DsExam_Sample_Questions

Basics about DS

- **Define the term "Distributed System". Be brief and precise.**
  - Multiple Devices
  - Connected by a network
  - Cooperating on some task.
- **The following statement is attributed to Thomas J. Watson, Chairman and CEO of International Business Machines (IBM), in 1943: "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.**
  - If this sentence had been the correct vision, then we would not see such a vast distributed system that exists today.
- **When designing or developing a distributed system, its distributed nature presents a number of challenges, which we discussed in the course. List at least six such challenges.**
  - Network reliability
  - Latency
  - Bandwidth
  - Network security
  - Transport cost
  - Topology change
  - Homogenous network

- The centralized algorithm, decentralized algorithm, and distributed algorithm .
  - **Please define the terms centralized algorithm, decentralized algorithm, and distributed algorithm.**
    - Centralized algorithm:
      - One process is elected as a coordinator for a shared resource.
      - Coordinator maintains a Queue of access requests.
      - Example Application: Banking Service
    - Decentralized Algorithm:
      - Multiple coordinators each with replica of resources.
      - Requester Queries m coordinators.
      - Example Application: Google Search Engine
    - Distributed Algorithm:
      - Every node will be identical.
      - Example: Ricart & Agrawala's Algorithm, **BitTorrent**
  - **Briefly list the key differences between these three.**
    - One coordinator
    - Multiple coordinator
    - No coordinator
  - **Briefly, list two advantages of each approach.**
    - Centralize
      - Easy to implement
      - Easy to debug
    - Decentralized Algorithm:
      - More Robust than centralize
      - Fault tolerant
    - Distributed Algorithm:
      - Most Robust
      - Scalable

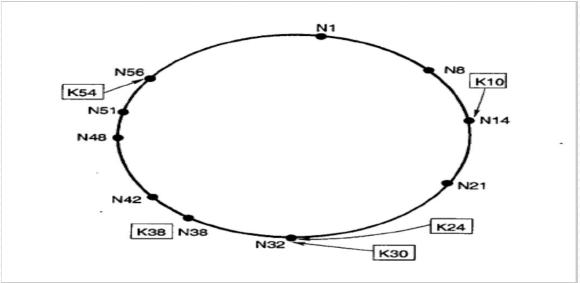
- Peer-to-Peer Architectures vs. Client-Server.
  - **Please explain and summarize the key differences of the two approaches.**
    - Single Centralize coordinator node with multiple client node in client- server.
    - Every node has identical role.
  - **Are Client-Server Architectures commonly designed for a centralized algorithm decentralized algorithm, or distributed algorithm?**
    - Client - server use centralize algorithm
  - Are Peer-to-Peer Architectures commonly designed for a centralized algorithm decentralized algorithm, or distributed algorithm?
    - Peer to peer use decentralize or distributed algorithm
- **Ethical challenges: Certain distributed systems such as BitTorrent, Bitcoin and TOR trigger ethical challenges. Please select two of these systems and for each of them please discuss two ethical challenges.**
  - Bittorrent
    - Illegal file transfer
    - No central control
  - Bitcoin
    - No governmental entity
    - Illegal untraceable payment
  - Tor
    - Conceal interaction with gambling, drug, … sites
    - Illegal hidden services
    - Bypass internet cencorship

- **When designing or developing a Distributed System, its distributed nature presents a number of challenges. List at least 4 challenges.**
  - Network reliability
  - Latency
  - Bandwidth
  - Network security
  - Transport cost
  - Topology change
  - Homogenous network
- **Select two challenges you listed in 1b) and detail on them. Answer the following two questions for each of them: Why do these make developing Distributed Systems challenging? What mechanisms did we study in the course to deal with them? Be brief and precise**
  - Latency
    - We face different latency issues in different design aspect in distributed system for large number of clients across the internet
      - For solving this we studied **Consistency and replication (cdn) Mechanism**
  - Network reliability
    - Within internet, communication might not be reliable but user expect desired distributed services to be working properly for example server(crash) may not be reachable , or too many concurrent request
    - For solving this we studied **Fault tolerant and Mutual exclusion (election algorithm) mechanism**

- (4 points) **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 in keywords.**
  - Application Layer
    - Collection of application-specific Protocols.
    - Email(SMTP, POP, IMAP), File Transfer(FTP), Directory Services (LDAP)
  - Presentation Layer
    - Data Representation
    - Concerned with the meaning of data bits.
    - Convert between machine representations.
    - XDR, ASN.1, MIME, MIDI.
  - Session Layer
    - Services to coordinate dialogue and manage data exchange.
    - Establish and end communications.
    - Manage multiple logical connections.
    - HTTP, HTTPS, SSL, NetBIOS
  - Transport Layer
    - Provides a consistent interface for end-to-end communication
    - Congestion, flow control.
    - TCP, UDP
  - Network Layer
    - Relay and route Information to destination.
    - Manage journey of packets and figure out intermediate hops.
    - IP, X.25
  - Data Link Layer
    - Detect, Correct errors.
    - Organizes data into packets before passing it down.
    - Sequences packets.
    - MAC, PPP
  - Physical Layer
    - Transmits and receives raw data to communication medium.
    - Does not care about contents.
    - Voltage levels, speed, connectors, modulation.
    - RS-232, 10BaseT.

Naming

- **Distributed Hash Tables (DHTs): Below you find a picture of a Chord ring, with nodes N1, N8, N14 etc.**
  - **Please list the finger table of node N8, i.e., list to which nodes the figures point and explain your reasoning and calculations. Note: in this example, the finger table size is 6.**
    - Answer is given Below of this slide.
  - **Assume that node N8 aims to lookup the data item K54, stored on node N56. How is this lookup request routed from N8 to N56, the node storing data item K54? Please explain your reasoning.**
    - Key 54 is greater than last value N42 of Finger table in N8, So, Move to N42
    - From N42, Move to N51, (We have to find such a I, where Fp[i] <= Key < Fp[i+1]
    - From N51, Move to N56 to get K54
  - **What are the key differences between the Chord and CAN DHTs ?**
    - CAN
      - A unique id in an d-dimensional Cartesian space on a d-torus.
      - Item is found in at most d * power(n, 1/d)
    - CHORD
      - Nodes are structured in a ring.
      - Item is found in at most log2(n).

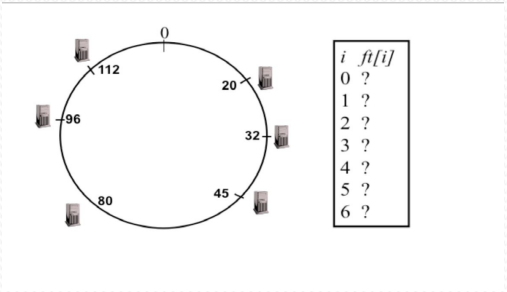


I (N8)	(fp(I) = P + 2^(I -1))
1	14
2	14
3	14
4	21
5	32
6	42

I(N42)	(fp(I) = P + 2^(I -1))
1	48
2	48
3	48
4	51
5	1
6	14

- Amazon Dynamo, a (key, value) store for data centers. Dynamo is a modified version of the Chord protocol.
  - **What are the key differences? And Why?**
    - Chord
      - Routing (also finger) table size: log(n)
        - Scales to millions of nodes
      - Lookup: use finger table for log(n) hops
    - Dynamo
      - Routing table size: n
        - Fully meshed
        - Scales to thousands of nodes
        - Perfect for data centers
      - Lookup: one hop
        - Strong performance
  - **Dynamo introduces the concept of virtual nodes, what are they used for? (VVP)**
    - To make load distribution easier.
  - **How are events in Dynamo timestamped?**
    - Vector Clocks
  - **What consistency model does Dynamo use? And why?**
    - Eventual consistency
      - Multiple versions of the same object might co-exist.
  - **How are conflicts resolved in Dynamo? During which operation? And When?**
    - Syntactic Reconciliation
      - System might be able to resolve conflicts
      - Automatically: using timestamp
    - Semantic Reconciliation
      - Conflict resolution pushed to application.

- **Define the terms "naming" and "name resolution" in Distributed Systems in your own words. Be brief and precise.**
  - Naming: Names are used to uniquely identify entities in Distributed Systems. Entities may be processes, remote objects, newsgroups. Names are mapped to an entity's location using a name resolution.
  - Name Resolution:
    - For example a name can be: http://abc.com:8974/sample/index.html
    - DNS Lookup can return: 55.55.55.55
    - Resource ID(IP Address, Port, File Path) : (55.55.55.55, 8974, sample/index.html)
    - Mac Address (02.60.8c.02.b0.5a)
- **In the lecture we discussed the concept of Chord. Chord is a Distributed Hash Table (DHT).**
  - What operations does a DHT, e.g., Chord, provide?
    - GET and PUT
  - In Chord, how many hops does it take on average to lookup a data item?
    - log2(n)
  - Finger tables in Chord
    - Fp[i] = p + pow(2, i-1)
  - What is the finger table in Chord used for?
    - To lookup



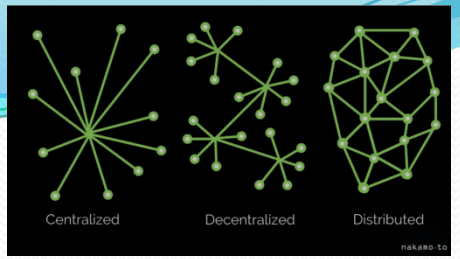
- **Below you find a picture of a Chord ring (on the left) and the finger table of node 80 (on the right). Please complete its finger table, i.e., list to which nodes the fingers point. In this example the finger table size is 7.**

I	Ft[i]
0	96
1	96
2	96
3	96
4	96
5	112
6	20

- **In the course we discussed two concepts for name resolution: Iterative and recursive name resolution. Briefly describe each concept and highlight their key Differences.**
  - Iterative Name Resolution:
    - Clients hands over the complete name to root name server.
    - Root name server resolves the name as far as it can and returns the result to the client
    - Client passes the unresolved part of the name to the NLNS.
    - The process iterates until full name is resolved.
  - Recursive Name Resolution:
    - Client provides the name to the root name server.
    - The root name server passes the result to the next name server it finds.
    - The process continues till the name is fully resolved.

Leader Election and Mutual Exclusion

- Bully Algorithm for Leader Election:
  - **Please describe how the algorithm works**
    1) A process initiates election algorithm when it notices that the existing coordinator is not responding.
    2) Process Pi calls for an election as follows:
       - Pi sends and "Election" message to all processes with higher process IDs.
       - When Process Pj with j>i receives the message, it responds with a "Take-over" message. Pi no more contests in the election. Process Pj re-initiates another call for election. Step 1 and 2 continue;
    3) If no one responds, Pi wins the election. Pi sends "Coordinator: message to every process.
  - **How does the algorithm deal with nodes failing during the election?**
    - If failed node is the highest process ID then immediate lower process ID become next co-ordinator.
    - If failed node is not the highest process ID then bully doesn't care about this node failure.
    - **What message complexity does the algorithm have and why?**
    - Best Case: N-2. The process with the second highest id notices the failure of the coordinator and elects itself.
    - Worst Case: O(N^2). When the process with the lowest id in the system detects the failure, then need to send too many messages.
  - **How does the algorithm ensure safety and liveness?**
    - Safety: After execution of this algorithm either each process selects one particular leader or none.
    - Liveness: Algorithms is re-initiated if coordinator fails.
  - **When you compare both algorithms, what key advantages and disadvantages do your see for each?**
    - Bully Algorithm
      - - Need to send more messages. In worst case O(n * n).
      - + Always maintain liveness even on node crash during the execution of algorithm.
    - Ring Algorithm
      - + Need to send less messages. In worst case O(2*n – 1).
      - - Liveness is not satisfied if one process fails during the execution of algorithm.



- **In the course, we discussed the Ricart & Agrawala algorithm for Mutual Exclusion.**
  - **Please explain this algorithm.**
    1) Broadcast a timestamped request to all.
    2) Upon receiving a request, send ack if
       i. You do not want to enter your Critical Section (CS) or
       ii. 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. (No release message is necessary).
  - **What is the message complexity (please explain)?**
    - 2(n-1) messages per entry operation
    - (n-1) unicasts for the multicast request + (n-1) replies as ack (send over the network)
  - **Any algorithm for mutual exclusion must fulfill two goals: safety and liveness. Explain how the Token Ring algorithm achieves these.**
    - Safety: Only those nodes can get access to the shared resource which has the token.
    - Liveness: Each process will receive the token. This is how it solves starvation. But to meet liveness , a token should be generated when a token is lost. Detecting the loss of token is difficult since the amount of time between successive appearances of the token is unbounded. In my opinion token ring algorithm doesn't fulfill livness property fully.
  - **Is Ricart & Agrawala a centralized, decentralized, or a distributed algorithm (please explain)?**
    - Distributed Algorithm. Because:
      - Centralized Algorithm: There is only one coordinator node that will communicate with other nodes.
      - Decentralized Algorithm: There are several coordinating nodes that will communicate with other client nodes. Client nodes can connect one of these coordinating nodes.
      - Distributed Algorithm: Every nodes are identical, no coordinator or no clients.

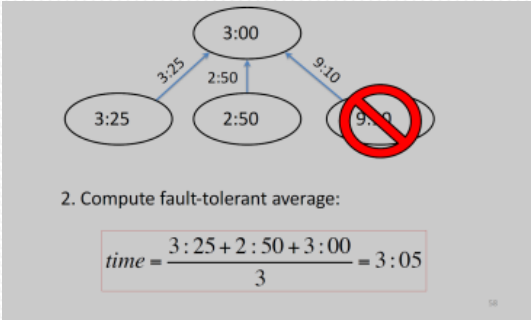
- **What properties should a leader election algorithm in distributed systems have?**
  - **Safety**: সকল নন ফল্টি প্রসেস হয় যেকোন একটা প্রচেস 'ক' কে লিডার বানাবে, না হয় কাউকে বানাবে নাহ।
  - **Liveness**: যদি কোন লিডার হেং , ক্রেশড অথবা হারিয়ে যায় তাহলে এই সমস্যা টি সমাধানের একটা মেকানিজম থাকবে। লিডার যদি কোন একটা মেসেজের রিপ্লাই দিতে নির্দিষ্ট সময়ের বেশি লাগায় তাহলে এল্গরিদম টি আবার অটোমেটিকালি চালু হয়ে যাবে এবং নেটওয়ার্ক এর আরেকটি প্রসেস কে সবাই মিলে সিলেক্ট করে ফেলবে।
  - **Network Bandwith**: প্রত্যেক্টা এল্গরিথম এ কতোগুলা মেসেজ যাবে এইটার একটা লিমিট থাকতে হবে। উদাহরণস্বরুপ, বুলি এল্গরিদম এ মেসেজ সর্বোচ্চ হতে পারে O(n*n) .
- **What challenges does leader election in distributed systems face?**
  - Node can crash anytime.
  - Node can out of the network for some time.
- **Please name two algorithms we described in the course that elect a leader.**
  - Bully Algorithm
  - Ring Algorithm
- **When you compare both algorithms, what key advantages and disadvantages do your see for each?**
  - Bully Algorithm
    - - Need to send more messages. In worst case O(n * n).
    - + Always maintain liveness even on node crash during the execution of algorithm.
  - Ring Algorithm
    - + Need to send less messages. In worst case O(2*n – 1).
    - - Liveness is not satisfied if one process fails during the execution of algorithm.

- **Define the terms "Mutual Exclusion" and "Election" in your own words. Be brief and precise.**
  - Mutual Exclusion:
    - Manage to access to resources
    - Goal: unique access
  - Election: In a group of processes, elect a leader to undertake special tasks.
- **Please describe how the Token Ring algorithm works.**
  - Token Ring Algorithm:
    - Each resource is associated with a token.
    - The token is circulated among the processes
    - The process with the token can access the resource.
    - Circulating the token among processes:
      i. All processes form a logical ring where each process knows its next process.
      ii. One process is given a token to access the resource.
      iii. The process with the token has the right to access the resource.
      iv. If the process has finished accessing the resource OR does not want to access the resource, it passes the token to the next process in the ring.
- **List at least two limitations of its design.**
  - Token ring has a high message overhead. Particularly, when no process need the resource, the token circulates at a high-speed.
  - If the token is lost, it must be regenerated. Detecting the loss of token is difficult since the amount of time between successive appearances of the token is unbounded.
  - Dead processes must be purged from the ring. ACK based token delivery can assist in purging dead processes.

- **We discussed the Ring Algorithm for Election.**
  - **Please describe how the Ring algorithm works. For simplicity, assume that there is only one initiator for the election. Please also note what assumption the algorithm makes on the topology.**
    - Ring Algorithm:
      - This algorithm is generally used in a ring topology.
      - When a process Pi detects that the coordinator has crashed, it initiates an election algorithm.
      - Pi builds an "Election" message €, and sends it to its next node. It insert its ID into the Election message.
      - When process Pj receives the message, it appends its ID and forwards the message. (If next node has crashed, Pj finds the next alive node).
      - When the message gets back to the process that started the election:
        - It elects process with highest ID as coordinator,
        - Changes the message type to "Coordination message and circulates it in the ring.
    - Assumption:
      - No failures happen during the run of the election algorithm.
  - **What is its message complexity? Assume that there are n nodes and one initiator.**
    - 2n - 1

Time and Synchronization

- Assume we need to synchronize the physical clocks of "n" nodes but do not have access to a reference clock.
  - **Which algorithm (that we discussed in the course) would you choose and why?**
    - Berkeley Algorithm
    - Because it obtains average from participating nodes and synchronizes all clocks to average. No need to get access to a reference clock.
  - **Briefly illustrate that algorithm. You can draw a figure to support your argumentation.**
    - Machines run time daemon, process that implements this protocol.
    - One machine is elected as the master server and others are slaves
    - Master polls each machine periodically , ask each machine for time. Can use Cristian's algorithm to compensate for network latency.
    - When results are in, compute average , including master's time.
    - Hope: average cancels out individual clock's tendencies to run fast or slow.
    - Send offset by which each clock needs adjustment to each slave.
    - Algorithm has provisions for ignoring readings from clocks whose skew is larger. So that will compute a fault-tolerant average.
    - If master fails any slave can take over.



  - **Can this algorithm deal with a faulty clock? If so, how does it do this?**
    - Yes. Algorithm has provisions for ignoring reading from clocks whose skew is larger. That computes a fault-tolerant average.
  - **For time synchronization between two nodes, this algorithm relies on another algorithm. Please note which algorithms (that we discussed in the course) it might use and why.**
    - Cristian's algorithm.
    - Because, If any node tries to synchronize time with other nodes, there exists network latency which needs to be kept into consideration. Cristian's algorithm is designed to deal with such problems.

- In the course, we discussed how Vector Clocks help to distinguish causally related events and concurrent events. Below you see pairs of Vector clocks. Note for each pair whether they denote concurrent or causally related events. Briefly explain your reasoning.
  - **Are these two events causally related or concurrent?**

Event on Node 1	Event on Node 2
VC[1]= 4	VC[1]= 2
VC[2]= 2	VC[2]= 10

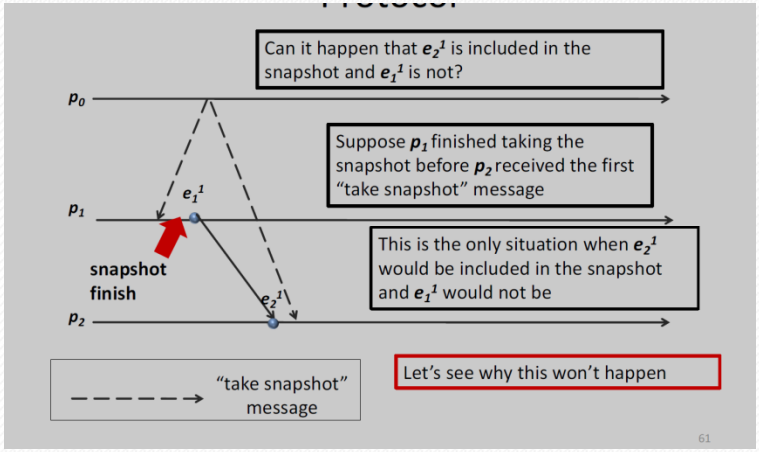
    - Concurrent.
    - Because, VCnode1[1] > VCnode2[1] and VCnode1[2] < VCnode2[2]. We can't be sure which event occurred first. So that's a concurrent event.
  - **Are these two events causally related or concurrent?**

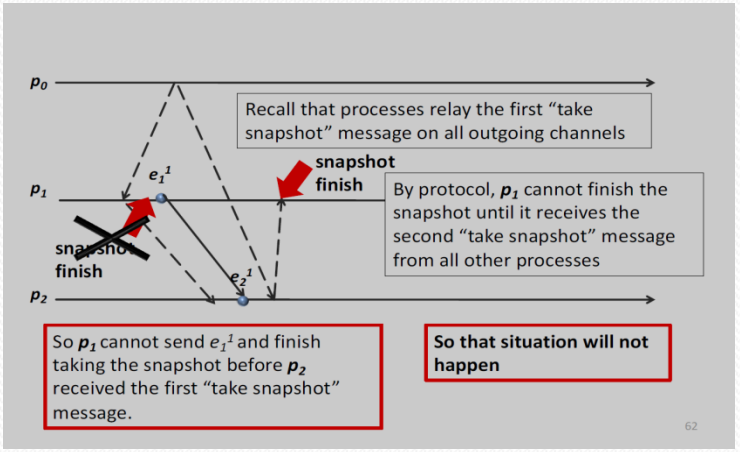
Event on Node 1	Event on Node 2
VC[1]= 4	VC[1]= 5
VC[2]= 2	VC[2]= 4

    - Causally Ordered.
    - Because, each VC[j] in Node 2 is greater or equal each VC[i] in Node 1. So (Event in node1) -> (Event in node2).

- In the course, we discussed a Distributed Snapshot Protocol (Chandy and Lamport).
  - **Please explain this algorithm: How does it construct a snapshot?**
    - Monitor Po sends “take snapshot” message to all processes Pi.
    - If Pi receives such “take snapshot” message for the first time, it:
      - Records its state
      - Stops doing any activity related to the distributed computation.
      - Relays the “take snapshot” message on all of its outgoing channels
      - Starts recording a state of its incoming channels
      - Records all messages that have been delivered after the receipt of the first “take snapshot” message.
    - When Pi receives a “take snapshot” message from process Pj, it stops recording the state of the channel between itself and Pj.
    - When Pi receives “take snapshot” message from all processes and from Po, it stops recording the snapshot and sends it to Po.

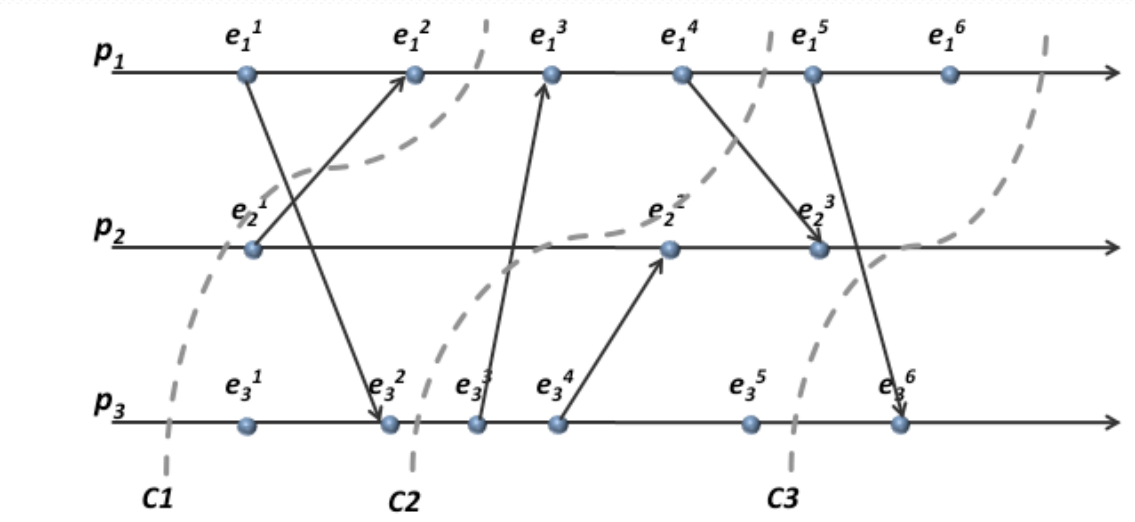
  - **Please explain why this algorithm works correctly, i.e., it does not lead to inconsistent snapshots.**
    - Recall the key property of a consistent snapshot.
      - If event (e -> e”) and e” is included in the snapshot , then e must also be included in that snapshot.





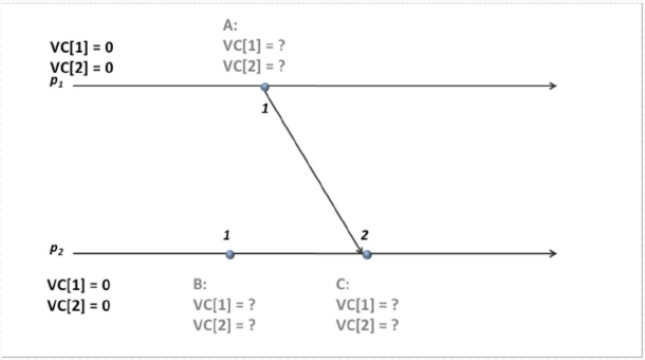
- **Please explain this concept briefly and formally define the term consistent cut.**
  - A cut is consistent if for any event *e* included in the cut, any event *e'* that causally precedes *e* is also included in that cut
  - For cut *C*: $(e \in C) \wedge (e' \rightarrow e) \Rightarrow e' \in C$

- **Please note for each cut (C1, C2, C3) depicted below whether it is a consistent or an inconsistent cut. Explain your reasoning briefly.**
  - C1 : Inconsistent. e(1,2) included in the Cut, but preceding event e(2,1) is not included in the cut.
  - C2: Inconsistent. e(1,3) included in the Cut, but preceding event e(3,3) is not included in the cut.
  - C3: Consistent. Every event e included in the Cut, also every e' such that e'-> e included in the cut.



- In the course, we discussed the concept of clock synchronization for physical clocks. Why is clock synchronization important in Distributed Systems that rely on physical clocks?
  - Write your own opinions.
- Please answer the following questions about logical clocks.
  - **How do logical clocks differ from physical clocks?**
    - Logical clock keeps track of event ordering among related events.
    - Physical clock keep time of day , consistent across systems.
  - **When do logical clocks "tick", i.e., when are they incremented?**
    - Logical clocks "tick" when an event occur.
    - Increment:
      - When a process receive a message from other process.
      - When a process send a message to other process.
      - When a new event occurs in own process.

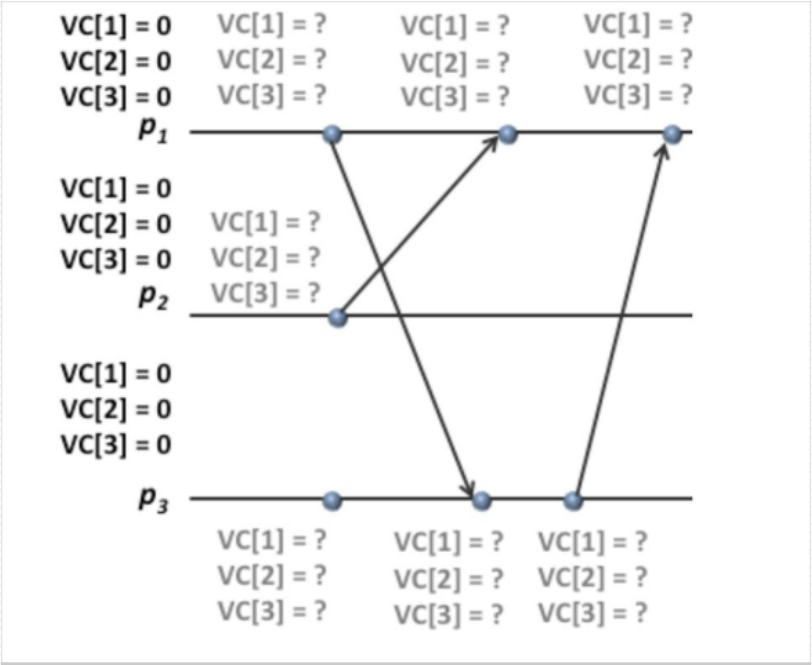
- In the course, we discussed the concept of Vector Clocks. Please answer the following questions about Vector Clocks.
  - **How many clocks does each node maintain?**
    - If we n Process, then each node maintain vector clock of size n.
  - **Upon receiving a message, what does a node do with its vector clock(s)? Below you see a figure of a vector clock for two nodes (p1, p2). All vector clocks are initialized to zero. Please list the vector clocks for events A, B, C.**
    - A: VC[1] = 1, VC[2] = 0
    - B: VC[1] = 0, VC[2] = 1
    - C: VC[1] = 1, VC[2] = 2



- **One algorithm for clock synchronization is called Berkeley Algorithm.**
  - **It operates in three steps. Please describe these.**
    - Master polls each machine periodically, ask each machine for time.
    - When results are in , compute average, including master's time.
    - Send offset by which each clock needs adjustment to each slave.

- **How do the goals of this algorithm differ from other algorithms for clock synchronization such as Christian's Algorithm or the SNTP Algorithm?**
  - Berkeley Algorithm synchronize time among n nodes.
  - Christian's and SNTP algorithm both adjust time between 2 nodes.

- **Below you find a figure of a vector clock for three nodes. All vector clocks are initialized to zero. Please list the vector clocks for each event.**
  - P1:
    - VC[1] = 1, VC[2] = 0, VC[3] = 0
    - VC[1] = 2, VC[2] = 1, VC[3] = 0
    - VC[1] = 3, VC[2] = 1, VC[3] = 3
  - P2:
    - VC[1] = 0, VC[2] = 1, VC[3] = 0
  - P3:
    - VC[1] = 0, VC[2] = 0, VC[3] = 1
    - VC[1] = 1, VC[2] = 0, VC[3] = 2
    - VC[1] = 1, VC[2] = 0, VC[3] = 3



- **Are these two events causally related or concurrent?**

Event on Node 1	Event on Node 2
VC[1]= 1	VC[1]= 0
VC[2]= 0	VC[2]= 6

  - Concurrent
  - Because, VCnode1[1] > VCnode2[1] and VCnode1[2] < VCnode2[2]. That violates causal relation.

- **Are these two events causally related or concurrent?**

Event on Node 1	Event on Node 2
VC[1]= 1	VC[1]= 1
VC[2]= 0	VC[2]= 6

  - Causal
  - Because, each V[i] in node 2 is greater or equal than each V[i] in node 1

- **Are these two events causally related or concurrent?**

Event on Node 1	Event on Node 2
VC[1]= 1	VC[1]= 0
VC[2]= 0	VC[2]= 6

  - Concurrent
  - Because, VCnode1[1] > VCnode2[1] and VCnode1[2] < VCnode2[2]. That violates causal relation.

- **Are these two events causally related or concurrent?**

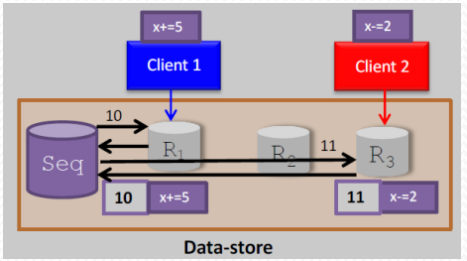
Event on Node 1	Event on Node 2
VC[1]= 5	VC[1]= 5
VC[2]= 2	VC[2]= 8

  - Causal
  - Because, each V[i] in node 2 is greater or equal than each V[i] in node 1

Consistency and Replication

- **Data-Centric vs. Client-Centric Consistency Models**
  - **Please explain the key difference between Data-Centric and Client-Centric Consistency Models.**
    - Data-Centric models define how the data updates are propagated across the replicas to keep them consistent.
    - Client-Centric models ensure that whenever a client connects to a replica, the replica is bought up to date with the replica that the client accessed previously.
  - **Please define Eventual Consistency (within one short sentence).**
    - A system is eventually consistent if all replicas will gradually become consistent in the absence of updates.
  - **Is Eventual Consistency a Data-Centric or Client-Centric Consistency Model?**
    - Client-Centric Consistency Model

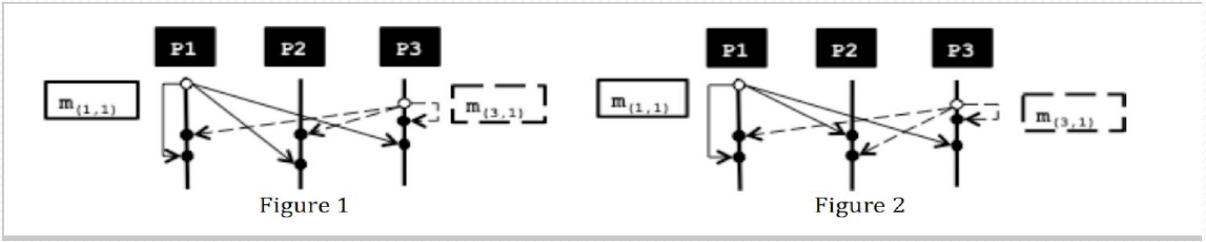
- **We discussed the concept of a Centralized Active Replication Protocol.**
  - **Briefly explain this concept. You can draw a figure to support your argumentation.**
    - There is a centralized coordinator sequencer.
    - When a client connects to a replica Rc and issues a write operation
      - Rc forwards the update to the Seq
      - Seq assigns a sequence number to the update operation.
      - Rc propagates the sequence number and the opration to other replicas
    - Operations are carried out at all the replicas in the order of the sequence number.



  - **Active replication uses a central entity for parts of its operations. Briefly explain why this is a reasonable design.**
    - To guarantee order of operations across the replicas.

- **Define the term replication. Be precise and brief.**
  - Replication is the process of maintaining the data at multiple computers
- **List at least three benefits that replication provides. Explain these benefits Briefly**.
  - **Improving performance**
    - A client can access the replicated copy of the data that is near to its location
  - **Increasing the availability of services**
    - Replication can mask failures such as server crashes and network disconnection
  - **Enhancing the scalability of the system**
    - Requests to the data can be distributed to many servers which contain replicated copies of the data
  - **Securing against malicious attacks**
    - Even if some replicas are malicious, secure data can be guaranteed to the client by relying on the replicated copies at the non-compromised servers
- **In the lectures, we discussed the concept of "Eventual Consistency". Explain it Briefly.**
  - A system is termed as eventual consistent when all replicas of that system will gradually become consistent in the absence of updates
  - Here write-write conflicts are rare
  - Read-write conflicts are more frequent
  - Can operate with limited connectivity
  - Handles network failure well
- **Please define the term "Consistency" and explain it briefly.**
  - In the scope of Distributed system consistency means for a system all of the replica for that system should be holding exact same information for certain object in other words every replica should be consistent about their information.
  - There are two kinds of consistency
    - Data centric consistency
    - Client centric consistency

- Below you see two figures showing an execution (sending and receiving of messages) on three nodes (P1, P2, P3). Two messages (m1,1, m3,1) are sent. Please note for each figure whether the depicted execution shows total ordering. Explain your decision; be precise and brief.
  - Figure 1
    - Total order maintained
  - Figure 2
    - Total order not maintained



- Bayou
  - **What is Bayou?**
    - A system for eventual consistency
  - **What consistency model does Bayou use? And Why? How are activities order in Bayou?**
    - Eventual consistency model is used
    - We want access to shared resource with limited connectivity
    - Vector clocks to order activities
  - **How are conflicts resolved in Bayou? Is this transparent to the user**
    - Application resolves the conflict
    - If same resource is accessed from multiple process at the same time conflict arises
    - Then we bayou updates the replica with the first request as higher priority and notify the other process about conflicts
    - for this vector clock is used
  - **When are conflicts resolved? Does Bayou give a guaranteed upper bound for This?**
    - Conflicts are eventually resolved
    - No guaranteed upper bound is given

Fault Tolerance

- **Orphans: A client might crash while the server is performing a corresponding computation requested by the client. Such an unwanted computation is called an orphan (as there is no parent waiting for it after done).**
  - **What problems do orphans cause?**
    - They waste CPU cycles
    - They might lock up files and tie up valuable resources.
    - If the client reboots, does the request again, and then an orphan reply comes back immediately afterwards, a confusion might occur.

  - **In the course, we discussed four strategies to deal with orphans. Please explain each of them.**
    - Extermination: Use logging to explicitly kill off an orphan after a client reboot.
    - Reincarnation: Use broadcasting to kill all remote computations on a client's behalf after rebooting and getting a new epoch number.
    - Gentle Reincarnation: After an epoch broadcast comes in, a machine kill a computation only if its owner cannot be located anywhere.
    - Expiration: each remote invocation is given a standard amount of time to fulfill the job.

- **We discussed the "Two Phase Commit" Protocol. As the name states, it consists of two phases.**
  - **Please name and describe phase 1 briefly.**
    - The coordinator sends a VOTE_REQUEST message to all participants.
    - When a participant receives a VOTE_REQUEST message,
      - It returns either a VOTE_COMMIT message to the coordinator indicating that it is prepared to locally commit its part of the transaction.
      - Or otherwise a VOTE_APORT message.
  - **Please name and describe phase 2 briefly.**
    - The coordinator collects all votes from the participants.
    - If all participants have voted to commit the transaction, then so will the coordinator. In that case, it sends a GLOBAL_COMMIT message to all participants
    - However , if one participant had voted to abort the transaction, the coordinator will also decide to abort the transaction and multicasts a GLOBAL_ABORT message.
    - Each participant that voted for a commit waits for the final reaction by the coordinator.
    - If a participant receives a GLOBAL_COMMIT message, it locally commits the transaction.
    - Otherwise, when receiving a GLOBAL_ABORT message, the transaction is locally aborted as well.
  - **Please discuss what happens in case of a failure during phase 1, i.e., a node not replying because it crashed.**
    - Participant crash in phase 1:
      - Before receiving VOTE_REQUEST: coordinator will not get that VOTE_REQIEST, system will halt.
      - After receiving VOTE_REQUEST , Before sending VOTE_COMMIT: other nodes will send VOTE_COMMIT, but not the crashed one.
      - After sending VOTE_COMMIT: Other nodes will continue to Phase 2, but not the crashed one.
    - Coordinator crash in phase 1:
      - Before VOTE_REQUEST: Protocol will not be initiated.
      - After VOTE_REQUEST: System halt, every other participants will wait for global commit.
      - After receiving VOTE_COMMIT: system halt. every other participants will wait for global commit.
  - **Please discuss what happens in case of a failure during phase 2, i.e., a node not replying because it crashed.**
    - Participant crash in phase 2:
      - Any participant crashes , crashed one will not commit anything, but other participants will commit.
    - Coordinator crash in phase 2:
      - Before sending GLOBAL_COMMIT/GLOBAL_ABORT: system will halt.
      - After sending GLOBAL_COMMIT/GLOBAL_ABORT: commit will be executed.

- **Define the term "Fault Tolerance". Be brief and precise.**
  - Fault-Tolerance: is the property that enables a system to continue operating properly in the event of failures.
  - For example, TCP is designed to allow reliable two way communication in a packet switched network, even in the presence of communication links which are imperfect or overloaded.

- **Failure Models: In the lecture we discussed different failure models. Please note four of them and describe each briefly.**
  - Crash Failure: A server halts, but was working correctly until it stopped.
  - Omission Failure: A server fails to respond to incoming requests.
    - Receive Omission: A server fails to receive incoming messages.
    - Send Omission: A server fails to send messages.
  - Timing Failure:
    - A server's response lies outside the specified time interval.
  - Response Failure: A server's response is incorrect.
    - Value Failure: The value of the response is wrong.
    - State Transition: The server deviates from the correct flow of control.
  - Byzantine Failure:
    - A server may produce arbitrary responses at arbitrary times.

Application

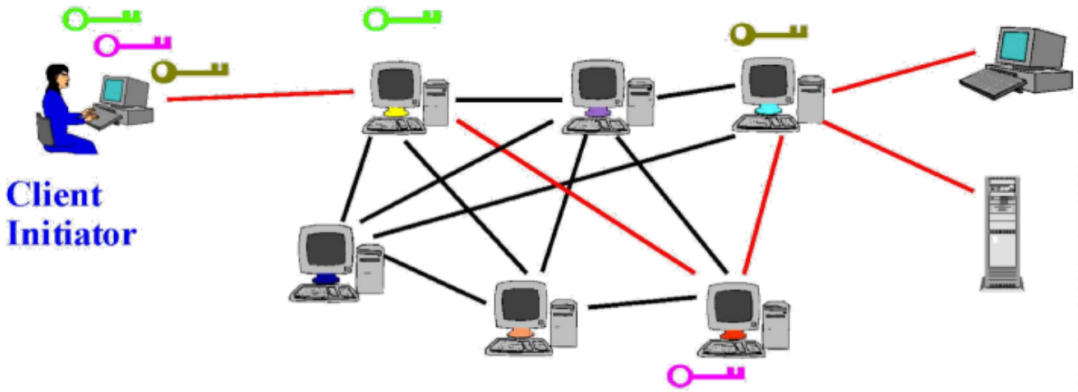
7. Applications.

1. We discussed TOR, which enables, for example, anonymous Internet browsing.

• Briefly explain how TOR provides anonymous Internet browsing. You can draw a figure to illustrate your argumentation.

• TOR also allows so called hidden services. Please briefly explain what a hidden service is and how TOR enables it. You can draw a figure to illustrate your argumentation.

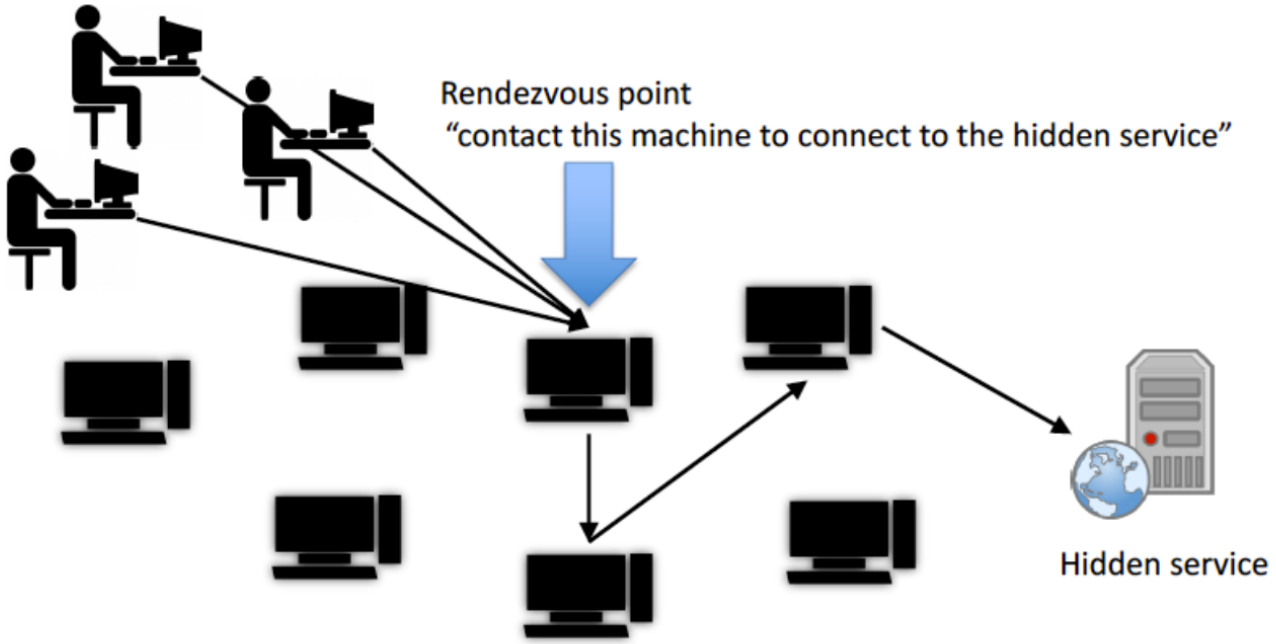
- TOR is based on onion routing. Client applications connect and communicate over the established TOR circuit.
    1. Circuit: anonymous tunnel between client and internet. Many applications can share one circuit.
    2. Directory servers: maintain list of active nodes, their locations, current public keys. Directory servers' key ship with TOR code.
    3. Layered encryption
    4. Each node only knows about each predecessor and successor



5.

- Hidden Service: We can deploy a hidden service on the internet that
    1. Anyone can connect to
    2. Without knowing where it is and who runs it
    3. Resistant to censorship, denial of service and physical attack

Sets up a tunnel, we are telling client to use this entry point, use TOR and use this specific public key to communicate this node to specific port. this entry point is called Rendezvous point. This will trigger the forwarding chain and access to the hidden service.



2. BitTorrent

Explain the basic concept of BitTorrent. Explain the terms “Seed(er)”, “Leech(er)”, “Tracker”, and “Swarm”,

Explain the concept of Tit-for-Tat that BitTorrent uses to avoid “free-riding”, i.e., ensuring that nodes that download also upload. What roles play “choking”, “unchoking” and “opportunistic un-choking” in this context?

- BitTorrent: Peer-to-Peer content distribution / File sharing
  1. Each file split into smaller pieces
  2. Nodes request desired pieces from neighbors
  3. Pieces not downloaded in sequential order
  4. Encourages contribution by all nodes
  5. Torrent file contents: the address of the tracker, Piece length (common 256 KB, configured by first uploader), SHA-1 hashes of each piece of file and ‘files’ allows download
  6. Tracker: runs on the web server (single point of failure). Keeps track of all peers downloading the same file
  7. Swarm: Set of all peers downloading the same
  8. Seed: Peer with entire file
  9. Leech: peer that’s downloading the file

- Avoid free riding: Restrict nodes only downloading
  1. Encourage all peers to contribute
  2. Choking: If peer A decides not to upload to B -> A said to choke peer B
  3. Tit-for-Tat: Each peer (say A) unchokes at most 4 interested peers. The 3 with largest upload rates to A
  4. Optimistic unchoke: Another is randomly chosen. To periodically look for better choices. If I unchoke a node, it might in turn unchoke me

3. Describe the map-reduce algorithm.Split it into two phases.For each phase include what it does and who is responsible. (map reduce framework or or the programmer).

-Map-reduce: A programming model for large scale computations. Process large amount of input, produce output.

You have huge input data. Data are processed in 2 steps.

1st phase: You feed into mappers. Once your mappers are done, you feed the result of each mapper to each reducer

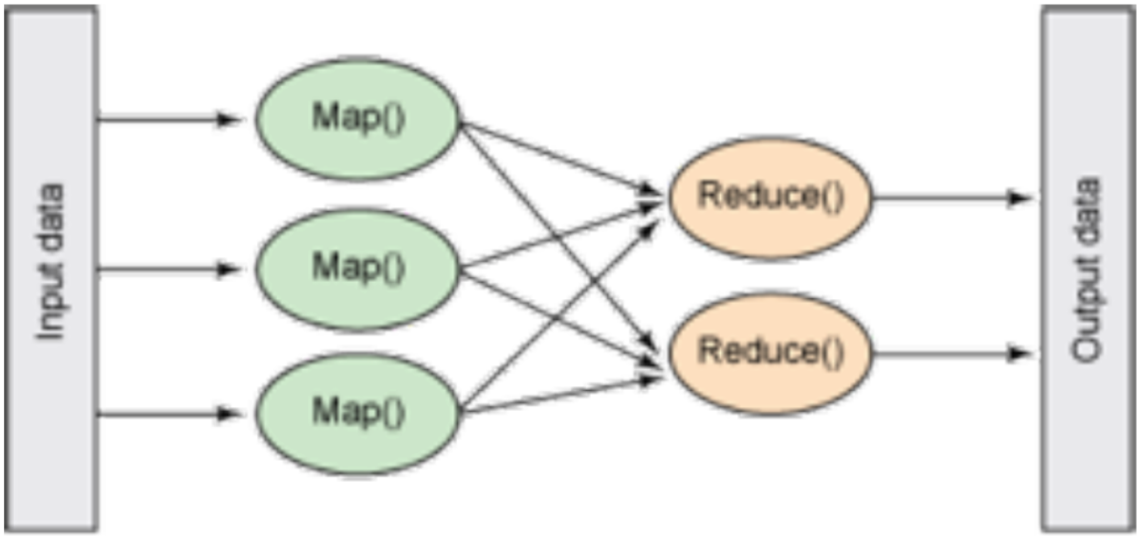
-Input data is partitioned into M splits (M map tasks)

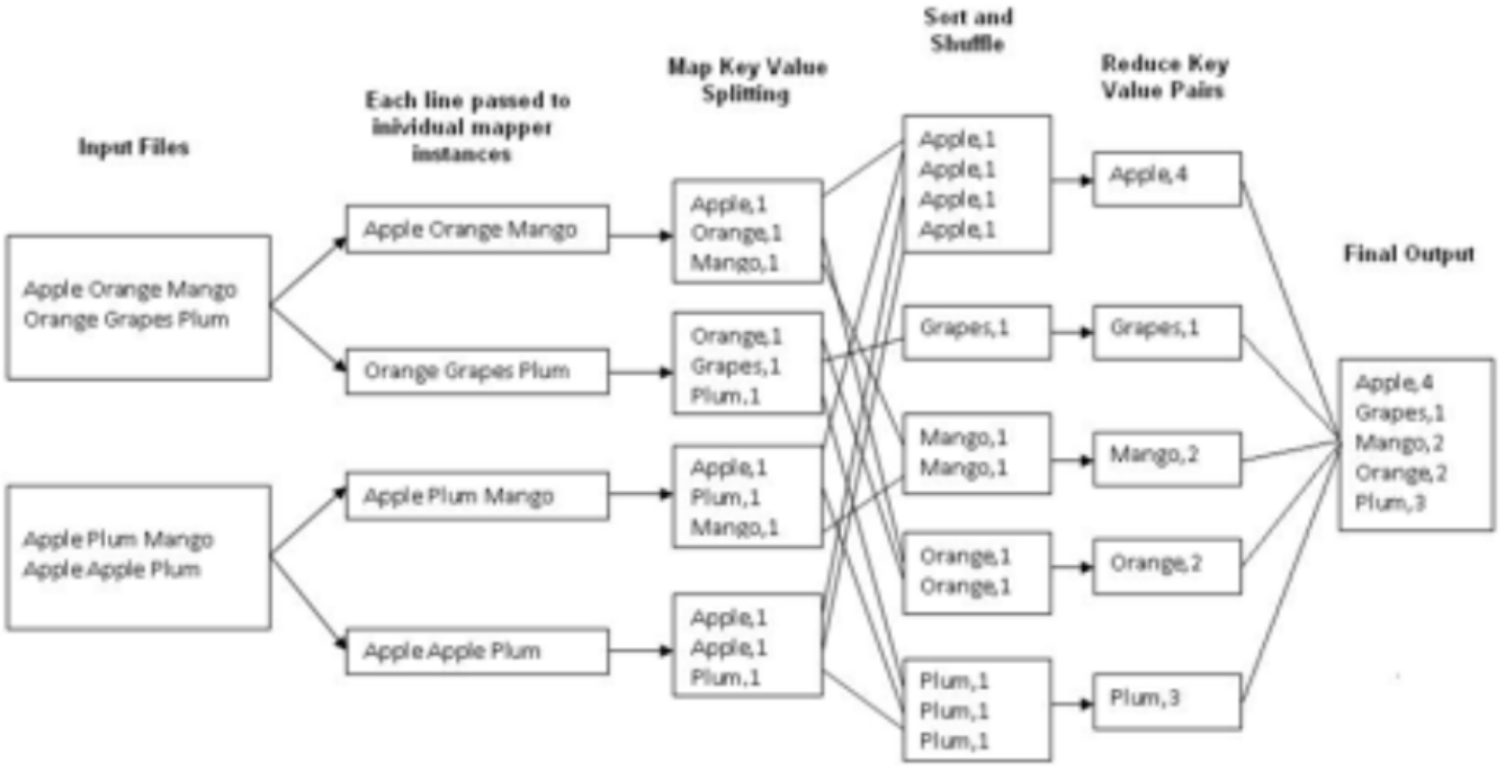
-Map: Extract information on each split. Each map produces R partitions (R reduce tasks)

-Shuffle and sort: Bring M partitons to the same reducer.

2nd phase: Reducer do the reduction task. Then you get the output.

-Reduce: aggregate, summarize, filter or transform




Input Files
Each line passed to individual mapper instances
Map Key Value Splitting
Sort and Shuffle
Reduce Key Value Pairs
Final Output
Apple Orange Mango
Orange Grapes Plum
Apple Plum Mango
Apple Apple Plum
Apple Orange Mango
Orange Grapes Plum
Apple Plum Mango
Apple Apple Plum
Apple,1
Orange,1
Mango,1
Orange,1
Grapes,1
Plum,1
Apple,1
Plum,1
Mango,1
Apple,1
Apple,1
Plum,1
Apple,1
Apple,1
Apple,1
Apple,1
Grapes,1
Mango,1
Mango,1
Orange,1
Orange,1
Plum,1
Plum,1
Plum,1
Apple,4
Grapes,1
Mango,2
Orange,2
Plum,3
Apple,4
Grapes,1
Mango,2
Orange,2
Plum,3

4. Amazon Dynamo, a (key, value) store for data centers.
• Dynamo is a modified version of the Chord protocol. What are the key differences? And Why?
• Dynamo introduces the concept of virtual nodes, what are they used for? (VVP)
• How are events in Dynamo timestamped?
• What consistency model does Dynamo use? And why?
• How are conflicts resolved in Dynamo? During which operation? And When?

- Dynamo is a modified version of Chord to adapt enterprise need.
  One problem with chord is nodes are randomly on ring that leads to uneven data and load distribution. Chord was designed to scale million of nodes but in dynamo we have small number of nodes like thousand. So chord is not a good fit for dynamo.

  Another problem of Chord is Finger table size: log(n), which scales million nodes. So for lookup: log(n) hops.
  For thousands nodes it isnt make sense. Dynamo stores every nodes stored in the ring.
  From log(n) routing performance to 1
  Routing table size: n
  Lookup: 1 hop

- Fix: Dynamo is a modified version of Chord DHT. Chord DHT is used to scale millions of node but in dynamo we have less nodes like thousands. So in dynamo they used introduced the concept of 'virtual nodes'. It means each physical node doesn't exist in ring only once but multiple of times. It is used to balance the ring. Each physical nodes have multiple virtual nodes. More powerful machines would have more virtual nodes.
-
- Events/ updates generate a new timestamp using vector clocks.
- Eventual consistency. Sacrifices consistency for availbility. Because the design choice is to show you old version of shopping cart over showing you no shopping cart. If there is a problem in system it may show you old version of shopping cart.
- Shopping cart is always writeable. So there might be conflicts. Conflict resolution is pushed to the reads. Its application-driven conflict resolution enforces last-writer-wins.

5.Ethical challenges: Certain distributed systems such as BitTorrent, Bitcoin and TOR trigger ethical challenges. Please select two of these systems and for each of them please discuss two ethical challenges.

-BitTorrent:

1. Use to distribute legal and illegal content
2. Hard to remove illegal contents
3. No central authority

Bitcoin:

1.can use it for drugs, illegal substances, guns and money laundering

TOR:

1. Illegal hidden web services
2. Can not trace, denial of physical attack

6. In the lecture, we discussed the concept of CAN.CAN is a distributed hash table. Answer the following questions.

> What topology does nodes form?
> What operations does a DHT eg provide?
> How is redundancy in CAN achieved.
> In CAN, how many hops does it make almost to lookup the data item? (Assume the number of nodes is “n”, and dimensions are “d”)
> How does a node join a CAN DHT?
> Neighbour table: Which nodes are stored in neighbour table of a node in CAN DHT?

-CAN: Content Addressable Network can maintain d-dimensional keyspace