Access Rights Revocation

Access List: delete access rights from access list. Simple, immediate.
Revocation of Access-Rights is implemented using an Access List.
Capability List: requires to locate the capability in the system before revoking.

Hydra System

Fixed set of access rights known and interpreted by the system.
operations on Objects are through pre-defined procedures.

Cambridge CAP System

Simpler but powerful
Data Capability & software capability

Language-Based Protection

More flexible than the OS, but high overhead with access validation.
Compiler-Based enforcement: restriction is built in to compiler.
Example: **JVM** (Untrusted Classes, Stack Inspection)

Forms of misuse

there are several forms of accidental or malicious misues:

Breach of **Confidentiality**: unauthorized reading of data.
Breach of **Integrity**: unauthorized modification of data.
Breach of **Availability**: unauthorized destruction of data by causing havoc or defacement.
Theft of Service: unauthorized use of resource.
Denial of Service: preventing legitimate use of service.

Attacking Methods

Masquarading: pretend to be s/o
Replay Attack: keep system busy by repeat of valid data transmission.
Message Modification
Man in the middle attack
Session Hijacking: MITM + intercepting an active session.

Program Threats

Trojan House: code segment misuses its env, login emulation, spyware. Attached to a program, but does not replicate.
Trap Door: leaving secret access point, backdoor
Logic Bomb: initiates a security incident under certain conditions.
Stack and Buffer Overflow: overload the data until return address is modified. Solution: SPARC and Solaris throws exception when exec from stack memory, Linux and Windows XP mark those page as non-exec.
Virus: attach itself to a program, self replicate & infect other program. Specific to CPU arch, OS, apps. **Why Windows has more virus?** Linux/UNIX has separated users/roots, where as Windows users usually have admin privilege, and Windows systems outnumber UNIX/LINUS. Categories of virus: Macros, File, Boot sector, Source Code, Polymorphic, Encrypted, Stealth, Tunneling, Multipartite, Armored.
Worm: replicate functional copies of themselves but as separate entities.

Symmetric Encryption

Mostly base on **Transformation**.
DES: Most commonly used **symmetric block-encryption** algorithm. 64-bit chunk value and 56-bit key. XORED previous ciphertext before encrypt.
Triple DES: improve version of DES with 3 times encryption using 2 or 3 keys.
AES (Advanced Encryption Standard): key length 128, 192, 256, and data chunk of 128-bit.
Twofish: variable key up to 256-bit, 128-bit chuk value.
RC4: **Stream Cipher**, when length of comm makes block cipher slow. Used in WEP, HTTPS.

Asymmetric Encryption

Key difference with symmetric: the enc and dec keys are different.
Mostly base on **Maths Function**, not Transformations.
Not for large amount of data.
Used for small amount of data, authentication, confidentiality, key distribution.
RSA: block-cipher public key algo.

- Encryption algo is $E(k_e, N)(m) = m_e^k mod N$, where k_e satisfies $k_e k_d mod(p-1)(q-1) = 1$; decryption is $D(k_d, N)(c) = c_d^k mod N$

Authentication

Authentication is used to verify a msg or doc was authored by a certain party, and not altered or modified, i.e. integrity verification.

- Each msg has an authenticator (generated by the sender) and will be verified by the receiver.
- There are 2 types of auth algos: MAC (symm enc) and Digital Signature (asym enc, key is inversed).
- Fewer computations, auth is shorter than msg, for non-repudiation.

Key Distribution: use CA to prove who owns the public key.
Application: SSL, HTTPS, IPSEC/VPN

User Authentication

Use password, symmetric, asymmetric enc, user identity (key, card attribute, fingerprint, retina, etc)
One Time password: uses SecurID, S/K system.

Security Defense

2 types: Intrusion Detection (IDS), and Intrusion Prevention (IPS).
Intrusion: signature-based detection of dangerous behavior patterns (need benchmark of normal behavior first, requires upgrade of signatures), anomaly detection.
False Alarm = False Positive (must be low), Missed Intrusions = False Negtive.
Network firewall: DMZ - semitrusted domain.

3.1. Security Classification

Base on US Department of Defense: D(minimal security), C(some protection), B(C+sensitivity labels), A(formal design, verification techniques).

1.1. Distributed System:

Definition: Collection of Processors that do not share memory or clock, communicating through networks.

- **Processors**: varies in size, functions; including all types of computers.
- **Site**: location of a computer system.
- **Host**: specific system at a site (e.g., server, client)

Four **reasons** for Distributed Systems: **resource sharing**, **computation speedup**, **reliability**, and **communication**.

1.2. Network-Based Operating Systems

There are 2 types of network based OSs.

1.2.1. Network Operating Systems

User must know the techniques, or which resources to query, where to obtain the resources, i.e. command sets before hand. **Example**: *Remote logging* (telnet, ssh), *Remote desktop* (RDP), *Data Transfer* (s/FTP, SCP)

1.2.2. Distributed Operating Systems

User does not need to have special knowledge about remote system, just **access them as local resources**, all *data migration*, *computation migration* and *process migration* is taken care by the distributed OS.

Two techniques to move processes in network: by OS (transparent to users), or by users' specific inputs.

Characteristics of Client / Server Computing: ely on user-friendly app, share services on server side, common DB server, high priority on network management and secury.

1.2.3. Client / Server Computing

Key feature is allocation of tasks between Client & Server.

- Client / Server must share same protocol, support same app.
- Easy to use GUI
- Possible to have different OSes

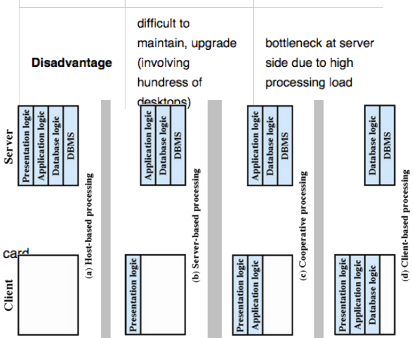
Client/Server Classes

There are 4 classes of Client/Server Computing:

- **Host-based**: dumb terminal, traditional mainframe
- **Server-based – aka Thin Client**: server does all processing, client only does presentation.
- **Cooperative (Fat Client)**: application logic is shared between client & server, complex to setup but greater user productivity & network effeciency.
- **Client-based (Fat Client)**: **Most common model**, all processing done at client, only DB logic at server side. Represented by Relational Database, example is SQL database provides db service to Client side.

Fat Client v/s Thin Client

	Fat Client	Thin Client
Advantage	less bottle neck at server side	migration path from mainframe to distributed computing network
Disadvantage	difficult to maintain, upgrade (involving hundress of desktops)	bottleneck at server side due to high processing load



2.1. LAN

Network communication can be implemented by:

- **Ethernet** (multiaccess bus): simple, reliable, cost effective; or
- **Token Ring**: deterministic, far distance, great throughput under heavy load.
  - Cons: Require special HW thus increase cost, and risk of losing token.

WAN

- Originated from Arpanet 1968, a packet switching network.
- Or PPP (Point-to-Point) connection over modems.

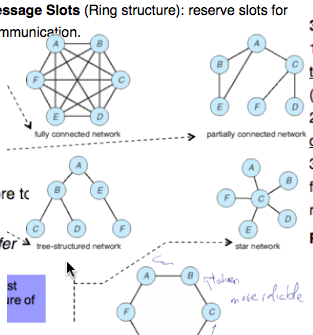
Communication Structure

5 basic problems to solve: **Naming and name resolution**, **Routing strategies**, **connection strategies**, and **Contention**.

5.1. Contention

There are 2 methods to solve contention in communication.

1. **Token Passing**: circulates a token in the system.
2. **Message Slots** (Ring structure): reserve slots for communication.



Network Topology

Has different topologies, such as Fully Connected Network, Partially Connected Network, Tree-Structured, Star Network, or Ring Network.

Routing Strategies

Fixed Routing: direct path between A-B is pre-defined.

- Pros: shortest path can be chosen to minimize cost, ensure ordering of messages.
- Cons: unable to adapt load changes.

Virtual Circuit: a path from A to B is defined for the session duration. Same pros as Fixed Routing, and improve the handling of load changes by specifying the path avoiding congestion.

Dynamic Routing: path between A to B is chosen when a message is sent, by the router or hops.

- Pros: adapt to load changes
- Cons: message may arrive out of order.

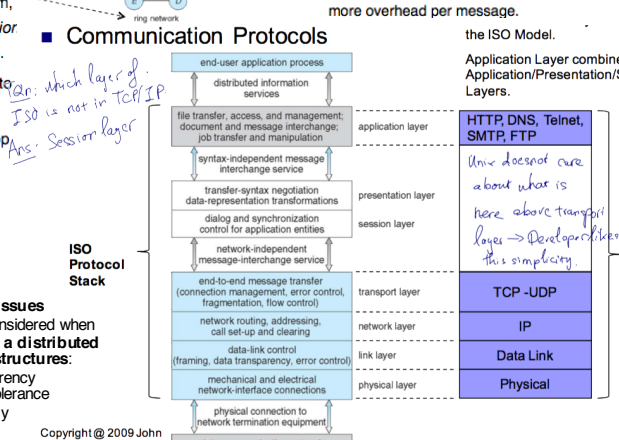
3 Connection strategies (scheme):

1. **Circuit Switching** - a permanent link is established throughout the communication session (telephone/modem).
2. **Message Switching** - a temporary link is established during one message tranfer.
3. **Packet Switching** - split the message into smaller fixed-length packets and dispatching to the destination, regardless of orders, routes.

Pros/Cons:

- Circuit Switching requires more setup time, but less overhead for each message, may waste bandwidth if idle.
- Message and Packet switching less setup time, but more overhead per message.

Communication Protocols



Copyright @ 2009 John Wiley & Sons Inc.

Q: which layer of ISO is not in TCP/IP
Ans: Session layer

Unix doesn't care about what is here above transport layers → Developer likes this simplicity

the ISO Model.

Application Layer combines the ISO Application/Presentation/Session Layers.

Design Issues
To be considered when designing **a distributed system structures**:
•Transparency
•Fault Tolerance
•Scalability
•Clusters

Robustness
Able to detect link failure, site failure or message lost.

2.1. Naming Structures

There are 2 types: **Location Transparency** – *i.e. one can't determinte the server hosting the file by the filename*, and **Location Independence** – *filename can stay intact when file physical location is changed.*

Comparison between Location Transparency and Location Independence

Location Transparency	Location Independence
Simple, and more desirable for security (as user does not know where the file location is)	*Convenient as user does not need to lookup where the file locates even it's moved between physical disk drives.*
Static, does not support file migration	*Dynamic, support file migration*
Movement of file names & disk are manual	*Migration is done by software*
not popular	*more popular (implemented in AFS: Andrew File System)*
Convenient to share remote files as similar as local files	*better abstraction since logical data container not attached to specific location*
Able to balance the utilization of disks across the system	*Naming hierarchy separated from storage devices hierarchy*

In current trend, **OS and Networking** software are locally stored, **User Data and System utilities** are remotely stored.

Reconfiguration

Ability to reconfigure & recover on failure and resume operation.

DFS is a file system where multiple users share files & directories which are physically dispersed among different distributed systems, using network and low-level protocols. A DFS system consists of:

- **Service**: is the certain type of functions provided by the software entity, running on the systems, to the client.
- **Server**: is the computer system that consists of multiple services.
- **Client**: is the process that invokes and consumes the service from the server.

- *Client Interface for a file service is the form of primitive file operations (Create/Delete/Read/Write), should be transparent. Client Interface does not distinguish between local and remote files.* [1] Ans: … The client **needs to know about** the protocol & structure & which server to connect to on the server.

Component Unit: smallest set of files that can be stored on a system

DFS Performance Measurement: 2 factors, (1) time to satisfy the Service Request, (2) Transparency *(how comparable the DFS is to conventional file system)*

- **Client Initiated approach**: cache validity check is initiated by the Client (session base, thus stateful). It checks (a) before every access, or (b) on first access, or @ a fixed interval time.
- **Server Initiated approach**: cache validity check is initiated by the Server (stateful as well). It records and resolve conflicts if it detects a potential inconsistency. Caching could be disabled for particular file while switching to Remote Server mode.

2.3.4. Consistency

Main question about consistency: is Local cached data consistent with Master copy?

There are 2 solutions:

- **Client Initiated approach**: cache validity check is initiated by the Client (session base, thus stateful). It checks (a) before every access, or (b) on first access, or @ a fixed interval time.
- **Server Initiated approach**: cache validity check is initiated by the Server (stateful as well). It records and resolve conflicts if it detects a potential inconsistency. Caching could be disabled for particular file while switching to Remote Server mode.

2.3.3. Cache

Cache Scheme: only blocks, not entire file are copied from server to local client. LRU is used to contain cache size. Since copies of master file are scattered on different systems, there will be **Cache-Consistency Problem** *(keeping cached copies consistent with the master file).*

Cache Size: Cache can be large, or small. **Large caches** increase hit and miss ratio at the same time, have more data transferred) and increase consistency problems. **Small caches** are not useful with large block size.
AFS cache size is 64KB, Unix block sizes are 4KB and 8KB.

Cache Location: either in memory or on disk. **Disk caches** is more reliable and does not require re-fetch after recovery. **Main-memory caches** is faster, performance speedup in larger memory, workstation can go diskless. [1]
Ans: Memory cache is more reliable than disk cache (false statement).

> **Most DFSs only implement memory cache**, not disk caching.
> **NFS**: use both Caching (main-memory) and RPC. However in Solaris 2.6+, NFS uses Memory, Disk cache, then RPC in corresponding order.

Cache Update Policy: policy to update master copy.

- **Write-through**: immediately write to disk as soon as updates on cache. Reliable, but poor performance *(every update triggers a write to server, and network latency can cause waiting)*. a.k.a. Read Cache - Write RPC. [2]
  Ans: … it has poor performance in Read. (false statement)
- **Delayed-write**: update cache first, then update master later. Good performance, fast turnaround, only last update is going to master, but poor reliability *(lost unwritten data if user machine crashes)*.
  - Mechanism: Scan cache at regular intervals and flush the variation (difference since last scan) of modified blocks or removed blocks are flush to server, result in good performance and client cache might not be flushed for a long time.
  - Variation **Write-on-close**: update master when the file is closed, good for frequently modified files or files open for long periods (less refresh, less hit on the traffic).

> - NFS uses delay-write, syncs metadata changes synchronously to the server to avoid directory structure corruption if Client or Server crashes.
> - AFS uses Write-on-close, delay closing process on client until file is written on server.

	Caching	Remote Service
Pros	Faster with local cache	Simplicity, Suitable for diskless
	lesser traffic & load, esp. large data blocks	small-memory-capacity systems
	Suitable for **Stateful** connection	Suitable for **Stateless** connection
Cons	Complicated, not so effecient when Writes are frequent than Read	Slow, high network turnaround time and server operation overhead

Stateful	Stateless
Server tracks each file being accessed by client	no knowledge about how file are used
File identifier is used through the session, memory space (for caching on server side) is reclaimed when session ends or inactive	Each request is self-contained, No requirement to keep open files list in memory. Take longer request time
Prediction of file access & loading the content ahead	slower processing, using low level naming scheme, and may serve duplicate request.
May lose all volatile state if crash	almost unnoticeable after failure recovery

NFS V4
- stateful, no mount protocol
- support open() and close()
- Client local cache, file locks (by client to server)

Distributed coordination controls several **mechanisms** to ensure the Distributed Environment is working in order.
- Event ordering
- Mutual Exclusion
- Atomicity
- Concurrency Contrd
- Deadlock Handling
- Election Algorithms
- Reaching Agreement

AFS
Has 2 namespaces: local, and shared.
Local: its root file system
Shared: made up of several component units (volumes), communicate to server through Vice, using fid to identifies the Vice. A fid contains:
- **Volume Number**: access point to server, change if server change. *(there is mapping of VolumeID to IP)*
- **Vnode Number**: inodes of files in single volume.
- **Uniquifier**: unique identifier, allows reuse of vnode numbers for same file path.

•Fid is location transparent, file moved on server does not affect fid. The Volume location is kept on Vice server thus Fid does not need to care about real location of the server.
•**Fundamental Architecture**: Use **whole file caching**, 64KB cache size.
•Open: Read cache from Vice
•Close: write changes to Server.
•Cache is assumed valid unless notified by Server using **callback policy** *(consistency of files are managed by the server, using callback for notification of changes).* > **callback_policy**: when a file is cached, the server makes a note of this and promises to inform the client if the file is updated by someone else. Callbacks are discarded and must be re-established after any client, server, or network failure, including a time-out. Re-establishing a callback involves a status check and does not require re-reading the file itself.
•**Security:**
- No execution of client program on Vice System.
- Encrypted messages for communication

•**Protection**: has Access Lists for access control.
•**Implementation**: at UNIX kernel with a set of system calls, include
- path-name translation (volume, file, fid)
- Venus uses client's disk for data caching. For file status caching, it uses virtual memory.
- the Venus (AFS interface) and server use inodes for file access instead of path name.
- Uses LRU algorithm for cache size containment.

VxWorks
•VxWorks 5 does not distinguish between User mode and Kernel mode (only in v6).
•Events are handled in kernel.

Mutal Exclusion - In Distributed Environment
Using one of these **2 approaches: event ordering & timestamp, token passing**
Event Ordering:
if other process is about to enter its CS, it will compare its clock v/s the timestamp in the message.
if its clock ≤ timestamp, it proceeds to CS and defer replying.
if its clock > timestamp, it replies immediately to the requestor and defer going to CS.

. Preemptive, Priority Based heduling

Is a MUST for Real-Time Systems to be preemptive and real-time process should be assigned highest scheduling priority.
Preemptive **Soft** Real-Time Systems: assign highest scheduling priority, for e.g. Solaris, Windows, Linux.
Preemptive **Hard** Real-Time Systems: guaranteed service within deadline requirements.

Interrupt Latency: time from arrival of interrupt to start of routine that handles interrupt:
- Save state of current process, determine interrupt type, context switching, then hand over to ISR to handle the interrupt.
- Increased when kernel disables interrupt handler.

Dispatch Latency: time for scheduler to take current process off CPU and switch to another.
- Preemptive Kernel keeps dispatch latency low. During the conflict phase (i.e. preemption), preempt any process running in Kernel, and release resource of low-priority processes. Some times Priority Inversion is used to resolve Dispatch Latency issue.

Token Passing Potential issues:
•Lost token –> Re-election.
•Failed process –> form a new logical ring.

2. Characteristics of Real-time System

Single purpose, small size, mass-produced, specific timing requirements, and does not always provide all features such as standard desktop system.

Realtime Systems is Computer System that requires results produced within specified deadline.
There are 3 types of realtime systems:
- **Safety-Critical Systems**: if miss deadline –> CATASTROPHIC. E.g Weapon, ABS, Flight Control, etc.
- **Hard Real-time Systems**: Guaranteed critical real-time (must complete within deadline)
- **Soft Real-time systems**: Critical real-time tasks are scheduled (but not forced).

Proportional Share Scheduling
•CPU shares are divided proportionally to the deadline, if deadline is short, more shares.

Multimedia Timing Requirements
- Multimedia data must be accessed within specific timing requirements (for e.g. 24-30 fps)
- **Continuous media data** is data with specific rate requirements.

Streaming has 2 types:
- Progressive Download: content stored on client's computer (Youtube, ESPN, CNN)
- Realtime Streaming: content not stored on client's computer (TV broadcast, Internet Radio, etc.)
  - Live Streaming
  - On-Demand Streaming

Quality of Service (QoS) Requires both **CPU, Disk, and Network Rate requirements** are most important to QoS

Atomicity

Ensures all operations of a program unit are executed until completion.
Requires a Transaction Coordinator.
Using two-phase commit protocol.
- Phase 1: Coordinator sends request for prepare to all sites and wait for responses from ALL sites. If timeout or one "aborts", the whole transactions will be aborted.
- Phase 2: Coordinator makes a decision to commit or abort and inform all sites for their local recording. ACK is required from each site.
- If Coordinator dies: sites decide the state of T base on their log.
- If Site dies: it looks at it logs to recover (commit->redo, abort->undo, ready->ask C)

4. Concurrency Control

Uses **locking protocols** & **timestamp** to ensure multiple atomic transactions can execute in some "serial order", but still allow concurrent execution.

4.1. Locking Protocols

- **Single Coordinator Approach**: single site is acting as the **central Coordinator**, all lock/unlock request go thru this site.
  - Pros: Simple implementation, simple deadlock handling.
  - Cons: possible bottleneck, SPOF.
- **Multiple Coordinator Approach**: **distribute coordinator to multiple sites**, each handle different sets of resources.
  - Pros: reduces bottleneck.
  - Cons: multisites complicate deadlock handling.
- **Majority Protocol Approach**: **Lock Manager at each site**, trx sends lock request to more than half of sites where the data is stored, when majority reply, lock is obtained.
  - Pros: avoid central control, SPOF
  - Cons: complicated to implement, 2(n/2+1) messages for lock req, and n/2+1 for unlock req.
- **Biased Protocol Approach**: **Lock Manager at each site**. Read: shared locks; Write: exclusive locks. Shared locks priority > exclusive locks.
  - Pros: less overhead on read, but additional overhead on writes.
  - Cons: complex deadlock handling.
- Primary Copy Approach: one of the sites chosen as Primary Site, for handling request to lock.
  - Pros: simple implementation.
  - Cons: if primary fails, data item is unavailable.

5.1. Deadlock Prevention

Main idea is to **order the resource**, to prevent Circular Wait to happen, using timestamp and/or priority.

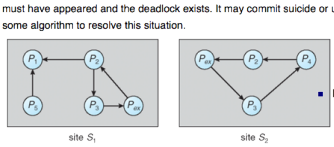
- **Wait-Die Non-Preemptive Scheme**: when a new resource request comes u
  - **older process** can wait for younger process to release its resource.
  - **younger process** will not wait for older process and thus rollback. Hence in this scheme the younger ones may tend to roll back more.
- **Wound-Wait Preemptive Scheme**: when a new resource request comes up
  - **older process** will preempt the younger one for resource, forcing it to roll-back.
  - **younger process** will wait for older process.

. Deadlock Detection

Using Wait-For Graph, both locally and remotely.

Centralized Approach: uses a central deadlock-detection Coordinator maintaining global wait-for graphs.
- Cons: as a result of false cycles (unsync arrival of msg), unecessary rollbacks may occur.

Fully Distributed Approach: uses an algorithm developed by Chandy-Misra-Haas to circulate a probe message through out the systems with its process ID in the blocked field. If the originator of the message gets the message (after it was circulated) and sees its process ID in the blocked field, a cycle must have appeared and the deadlock exists. It may commit suicide or use some algorithm to resolve this situation.



- Both sites discovers cycle in local Wait-For Graph:
  - Site S1: *Pex → P2 → P3 → Pex*
    - The edge with *Pex* is *P3 → Pex → P2* = ID(*P3*) > ID(*P2*)
  - Site S2: *Pex → P3 → P4 → P2 → Pex*
    - The edge with *Pex* is *P2 → Pex → P3* = ID(*P2*) < ID(*P3*), Send Deadlock-Detection Message.

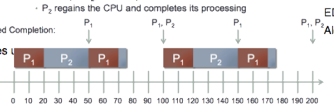
4.1. Rate-Monotonic Scheduling Algorithm

- Task's priority is inversely assigned with their period: The **shorter** the period, the **higher** the priority, and vice versa.
- Higher priority preempts the lower one.
- Process must execute on specific period p, and complete burst t during each period.
- Rate-Monotonic Scheduling can only shedule **n** processes with no more CPU Utilization than $n(2^{1/n}-1)$, in **general is 0.69**.

Example: two real-time processes
- P_1 has a period of 50 clocks, CPU burst of 20 clocks
- P_2 has a period of 100 clocks, CPU burst of 35 clocks

P_1 and P_2 can begin executing at the same time…
- P_1 has the higher priority, so it takes the CPU first
- P_1 completes its processing, and then P_2 starts…

Part way through P_2's CPU burst, P_1 must execute again
- Preempts P_2, and completes
- P_2 regains the CPU and completes its processing



. QoS Parameters

Throughput, i.e. data rate.
Delay, delay of stream data delivery.
Jitter, delays during playback.
Reliability: errors rate in transmission and processing of data.

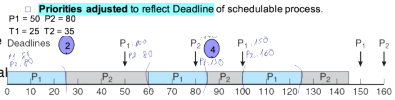
QoS Levels
Best Effort: no guarantee of requirements, mostly used in traditional OS.
Soft QoS: prioritize certain traffic streams, but still no guarantee of requirement.
Hard QoS: guarantee the quality requirements.

Guarantee of QoS is provided by **Admission Control**.
- Using Semaphores: simple admission control policy.
- Using Resource Managers: used for each type of resource
  - **CPU Scheduling**: soft real-time, and hard real-time (requires critical process to be serviced within a guaranteed period of time)
  - **Disk Scheduling**
    - using EDF, i.e. Earliest Deadline First, similar to Shortest Seek Time First, order requests by deadline. EDF may have higher seek times.
    - using SCAN-EDF, similar to EDF but grouping and order requests with relatively close deadlines. *Batch reordering must ensure request being served within deadline.*
  - **Network Management**
    - Protocols: RTP/RSTP, HTTP (stateless)
      - RTSP/RTP: Use HTTP to transfer the meta file, and RTP/RTSP for content delivery. RTSP commands include SETUP, PLAY, PAUSE, TEARDOWN.
    - Methods: Unicast, Broadcast, and Multicast. Unicast is most common.

2. Earliest Deadline First (EDF)

Priority sticks to the deadline, i.e. the earlier the deadline the higher priority.
EDF is theoretically optimal.
Algorithm
- when a process finishes (and at the beginning), take the process with the lowest
  `processTimeToDeadline - processTimeToExecute` as the new current process
- When a new process arrives, replace the current process if and only if
  `newProcessTimeToDeadline - newProcessTimeToExecute < currentProces`

☐ **Priorities adjusted** to reflect Deadline of schedulable process.



Pthread API

•Pthread API is used to manage real-time threads.
•SCHED_FIFO - for FCFS with a FIO queue, no time slicing.
•SCHED RR - SCHED FIFO with time-slicing for equal share of CPU time with same priority threads.
•SCHED_OTHER: not sure.

Election Algorithm

Election is used when the Coordinator fails. It promote active process with highest priority to become Coordinator.
There are 2 Election algorithm:

Bully Algorithm: process P discovers Coordinator is dead (no response after T time), and tries to elect itself to become the new coordinator. It sends messages to all higher numbers processes, if no response, it will forces all process with lower number to let it become the coordinator process, even there is already an active coordinator with lower number.

Ring Algorithm: circulate the active process (with priority number) to the right as candidate for election. The largest number in the active list is promoted to be the new Coordinator.

1. Which one is life threating hard realtime system scheduling?
ABS.

2. Why Virtual Mem is not good for Hard Real-time system?
Translation Time.

3. Which is not related to Interrupt Latency?
Use the scheduler to schedule the highest priority ISR (in fact, dispatch latency)

4. Which one is NOT associate with Dispatch latency
Select the Interrupt Service Routine. (this is interrupt Routine)

5. Which statement is FALSE with Rate Monotonic?
The lower the rate, the **lower** the priority (should be higher)

6. Which one is NOT the parameter for Realtime scheduling?
SHED_NORMAL

1. What is FALSE for distributed systems?
Site separated for security.

2. Which is not the behavior of Distributed OS?
Desktop migration used by OS to transfer GUI, etc.

3. Which is server based processing?
Client responsible for GUI, server do all the work.

4. FALSE statement about Token Ring?
All statements are true.

5. What is FQDN address?
sau@svuca.edu

6. Which routing is more adaptable to load changes?
Dynamic Routing.

7. Which message switching with variable length can take different path in the network?
Packet Switching.

8. Which layer of ISO is not in TCP/IP?
Session layer.

9. Which behavior does NOT belong to Network OS?
Need to know which resource to look for.

There are 3 or 4 MPEG schemes
•**MPEG-1**:
- Used for low-res video, 1.5Mbps.
- MP3 audio
- Compression ratio 200:1

•**MPEG-2**:
- Used for DVD and HDTV local playback
- Higer level (res) and profile (quality) vid compression, 1.5Mbps to 15Mbps.

•MPEG-3 (discontinued)
•**MPEG-4**:
- Used for transmitting audio/video via internet.
- Scalable quality.

1. What is the key security enabler?
Encryption.

2. Difference between Virus & Trojan Horse?
Trojan Horse does **not** replicate, while Viru does.

3. Why Windows has more virus?
- Windows is more popular, and Windows users usually have admin rights.

4. Difference between Virus and Worm?
Worm replicate itself as virus, but different entities (virus attaches itself to the host).

5. Is it feasible to derive Dec from End in Asymmetric, Symmetric?
No, Yes.

6. What are the key part in Authentication?
Authenticators, used to verify the authenticity of the messages.

7. Is Authentication Symmetric or Asymmetric?
Asymmetric, computer holding verification method can't generate authenticator.

8. How many bits MD5 produces?
128 bits.

9. How many bits SHA-1/2 produces?
160 / 224 or 256

10. Which division of computer security is highest/lowest in US Department of Defens
Highest: A, Lowest: D

1. False **Statement about DFS?**
The client needs to know about the protocol & structure & which service to connect to on the server.

2. What is wrong for memory cache?
Memory cache is MORE reliable than disk cache.

3. Which one False about Write Through?
It has poor performance in READ.

4. Which one is TRUE for stateless?
Each request identifies the file and position in full.

5. In AFS, consistency of files is managed by whom?
By server, relying on the callback provided by the client so the server can tell client when file got change.

6. What is the technique used by AFS?
Whole File Caching.