast group using a multicast address
- Only a restricted group of hosts receives the request
- A multicast address can be used as a general location service for multiple entities
- Another way to use a multicast address is to associate it with replicated entity, and to use multicasting to locate the nearest replica

Forwarding pointers

Pensum Distribuerte systemer

- Chapter 1: Introduction
  - Section 1.1-1,3
- Chapter 3: Processes
  - Threads (3.1), Clients(3.3), and servers (3.4, pg 128-138, pg. 141-146)
- Chapter 4: Communication
  - RPC(4.2)
    - Remote Procedure Call - program causes a procedure to execute in a different address space
  - Message-oriented communication(4.3, pg. 193-212)
    - Communication models framed in terms of the transfer of messages or information, or which reduce meaning to explicit content
  - Multicast communication(4.4)
    - data transmission is addressed to a group of destination computers simultaneously. Multicast can be One-to-Many or Many-to-Many
- Chapter 5: Naming

- Names, identifiers, and addresses (5.1)
    - Names: Bits or characters used to refer to an entity
    - Identifiers: A name that uniquely identifies an entity
    - Adresses: We need the address of the access points, where resources are contained/can be found, to be able to find or locate resources
- Flat naming: Distributed hash tables (5.2)
    - Entities are refered to by an identifier/name with no meaning with no structure. Example: MAC or memory addresses
- General about Naming:
    - Share resources
    - Uniquely identify entities
    - Refer to locations

- Chapter 6: Coordination
    - Clock synchronization(6.1)
        - Coordinate independent clocks
    - Logical clocks(6.2): Lamports clocks, vector clocks
        - Lamports
            - Process increments its counter before each local event. Includes counter value when sending message after incrementing it.
        - Vector
            - All clocks init at 0.
            - Each time a process experiences an event, it increments its own in the vector by one
            - Each time a process sends a message, it increments its own clock in the vector by one and then the message piggybacks a copy of its own vector
            - Each time a process receives a message, it increments its own clock in the vector by one and updates each element in its vector by taking the max value in its own clock and the value in the vector in the received message.
    - Mutex (6.3)
        - property of concurrency control, which is instituted for the purpose of preventing race conditions. It is the requirement that one thread of execution never enters a critical section while a concurrent thread of execution is already accessing critical section.
    - Election algorithm(6.4, pg 329-333(Up to ring algorithm))
        - Choose a process from the group of processors to act as a coordinator. if the coordinator crashes due to some reason, then a new coordinator is elected on other processor. All processes have a unique priority, highest priority gets coordinator role.
            - Bully algorithm
            - Ring algorithm
- Chapter 7: Consistency and Replication
    - Introduction (7.1)
    - Data-centric consistency models (7.2, pg 358-370)
        - contract between the software and memory implementation. guarantees that if the software follows certain rules, the memory works correctly.
    - Client-centric consistency models(7.3, pg 375-382)
        - can deal with inconsistencies in a less costly way. Instead of globally conistent view, maintains consistent view for individual clients.
    - Consistency protocols (pg. 398-405)

        - Implementing consistency models.
            - Primary-based and replicated-write protocols
- Chapter 8: Fault tolerance
    - Introduction (8.1)
    - Process resilience (8.2, pg 432-438), consensus in faulty systems with arbitrary failures (pg. 449-456)
        - Key approach to tolerating a faulty process is to organize several identical processes into a group so that if one fails, another can take over.
    - Failure detection (pg. 462-464)
        - Actively send "are you alive?" messages (pinging eachother).
        - Passively wait until messages come in from different processes
    - Reliable client-server communication(8.3)
        - based on communication primitives (doOperation, getRequest, sendReply)4
    - Reliable group communication (8.4)
        - Multicasting
- Chapter 9: Security
    - pg. 501-529

Pensum Nettverksteknologi

- Chapter 1: Computer networks and the internet
    - Section 1.1
    - Section 1.2, pg 37-40, 46-49
    - Section 1.3, pg 49-54, 59-62
    - Section 1.4-1.6
    - Section 1.8
- Chapter 2: Application layer
    - Section 2.1, 2.2, 2.4, 2.7, 2.8
- Chapter 3: Transport layer
    - Section 3.1-3.5, 3.8
- Chapter 4: The network layer - data plane
    - Section 4.1
    - Section 4.2, pg. 341-353
    - Section 4.3, 4.5
- Chapter 5: The network layer - Control plane
    - 5.1-5.4, 5.6, 5.8 (Summary - mistakenly numbered 5.7)
- Chapter 6: The link layer
    - Section 6.1, 6.2
    - Section 6.3, pg. 479-483, 487-493
    - Section 6.4, pg. 495-515
    - Section 6.7, 6.8
- Chapter 8: Security in Computer Networkds
    - Section 8.1-8.4, 8.6, 8.9