Distributed Systems - Q & A

Basics about Distributed Systems

✦ **Define the term “Distributed systems”. Be brief and precise.**

A Distributed System is characterized by:

- Multiple devices
- Connected by a network
- Cooperating on some task

✦ **Consider this statement: “I think there is a world market for maybe five computers.” Please state the key consequences for distributed systems if this sentence had been the correct vision.**

Distributed system would not exist according to the definition we have for it today. Each computer would be more or less isolated and dedicated to performing only its own tasks and there would be little chans for the computers working together.

✦ **When designing or developing a DS, its distributed nature presents a number of challenges, which we discussed in the course. List at least six such challenges.**

- Latency in communication
- Changes of network topology
- Security over network
- Restrictions on bandwidth
- The network is not homogeneous
- Network is not reliable. We can lose communication during transferring
- There is no admin in the system to manage

✦ **The internet is one of the largest Distributed systems today. Its architecture follows the so-called OSI Reference Model of 7 layers. Briefly name the seven layers and note the functions of each layer.**

1. Physical: Transmits and receives the raw data to communication medium. (e.g. 10BaseT)
2. Data link: Detect and correct errors, Organize data into packing and sequence packets, Accepts ACK from receivers. (e.g. Ethernet MAC)
3. Network: Relay and route information to destination. Manage journey of packets and figure out intermediate hops(e.g. IP)
4. Transport: Provide consistent interface for end-to-end communication. Congestion and flow control. (e.g. TCP, UDP)

5. Session: Manage multiple logical connection, Software implemented switch, established and end communication(e.g. HTTP, SSL)
6. Presentation: Data representation, Convert between machine representations (e.g. MIME, MIDI)
7. Application: Collection of app-specific protocols. (e.g. SMTP, FTP, …)

* Internet stack contains Network access(physical+data link), Internet, transport and Application.

✦ Why is IP a good choice for the core of protocol stack?

IP is a simple protocol:

– Single main task
  - Forwarding: try to bring packet from source to destination (end-to-end)
  - Other services (reliability, congestion control, security, …) on layers above (and below)
  - All lower layers:
    do not provide globally valid addresses

– Reason for success of the Internet

✦ What is the protocol layering and what is a good one?

- Communications stack consists of a set of **services**, each providing a service to the **layer** above, and using services of the layer below
- Narrow, simple interfaces between the layers

✦ Name network components

- **Network**: Collection of hosts, links, and routers
- **Site**: Stub network, typically in one location and under control of one administration
- **Firewall/NAT**: Box between the site and ISP that provides filtering, security, and Network Address Translation
- **ISP**: Internet Service Provider. Transit network that provides IP connectivity for sites
- **Backbone ISP**: Transit network for regional ISPs and large sites
- **Inter-exchange (peering point, IX)**: Broadcast link where multiple ISPs connect and exchange routing information (peering)
- **Hosting center**: Stub network that supports lots of hosts (web services), typically with high speed connections to many backbone ISPs.
- **Bilateral peering**: Direct connection between two backbone ISPs

✦ Ethical challenges: Certain distributed systems such as bitTorrent and TOR trigger ethical challenges. List and briefly discuss two ethical challenges for each of these two systems.

BitTorrent:

- Use to distributed legal and illegal content (Copyrighted material VS Documents of corruption)
- Hard to remove illegal content

TOR:

- Bypass Internet censorship in parts of the world
- Conceal interaction with gambling, drug, … sites
- Law enforcement
- Provide anonymous services

✦ What entities are communicating in a DS?

- System-oriented entities
  - Processes
  - Threads
  - Nodes
- Problem-oriented entities
  - Objects (in *object-oriented programming* based approaches)

✦ How do the entities communicate ?

If they communicate directly they will be suffer by space coupling and time coupling. Aiming to fix this problem they communicate indirect via a middleware. Types of indirect communications is:

1. Group communication: Enable one-to-many communication and use the bandwidth efficiently but nodes have time coupling

Communicating entities (what is communicating)	
System-oriented	*Problem-oriented*
• Nodes • Processes • Threads	• Objects

Communication Paradigms (how they communicate)		
"Raw"	*Middleware*	*Indirect Communication*
• Sockets	• "Library"	• Group communication • Publish-subscribe • Message queues

2. Publish subscribe: An event-based communication mechanism
3. Message queues: This is a refinement of publish subscribe arch. where queues ensure message delivery. This arch. enables space&time decoupling.

✦ Define Centralized arch. Pros/Cons

Server provide a service and will be serve the service upon a request from a client.
Advantages:

- Simplicity and centralized control

- Computation-heavy processing can be offloaded to a powerful server, Clients can be “thin”

Disadvantages:
- Single-point of failure at server
- Scalability: We have only one server, it is not scaled.

✦ Define decentralized systems and explain

All nodes are peers here and peers are equally privileged participants in the app
Advantages:
Scalable design
limit cost for service provider
Disadvantages:
Free riding
Limited control

✦ Layers & Tiered Arch:

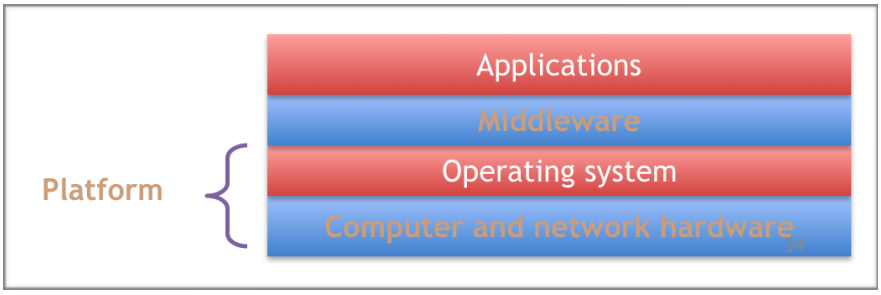
Layering is horizontal splittring of services while tiering is vertical
Tiered arch: Organize the functionality of a service and place the functionality into appropriate servers.
Layer: Layering simplifies design of complex distributed systems by hiding the complexity of below layers

✦ Distributed Systems can be organized into three layers

1. Platform
   - Low-level hardware and software layers
   - Provides common services for higher layers

2. Middleware
   - Mask heterogeneity and provide convenient programming models to application programmers
   - Typically, it simplifies application programming by abstracting communication mechanisms

3. Applications



✦ **In dinning philosophers problem: sketch in pseudo code a solution that is**

```
Think
trying = true or die
While trying
    s = random(left,right)
    Wait for fork s then take it
    If fork ~s available take it
       else drop fork s
Eat
drop s=random(left,right)
drop fork ~s
```

deadlock free, starvation free and distributed. Briefly explain why your solution is deadlock free and starvation free.

This algorithm is deadlock free because if the a philosopher who has taken a fork release the fork in the other side is not available, will be drop the taken fork and we don't end up in a situation that any of process can't make progress.
This algorithm is also starvation free because the randomization characteristic of this algorithm which breaks symmetry, it means forks are acquired and released in a random order so each philosopher will finally manage to eat. Thus, starvation does not occur.

Mutual Exclusion and Election

✦ **Define the terms "Mutual exclusion" and "election".**

Mutual exclusion: manages access to a shared resources in the way that at most one process at the time can access to it.

Election: are used for choosing a unique process that will coordinate an activity. At the end of the election algorithm, all nodes should uniquely identify the coordinator.

✦ **Any algorithm for mutual exclusion must fulfill two goals: safety and liveness. Define them.**

Safety: At most one process may execute in critical section at any time

Liveness: Every request for a critical section is eventually granted.
Ordering(desirable): Request are granted in the order they ere made (fairness)

- **Mutex algorithms are classified into two categories. which one? define them**

Permission-based approaches: A process, which wants to access a shared resource, requests the permission from one or more coordinators.

Token-based approaches: Each shared resources has a token and a process can access the resource if it has the token.

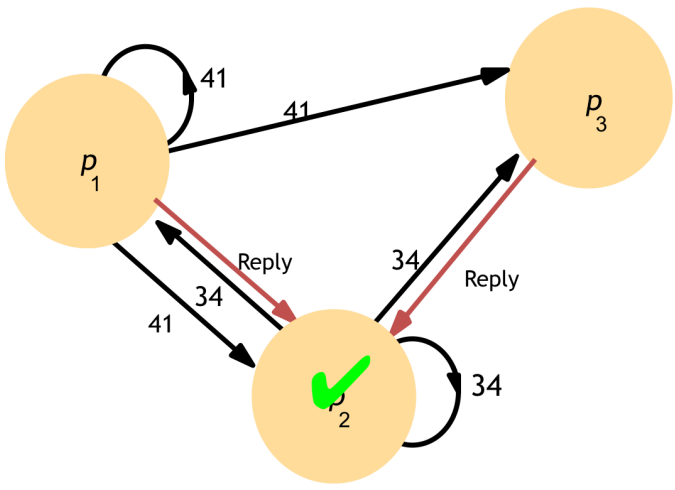
- **There are 3 types of permission-based approach, name and define them and name advantages and disadvantages.**

Centralized: One process is elected as coordinator and maintain a queue of access requests. The coordinator reply to a request only if the resource is not taken by another process.
+ Simple to implement
- Single-point of failure / Bottleneck in the coordinator

Decentralized: A processes which wants to access the resource, it will have to get a majority vote from more than half of the coordinators.

Distributed: 1- Each node broadcast a timestamped request to all other nodes. 2- Nodes send ACK if they don't want to enter CS or trying to enter CS (but their timestamps is larger than the sender's timestamps). If the node is already in CS, then buffer the request. 3- Enter CS when receive ACK from all. (Ricart & agrawala)

1. Broadcast a timestamped ***request*** to all.
2. Upon receiving a request, send ***ack*** if
   -You do not want to enter your CS, or
   -You are trying to enter your CS, but your timestamp is **larger** than that of the sender.
   (If you are already in CS, then buffer the request)
3. **Enter CS**, when you receive ***ack*** *from all*.
4. Upon **exit from CS**, send ***ack*** to each **pending request before making a new request.**



✦ **Describe how the Token ring algorithm works.**

All processes form a logical ring where each process knows its next process. Only the process that has the token can access the resources and pass it to the next process when it is done.

Algorithm	Delay before a process can access the resource (in message times)	Number of messages required for a process to access and release the shared resource	Problems
Centralized			
Decentralized			
Distributed			
Token Ring			

✦ **List limitation of token ring design**

If the token is lost, it must be regenerated.
Dead processes must be purged from the ring
Has a high-message overhead. (The token must circulate in the ring even if only two process need to access the resource.)

✦ **Name and describe two election algorithm.**

Bully algorithm:

1. P_i sends an “Election” message to all other processes with higher ID
2. When P_j with higher ID receives the message, it response with a “Take-over” message. and re-initiate the algorithm
3. If no one respons, P_i wins the election and send “Coordinator” message to every process.

Ring algorithm:

1. When node P_i release the leader has died, start the algorithm. It build a "election" message, append its id to the next node in the ring.
2. Each node that receives this message, append its id to the message and send it to the next node.
3. Eventually the node who started the algorithm gets election message back and elect the node with highest id as leader. Then the P_i node circulates a "coordination" message to announce the leader.

✦ **Compare two election algorithms.**

Algorithm	Number of Messages for Electing a Coordinator	Problems
Bully Algorithm	$O(n^2)$	• Large message overhead
Ring Algorithm	2n-1	• An overlay ring topology is necessary

Ring algorithm is more efficient and put stronger assumptions/requirements on the system(ring topology)

Naming

✦ **Define the terms "naming" and "name resolution" in Distributed Systems in your own words. Be brief and precise.**

Naming is the act of giving objects unique identifiers.
Name resolution is the process of looking up a name

✦ **We discussed two concepts for name resolution: Interactive and recursive. Briefly describe each concept and highlight their key differences.**

Interactive

1. Client hands over the complete name to *root name server*
2. Root name server resolves the name as far as it can, and returns the result to the client
3. The root name server returns the address of the next-level name server (say, NLNS) if address is not completely resolved
4. Client passes the unresolved part of the name to the NLNS

5. NLNS resolves the name as far as it can, and returns the result to the client (and probably its next-level name server)
6. The process continues till the full name is resolved

Recursive:
1. Client provides the name to the root name server
2. The root name server passes the result to the next name server it finds
3. The process continues till the name is fully resolved

✦ **Chord is a DHT. What topology do the nodes form?**

Ring topology

✦ **What operation does a DHT provide?**

put() / get()

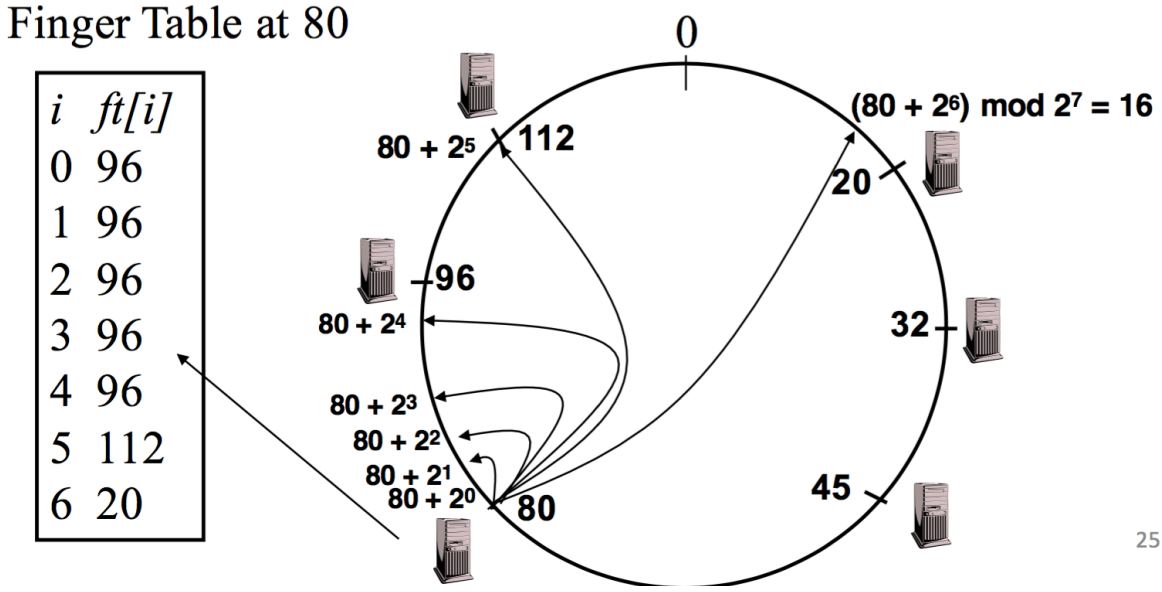
✦ **How is redundancy in Chord achieved?**

Each node keeps the backup of the data of its neighbour nodes.

✦ **In chord, how many hops does it take on average to lookup a data item?**

lookup takes O(log n)

✦ **How finger table works**



Time and synchronization

- **Why is clock synchronization important in DS that rely on physical clocks?**

All events are ordered according to the time given from the physical clock. Without synchronization of clock in the system, node can not determine the ordering of events.

- **How do logical clock differ from physical clocks.**

Logical clocks only “tick” on events, it means time is event-based. In physical clocks time is continuous.

- **Note two types of logical clocks**

Lamport clock and vector clock

- **When do logical clock “tick”, when are they incremented?**

Logical clock are incremented upon events in the algorithm, for example when a message are sent or receive.

- **When a node send a message to another node, what does it do with its logical clock?**

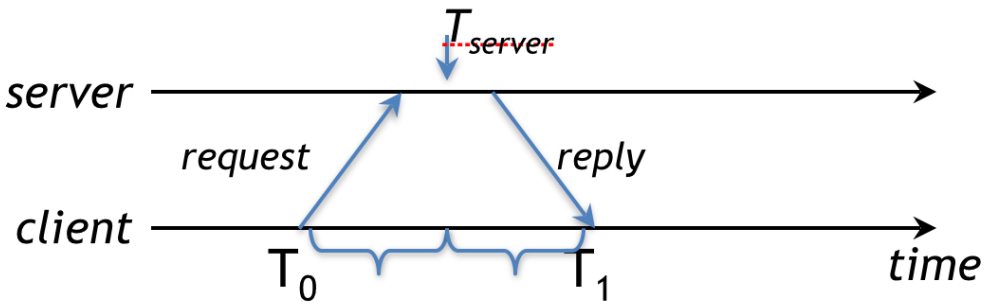
Reads its logical clock, and attach it as a timestamp to the sending message.

- **One algorithm for clock synchronization is called Cristian’s algorithm. Please describe these. Please also note the assumptions this algorithm makes and note the equation the client uses the set it new time.**

Main assumption in this algorithm is “delays are symmetric” which is not 100% correct but often close to reality.
In this algorithm clients compensate for delays. They estimate delay per direction of exchanging message with clock server and add it the timestamp they have received from server.

Be aware that adjustment of system time does not mean set the clock forward or backward. It means we go for a gradual clock correction. If our clock was faster than server’s clock we make clock run slower until it synchronize and if our clock is slower than the clock server make clock run faster until it synchronizes.

With error bounds (Network card delays, physical distans to server divided to light speed and so on) remove from equation we can get even better estimation of delays.



$$d = \frac{T_1 - T_0}{2}$$ d: estimated delay per direction

$$T_{new} = T_{server} + d = T_{server} + \frac{T_1 - T_0}{2}$$

38

- **Assume we need to synchronize the physical clock of n nodes but do not have access to a reference clock. which algorithm would you choose and why? Briefly illustrate that algorithm.**

Berkley Algorithm because it doesn't need access to a reference clock rather tries to synchronize all node's clock to average.
One machine must be elected as the server(leader). Leader pools each machine periodically and when results are in, compute a fault-tolerant average(Server will ignore readings from clocks whose skew is too great.) send offset by which each clock needs adjustment to each slave.

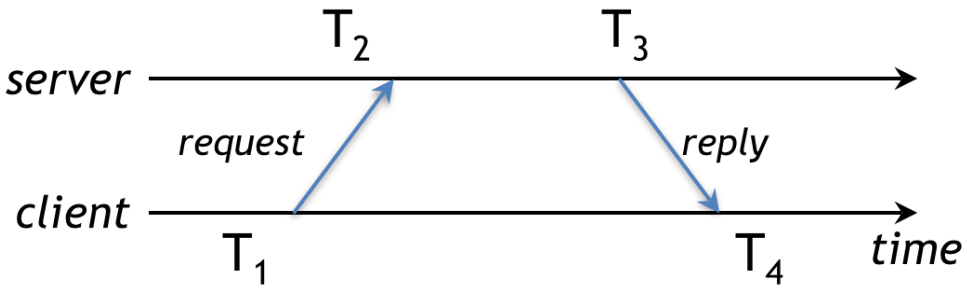
- **Assume we need to synchronize the physical clock of n nodes to a reference clock. which algorithm would you choose and why? Briefly illustrate that algorithm.**

Cristian's algorithm or NTP algorithm. We see Cristian algorithm above. Now we discuss SNTP algorithm.
SNTP is a modification of Cristian algorithm. Machines is arranged in strata (layers). SNTP is recommended for environments where server is root node and client is leaf of synchronization subset. All message delivered with UDP because we don't want re-transmitting which performs by tcp.

- **Why we need logical clock ?**

In reality, we do not know if T_c occurred before T_a and T_b, because in an asynchronous distributed system clocks are not sync !

- **Explain what is causal event ordering?**



$$delay = (T_4 - T_1) - (T_3 - T_2)$$

$$time\_offset = \frac{(T_2 - T_1) + (T_3 - T_4)}{2}$$

57

In an asynchronous DS there is no notion of global clock. Since there is no “real” order we can talk only about casual event ordering. that is one event affects the outcome of another event. For example receive(m) can not have occurred before send(m)
e_i = send(m) and e_j=receive(m) then e_i —> e_j

✦ **Explain concept of consistence and inconsistence cuts.**
Suppose we will constructs a global state of local histories by an external monitor. If that global state which called cut of a distributed computation include events in the way that the history is complete. it is a consistence cut. Otherwise not.

✦ **Define the term consistence cut**
A cut c is consistent if for any event e included in the cut, any event e’ that causally precedes e is also included in that cut.
(e € C) ^(e’ —> e) => e’ € C

✦ **Describe two methods of monitoring.**
We can do it in two ways: Active or Passive monitoring
Passive: Each process p_i sends to p_0(observer) a timestamp of each event it executes.
Active: When P_0(observer) desires to find out the state of the system it asks all processes p_i to send its state.

✦ **In absence of a Real Global Clock how can we know the casual ordering of events?**

We will construct our global clock. It can be done in two ways: Logical (lamport) clocks and vector clocks

✦ **Describe lamport clock**

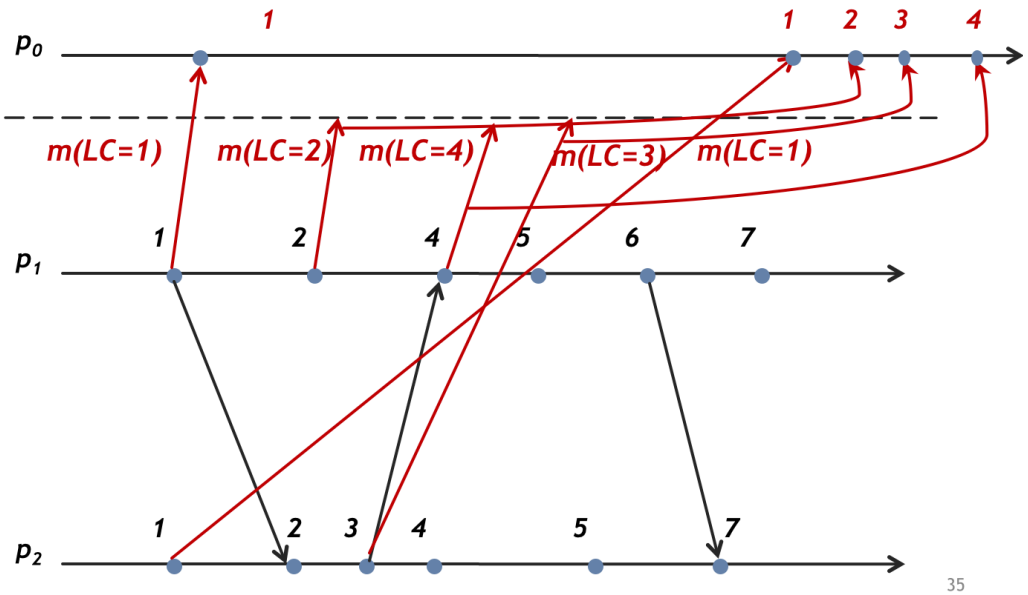
Logical clock of process p counts how many events in a distributed computation casually preceded the current event at p.
Each message m that is sent contains a timestamp TS(m) which is associated with sending event at the sending process.
When the receiving process receives message m, it updates its clock to:
max{ LC, TS(m)} + 1. It means max of local and received value and inc. by one.

✦ **What is causal delivery rule and what is its two rules ?**

Causal delivery rule: Event notification messages are delivered to P_0 in the increasing logical clock timestamp order.
Rule #1: FIFO delivery: Do not deliver event notification message from a process P_i unless all messages with smaller timestamps have been delivered.
Rule #2: Do not deliver to P_0 event notification message with a timestamp TS(m) unless P_0 received messages with timestamps one smaller, equal or greater than TS(m) from all other processes.

✦ **What is limitation of logical clock ?**

LC can not detect whether two events are casually related or concurrent (concurrent means two event is not casually related)

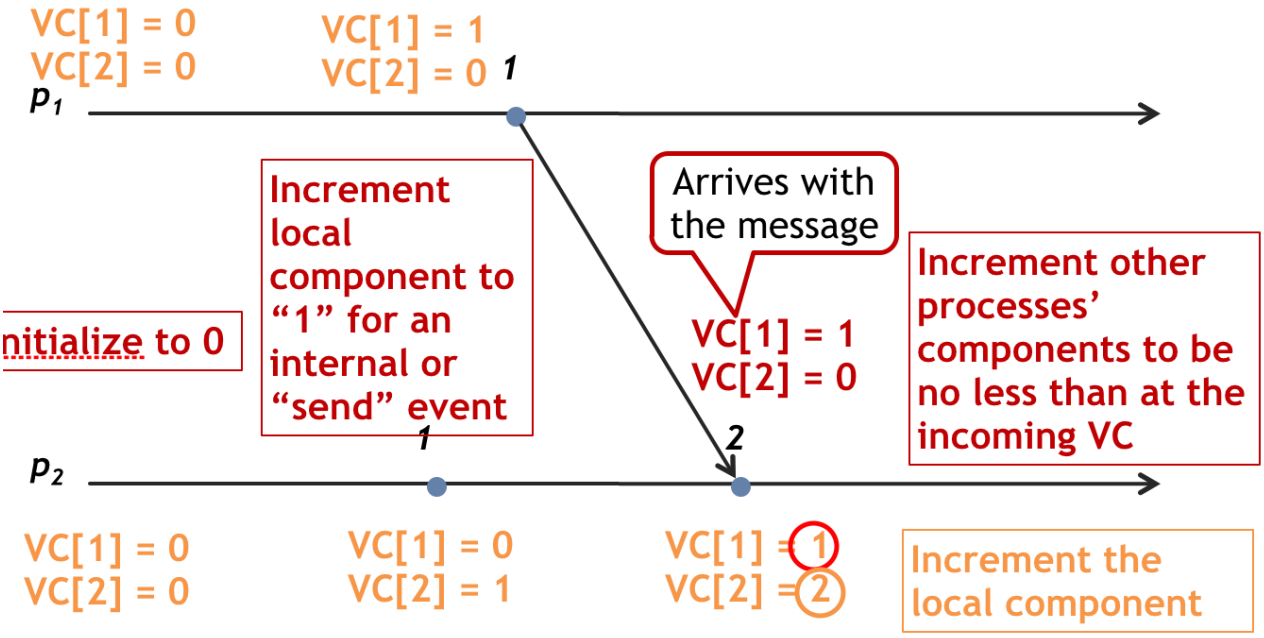


✦ **How many clocks does each node maintain?**

n clocks where n is number of nodes in the DS.

✦ **In vector clock, upon receiving a message, what does each node maintain?**

Inc the local component + Increment other processes' components to be no less than at the incoming VC



✦ **In the course, we discussed how Vector clocks help to distinguish causally related events and concurrent events. Explain how this can be achieved**

Assume two vector clocks on two processes as V1(c1, c2) and V2(c1,c2)
if (V1(c1) < V2(c1) && V1(c2) > V2(c2)) II (V1(c1) > V2(c1) && V1(c2) < V2(c2))
then the two events are concurrent, otherwise they are casually related.

Consistency and Replication

✦ **Define the term replication.**

Replication is a process of maintaining the data in multiple computers.

✦ **List benefits that replication provides. Explain them**

1. Improving performance: A client can access to a data that is near to its location like cashing
2. Increasing availability: can mask failure like in google file system
3. Enhancing the scalability: helps in avoiding bottle-necks at the main server
4. Securing against malicious attack: Number of servers with correct data outvote the faulty servers.

✦ **We discussed the concepts of Total ordering, sequential ordering and causal ordering. Briefly explain each concept.**

Total ordering: Same order of message on all nodes
If process P_i sends a message m_i and P_j sends m_j, and if one correct process delivers m_i before m_j then every correct process delivers m_i before m_j

Sequential ordering: Same order on all nodes & Same order as sent by the sender
If a process **Pi** sends a sequence of messages $m_{(i,1)},....,m_{(i,ni)}$, and Process **Pj** sends a sequence of messages $m_{(j,1)},....,m_{(j,nj)}$, Then, : At any process, the set of messages received are in the same sequential order

Causal ordering: Same order on all nodes only for causally related messages. Writes that are potentially causally related must be seen by all the processes in the same order. Concurrent writes may be seen in a different order on different machines

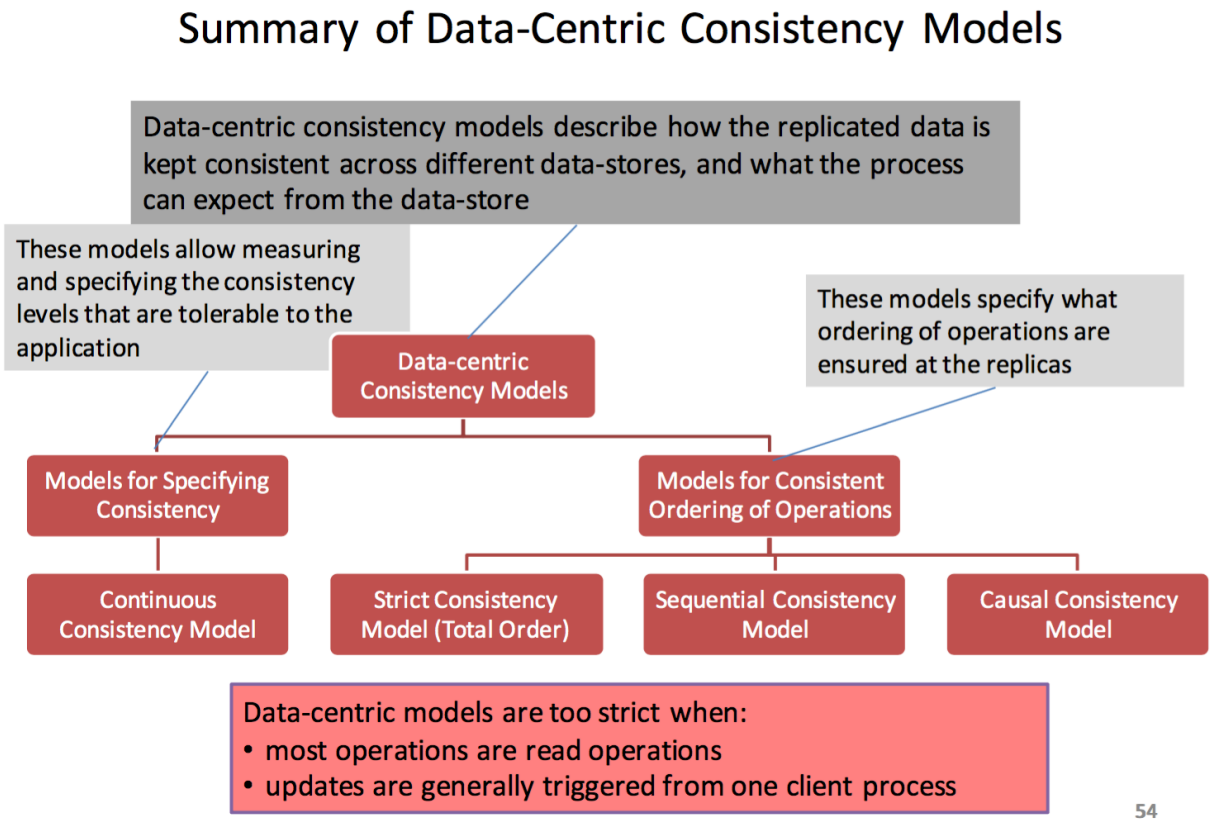
If process P_i sends a message m_i and P_j sends m_j, and if $m_i -> m_j$ (operator '->' is **happened-before** relation) then any correct process that delivers m_j will deliver m_i before m_j

✦ **Metrics to measure consistency?**
Numerical deviation: Deviation in the numerical values between replicas
Order deviation: Deviation with respect to the order of update operation
Staleness deviation: Deviation in the staleness between replicas

* Continuous consistency model define level of consistency over this three metrics

✦ **Define Consistency and Eventual Consistency**
Consistency means that one data item shall have the same value in all replicas
Eventual consistency means all replicas gradually become consistent in the absent of updates.

✦ **Explain Monotonic read**
If a client process reads the value of data item x, then any successive read operation by that process should return the same or a more recent value for x.
✳ The writeset in replica2 must include the history of the other replicas



✦ **Explain Monotonic write**

A write operation by a client process on a data item x is completed before any successive write operation on x by the same process

✳ A new write on a replica should wait for all old writes on any replica

✦ **Explain replica management**

Replica management describes where, when and by whom replicas should be placed

✦ **Name different consistency protocol and define them briefly**

1. Primary-based protocols: One primary coordinator is elected to control replication across multiple replicas
2. Replicated-write protocols: Multiple replicas coordinate to provide consistency
3. Cach-coherence protocol: client controlled replication

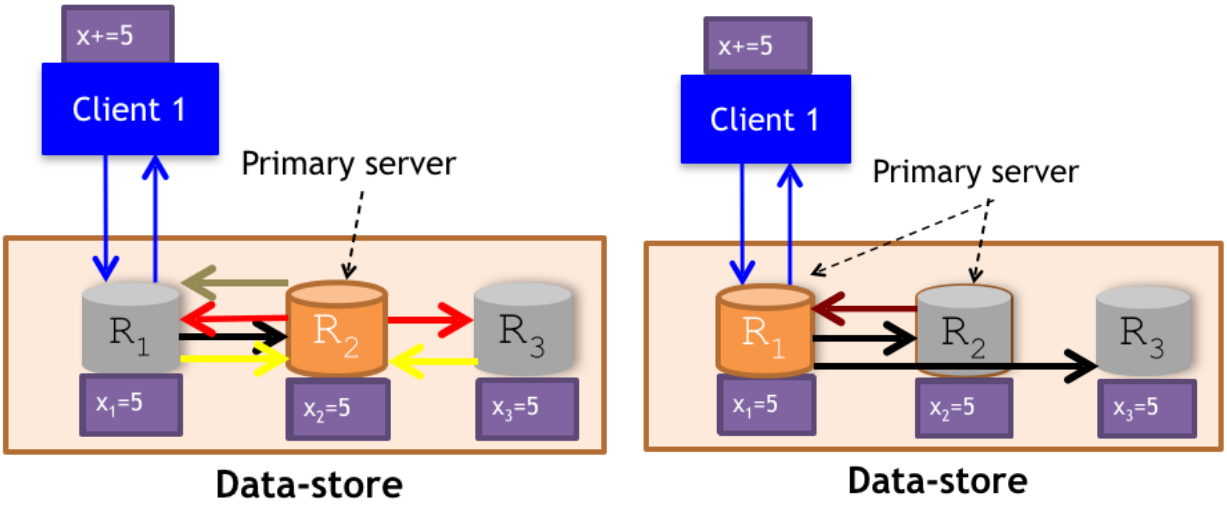
✦ **Name two Primary-based protocol that implement the sequential consistency model**

Remote-write protocol:

+ simple implementation
+ Guarantees that client see the most recent write
- Latency, bottleneck, overhead

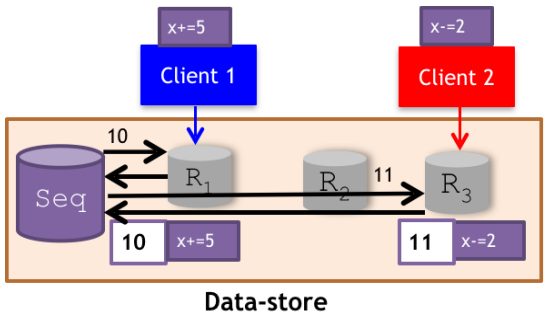
* Good for multiple writing clients & flight booking service

Local-write protocol:
+ Reduce overhead but - not suitable when client are writing at multiple replicas
* Good for single writing client & for dropbox-like service



✦ **Name Replicated-Write protocols. explain it**

Active Replication procotol: Here, we don't have a primary replica. a client can write at any replica. The replica will propogate updates to other replicas. But obviously this protocol can guarantee ordering of operation across the replicas. Fixing this problem could be achieved by using a centralized sequencer(nummerlap in Sweden). When a client connect to a replica and issues a write operation, first get a sequence number from sequencer then propagates the operation and the seqnum together.
* This protocol allows more read per second in compare with Quorum because client can read from any replica



Quorum-based Protocol: Each read and write operation needs to get majority vote of replica. But how many replicas shall the message be written or read to get majority. Read operation (quorum) Nr and write quorum Nw in N replicas:

1- Nr + Nw > N 2- Nw > N/2

Pros: Fully distributed protocol, no single point of failure Cons: complex to implement

* This protocol allows more read per second in compare with Active because write “to write” quorum size which is less than all replicas to which I have to write when using active replication.

✦ **What is cache coherence mean?**
The consistency of data stored in caches

Fault torarent

✦ **Define the term fault tolerance?**
Fault tolerance is the property that enables a system to continue operating property in the event of failures.
A system is said to be fault tolerant if it can provide its services even in the presence of faults.

✦ **List the failure models and explain them**
Crash failure: A server halts
Omission failure: A server fails to respond to incoming requests
Timing failure: A server’s response lies outside the specific time interval
Response failure: A server’s response is incorrect
Byzantine failure: A server may produce arbitrary responses at arbitrary times.

✦ **List redundancy examples**
Hardware: extra equipment (e.x power supplies)
Software: extra processes (ex same software concurrent on multiple systems)
Time: An action perform multiple
Information: extra bit (link layer in wireless communication)

✦ **What is the idea behind process resilience**
The key approach to tolerating a faulty process is to organise several identical processes into a “group”. If one process in a group fails, hopefully some other process can take over.

✦ **What is the goal of the fault tolerant in a faulty system**
Have all non-faulty processes reach consensus on some issue and establish that consensus within a finite number of steps.

A DS can just tolerance fault in a subset of scenarios

- **In "Byzantine Generals Problem" there are honest generals and traitors. What is the goal of the honest and goal of traitors?**

The goal of the honest generals is to come to a consensus, they want to all general either attack or wait but not some of them attack and some of them wait.
The goal of the traitors is to confuse general by sending different message to generals to some of them attack and some of them wait.

- **How an algorithm with multiple phases to enable consensus among the generals. Explain the algorithm and its different phases.**

The basic idea is 1- generals send votes to each other and 2- exchange what each general got from the others
1- All generals vote on a decision, they send this decision to all other generals.
2,3- When a general have received messages from all generals he sends those decision-vectors to all generals.
4- When a general has received all decision-vectors, he checks the first column and chooses any value has a majority, otherwise writes UNKNOWN. He continues with the second column and so forth.
and in that way all generals come to a same decision.

- **How many honest generals are required ?**

In a system with K faulty processes, an agreement can be achieved only if 2k+1 correctly functioning processes are present, for a total of 3k+1

- **There is two policies for failure detection. which ones?**

1- Processes ACTIVELY ping each other
2- Processes PASSIVELY wait until messages come in from other processes.

- **Reliable communication can separate to two parts, Request-Reply communication and reliable group communication. Explain different ways to provide different delivery guarantees.**

1- Retry request message: Controls whether to retransmit the request message until either a reply is received or the server is assumed to have failed.
2- Duplicate filtering: Controls when retransmissions are used and whether to filter out duplicate requests at the server
3- Retransmission of results: Controls whether to keep a history of result messages to enable lost results to be retransmitted without re-executing the operations at the server.

- **What is Orphan means in request reply context and which problem can they cause ?**

A client might crash while the server performing a corresponding computation. Such a unwanted computation called orphan can cause waste cpu cycle and resource locking.

- **We discussed the "Two phase commit" protocol and its two phases. Please name and describe phase 1**

The coordinator sends out a commit-ready message to all clients. clients can reply either with "ready" or "abort"

- **Please describe what happens during a failure in phase 1**

If a node fails and does not reply, then a timeout is reached and ABORT is chosen

- **Please name and describe phase 2**

  The coordinator collects all votes from the participants. If all participants have voted to commit, then it sends a $GLOBAL\_COMMIT$ message to all participants. If one participant had voted to abort, the coordinator will also decide to abort the transaction and multicasts a $GLOBAL..ABORT$ message.

- **Please describe what happens during a failure in phase 2**

  The simplest solution to this problem is to let each participant block until the coordinator recovers again.

  A better solution is to let a participant P contact another participant Q to see if it can decide from Q's current state what it should do.

- **Describe two general problems**

The Two Generals Problem

Two generals lead divisions of an army camped on the mountains on the two sides of an enemy-occupied valley

The two divisions can only communicate via messengers

We need a scheme for the generals to agree on a common attack time, given that attack by only one division would be disastrous

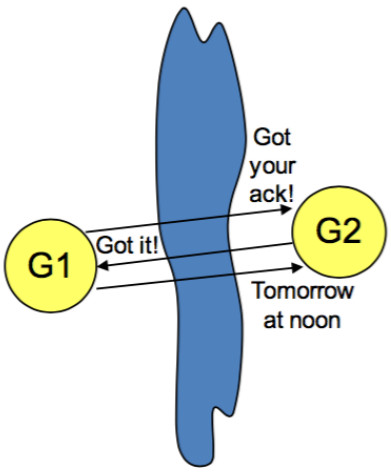
Messengers are totally reliable, but may need an arbitrary amount of time to cross the valley (they may even be captured and never arrive)

G1 decides on *T*, sends a messenger to tell G2

G2 acknowledges receipt of the attack time *T*

G2, unsure whether G1 got the ack (without which he would not attack), will need an ack of the ack!

This can go on forever, without either being sure



Applications

✦ **Define CAN (content addressable network)**

One type of DHTs. Each node and each data item has a unique id in an d-dimensional space on a d-torus.

* Routing table size in O(d) / item querying d*(n^1/d) [n is #nodes]
* DHTs at general is good to put and to get items (through the interface they provided for applications) but not efficient for search
* Another DHTs called chord. Routing table size in O(log n) / item querying O(logn)

✦ **How joining occurs in DHT - CAN**

1- A new node must discover some node “I” already in CAN
2- Pick random point in space
3- The existing node “I” route to (x,y) and discover responsible node “J” around (x,y)
4- Split J’s zone in half … new node owns one half

✦ **How departure shall occurs ?**

Controlled leave: hands over its zone and associated (key, value) to one of its neighbours
Sudden leave: Ping all neighbours + backup its neighbour

✦ **Explain the basic concept of Bittorrent. Seed, Leech, Tracker, Swarm**

Seed: peer with the entire file
Leech: peer that is downloading the file
Tracker: A service which keeps rack of all peers that are downloading file. .torrent file has the address of tracker
Swarm: Set of peers all downloading the same file
Sub-piece: Further subdivision of a piece. init for request

✦ **In peer-peer transaction how choosing pieces to request ?**

Rarest-first / Random first piece / End-game mode

✦ **What is free riding and how we can avoid it**

Free riding is a situation when a client just download pieces of a data file and doesn’t upload.
We fix this problem with tit-for-tat algorithm. It means we optimistic unchoke nodes which means offer pieces to download, if the node upload back to other nodes we offer more pieces to download.

✦ **Modern bitTorrent can work without trackers. This mode is called "trackerless" BitTorrent. Explain this mode and list of benefit.**

It means instead of a centralized-tracker we use a DHT to keep "tracker file name" and "tracker file" as (key, value).
+ remove single point of failure + prevent pollution attacks

✦ **Explain how TOR provides anonymous Internet browsing. You can draw a figure to illustrate your argumentation**

A client which want to connect to a service by TOR might build a TOR circuit(anonymous tunnel between client and internet). Initiatally establishes a symmetric session key circuit with relay node #1. Then with relay node #2 through the tunnel relay node #1. Finally to the service and connect and communicate over the established TOR circuit. Datagrams decrypted and re-encrypted at each link.

✦ **TOR also allows so called hidden services. Please explain what a hidden service is and how TOR enables it.**

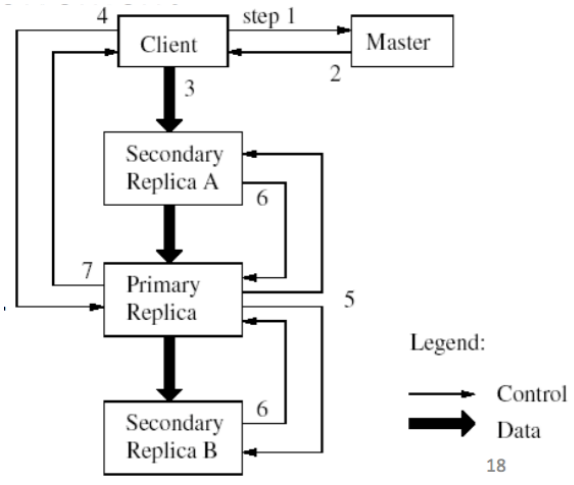
Hidden service means deploy a server on the internet that anyone can connect to without knowing its IP address.
Clients who want to connect to this service must connect to a rendezvous point at the beginning and this point through tunnels of a bunch of other relays node connect clients to the service.

✦ **How GFS designed**

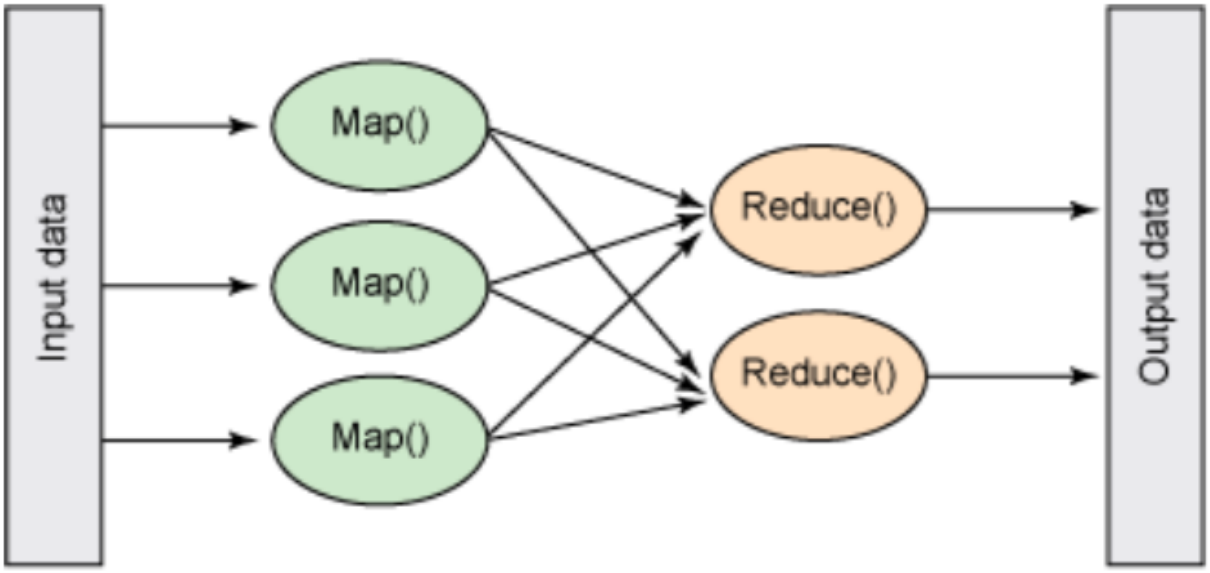
Files stores as fix sized chunks. Each chunks replicated across 3+ chunk-server and single master coordinate them and keep metadata. Single master causes single point of failure but google shadow masters and minimize master involvement to move data.
Metadata include: 1- file and chunk namespaces
2- Mapping from files to chunks
3- Locations of each chunk's replicas

✦ **How Mutation works**

Mutation means write or append a variable that must be done for all replicas. Client contact to master and get address of a primary replica, then contact primary replica to push the changes and primary replica replicates to other replica. When primary get ACK of others then ping back to client and stating the operation is done



✦ **What is master responsibilities in**



GFS?

- metadata storage
- Namespace management
- Locking (mutex)
- periodic communication with chunk-servers
- Stale replica deletion
- Garbage collection: There is no delete operation directly. If master find out there is no reference to the chunk anymore, will delete it.

✦ **Describe the MapReduce algorithm. Split it into it phases. For each phase include: the name of the phase, what this phase does, and who is responsible for its implementation(mapreduce framework or programmer)**

MapReduce is fast distributed data processing. It has 4 phases:
(Input data is split into M parts)
1- Map: extract information on each part [programmer]
2- Shuffle and 3-sort [framework]
4- Reduce which include aggregate, summarize, filter or transform on extracted info [programmer]

* Framework provides even load balancing, pipelining, restart of failed mappers and reducer. Unlike GFS, Mapreduce doesn't have shadow Master, if master which control workers(computers) fails whole of the process might be do again from the beginning.