OS for Database systems
File System and Networking

Seehwan Yoo

Dankook University

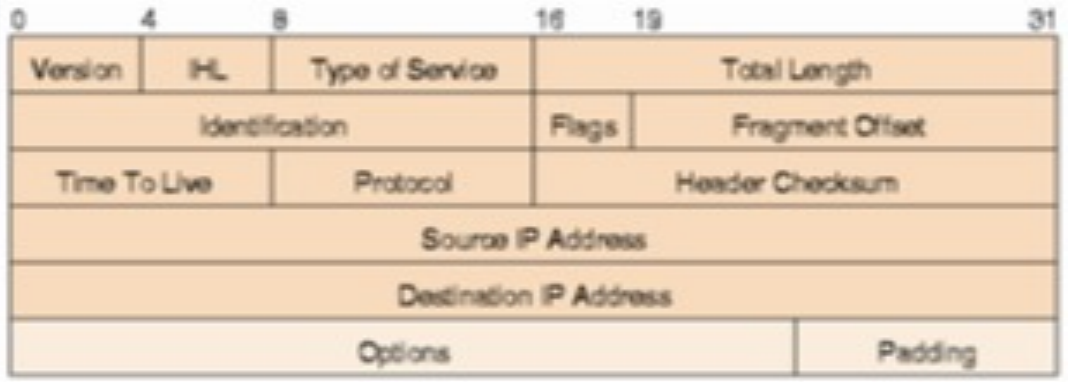
Disclaimer: Some slides are borrowed from UC. Berkeley's 2013 OS and system programming, and book slides from Computer Networks: A top-down approach

목차

- 강의 개요
- OS 이론 강의 (8/29~9/2)
  - OS 개론, 프로세스. + mini 실습
  - 병렬성. + mini 실습
  - 메모리, IPC. + mini 실습
  - 파일시스템, 네트워킹. + mini 실습
  - 클라우드 컴퓨팅 및 시스템 관리
- OS 실습 강의 (9/19~9/23)
  - 리눅스 개발 환경 구축 및 쉘 프로그래밍 (1d)
  - 다중쓰레드 자료구조 실습 프로젝트 (2d)
  - AWS 실습 및 WordPress 기반 DB 실습 프로젝트 (2d)

Internet & Distributed Systems

IP: Internet Protocol



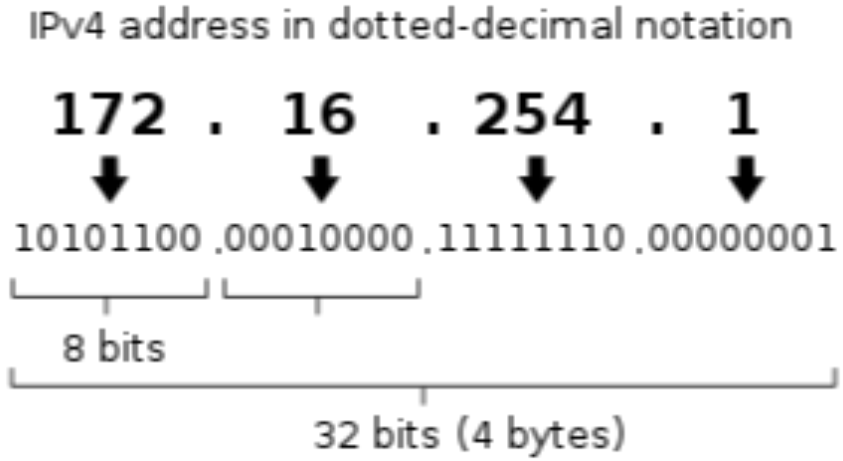
- Two important functions
  - Addressing and routing
- Address: location identifier, could be numeric value
  - Gyeonggi-do, Yongin-si, Suji-gu, Jukjeon-ro 152, Dankook University
  - 1, 2, 3, 4, etc.
  - IP address: pin-pointing a far-away computer, and computers
  - IPv4 (version 4) address size: 32 bits
    (Source IP Address, Destination IP Address in the packet header)
- Routing: finding route from any source network to destination network
  - forwarding data packets all over the world
  - How to know the best route (shortest path)?
    - Distributed graph algorithm (distance vector)
    - Optimal path finding through communication (Link state)

IP Address with classes

- IPv4: version4, 32 bits address, 4 bytes
  - Net ID + host ID = 32 bits
- A class citizens (large scale network)
  - Large network: host ID bits should be long
  - There are a few large networks, less bits for network ID
  - 7 and 24 bits for network ID and host ID, respectively
    - 1bit prefix (0) for A class network
- B class citizens (for medium size networks)
  - 2+14+16 bits (A network covers up to 65532 hosts)
    - 2bits prefix (10B) for distinguishing B class
- C class citizens (for small size networks)
  - Small network
  - 3+21+8 bits
    - 3 bits prefix (110B) for C class

Examples of IP address – from wiki

- 172.16.254.1
  - dot with every 8 bits
  - Prefix 10 → B class network
  - network ID: 172.16
  - host ID: 254.1
- Special address
  - Private address
    - Not used for public Internet ; instead, can be used for private network addr.
    - 10.0.0.0~10.255.255.255 for A class
    - 172.16.0.0~172.31.255.255 for B class
    - 192.168.0.0~192.268.255.255 for C class
  - Loopback or localhost : 127.0.0.1
  - 255.255.255.255: Limited broadcast



https://www.iana.org/assignments/iana-ipv4-special-registry/iana-ipv4-special-registry.xhtml

Subnet / CIDR

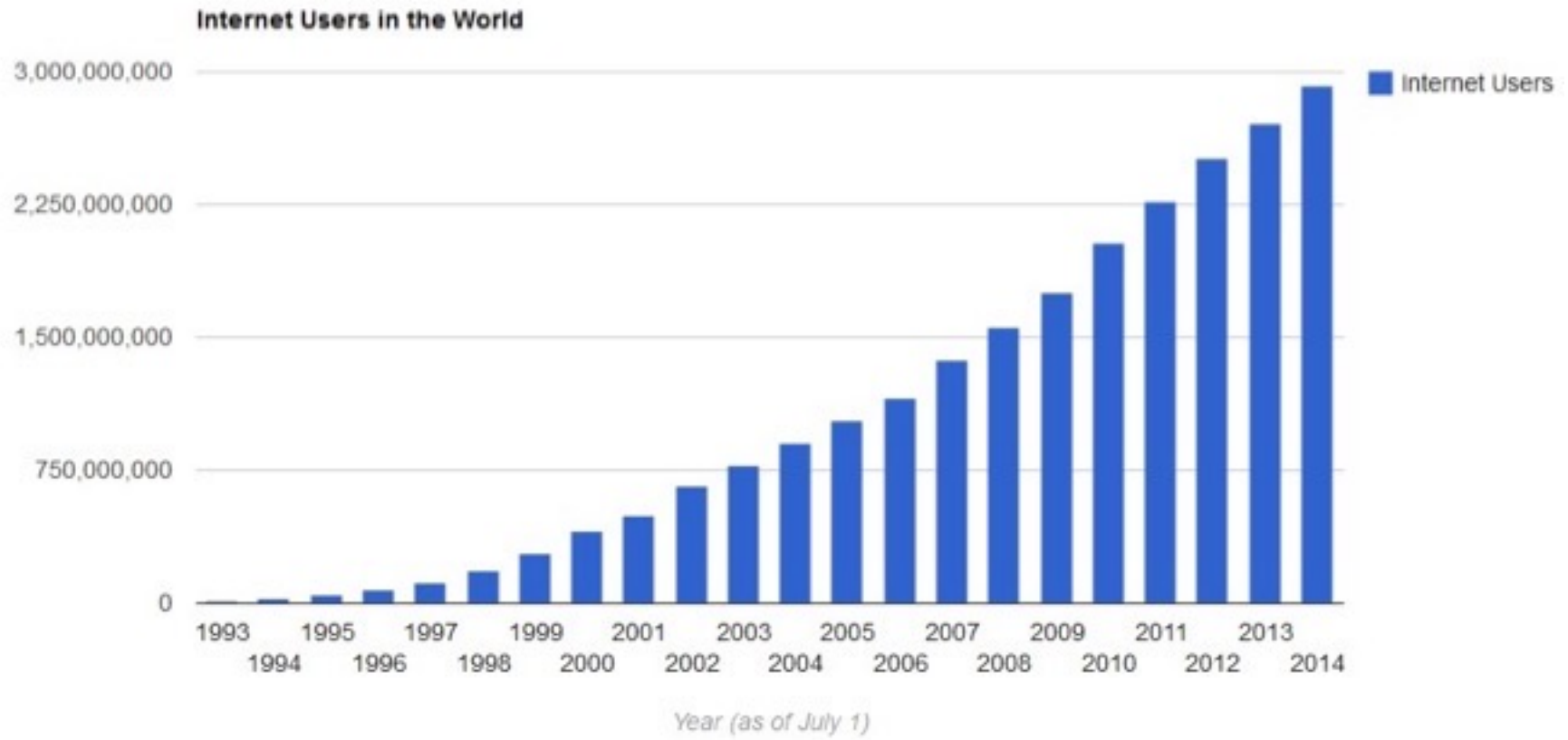
- Dividing a network into smaller ones
  - single B class network can hold up to 65532 hosts
    - Too large for single organization
  - C class network can hold up to 256 hosts
    - Too small for an organization
  - Dividing physical network into logical networks
    - Network addr → public network addr + subnet ID
    - B class network into smaller size multiple local networks
    - To distinguish public from local network, we use additional seperator '/XX'
- ex.) 172.25.145.21/20
  - 20 bits subnet mask: first 20 bits are used as subnet
  - B class private network, in the private network, it is identified with 172.25.90 (network ID) + 1.21 (host ID)
    - 145 = 0x91

CIDR: classless Inter-domain routing

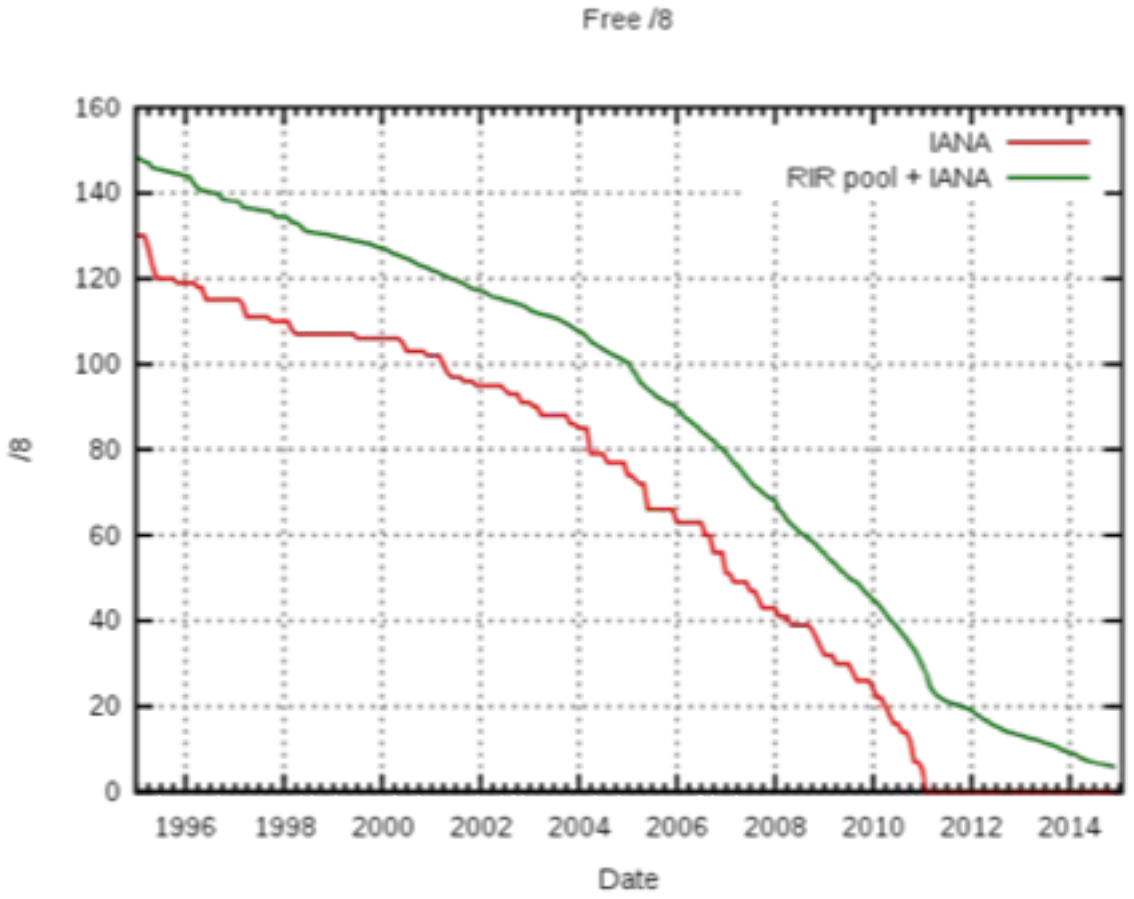
- Class-less Inter-domain routing
  - Instead of having static network ID bits, use more or less bits for network / host ID
- IP presentation
  - network ID host ID / subnet bits
  - Some bits in address are used as network ID
- My local IP address is?
  - ipconfig (windows), ifconfig (Linux)
  - Will show you some more info.

Unfortune! IPv4

- Depletion of IPv4 address was so quick
- How many computers can we have?



Depletion of IPv4 address was so quick



How large?

- How many computers can we have?
  - At most 2^{32} = 4G ~ 4,000,000,000
- World population?
  - Say, 7.6 Billion = 7,600,000,000
- Even with CIDR, IPv4 could not survive!

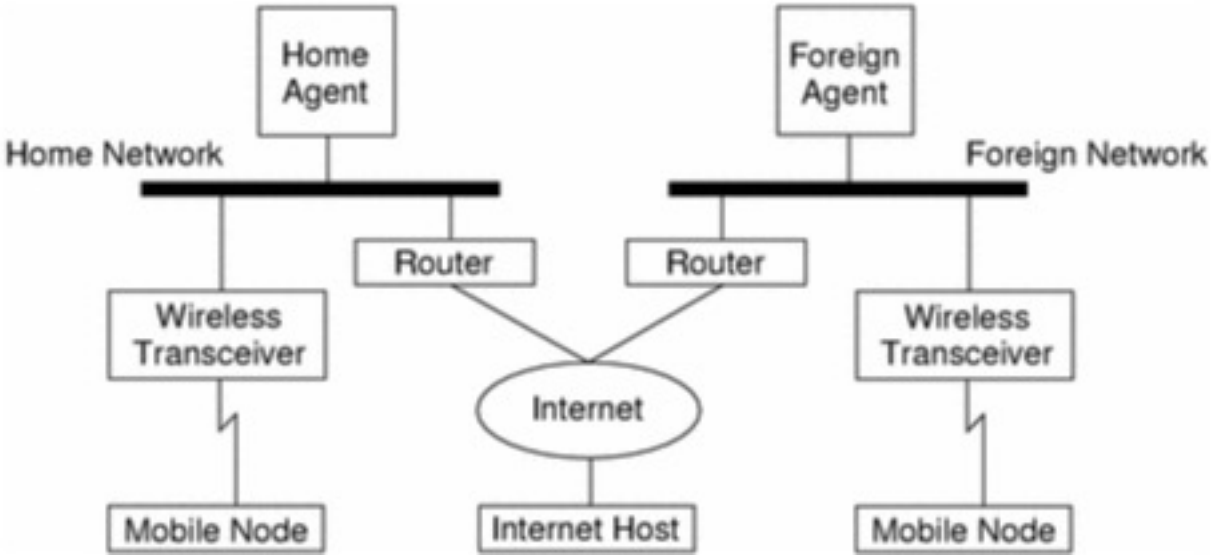
Welcome, IPv6 world

- Lets use more bits
  - How about 128 bits?
  - That's IPv6, XX bytes long
- Some more things of IPv6
  - IPv4 compatible addresses are statically allocated
  - Unicast, multicast link-local addresses are allocated
- It's already used in some applications

mobile IP

- IP address for mobile infrastructure

https://docs.oracle.com/cd/E19455-01/806-7600/6jgfbep0v/index.html



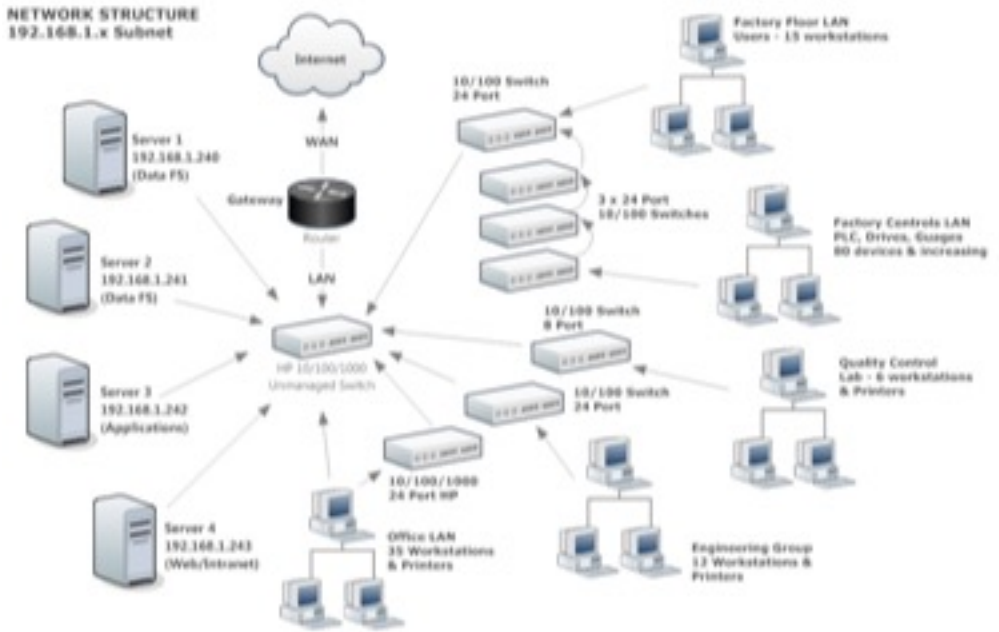
- As the user moves around the network access points, attached network (foreign network) is changing
  - your subscribed network is not changed (home network)
  - Home address, Care-of-Address are separated

Routing in the Internet

- Internet - named from Inter-networking
  - constructing a network consists of multiple networks
  - For example, Server - Ethernet - ATM - Optical Network - Ethernet - WLAN - Your mobile can possibly work
  - combining different kinds of networking / link technologies
- Two things to remember
  - Packet switching / forwarding
    - delivering data, packet by packet
    - IP packet: a unit of transferring data in the Internet
      - fragmentation: fragment user data into proper link frame size
  - Routing
    - algorithmic way to make a graph for the Internet

A basic workflow

- Terminal node (end host): where the user works on
  - sends packets to the default gateway
- Gateway (to the Internet): located in a network
  - forwards packets to the next router (one hop)
- Network: graph consisting of edges (links) and vertices (nodes)
  - forwards packets to the next router
  - repeat until the packet reaches the destination network
- Gateway receives the packet
  - sends the packet to local network



ARP: address resolution protocol

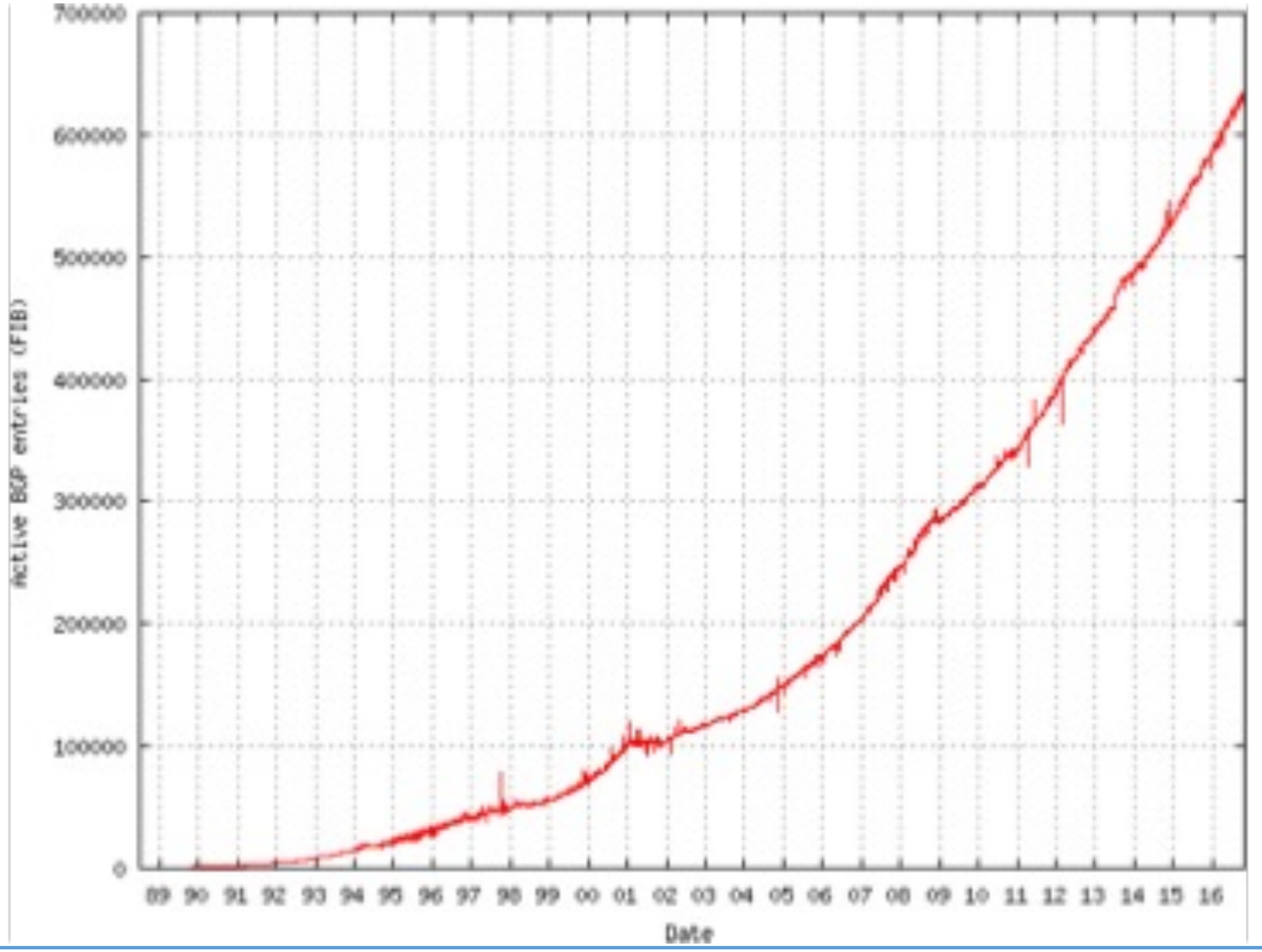
- Link protocol uses Link address, but IP addr is a global address
- Once a packet is received by the gateway,
  the gateway should deliver the packet to the destination host
  - using Link address
- How to translate IP addr to link addr?
  - broadcast ARP request, “who has this IP addr?”
  - the designated host replies with
    “I got the IP address, and my link address is this”

The gateway and the router

- gateway is a computer that distinguishes local / external networks
- The gateway has more than two network addresses
  - one for the local network
  - the other for the Internet
- To get the internet traffic, you need to register your router into the Internet, so that
  - the neighbor router will find you
- You can setup some security policies on the gateway
  - firewall, intrusion detection server, etc.
  - admission control to the local network (protection from DoS, port scan, block unused services, etc.)

The Internet, a pool of routers

- A router has a forwarding table
- When a packet is arrived,
  - the router looks up the destination, and
  - forward the packet to the next hop
- forwarding table has 600k~1M+ entries
  - long table implies long look up time
  - larger memory in the router
  - A modern router uses Content Addressable Memory (CAM) FIB / RIB structure



A private network

- cabling for yourself
- borrow / subscribe the dedicated cable (communication channel)
- Use VPN (virtual private network) service
  - ISP provides a logically separated network from the Internet
  - ISP makes tunnels among the VPN nodes and the VPN server (tunneling protocol)
    - uses encryption at the end points
  - Make sure that all your network connection goes through VPN tunnel

A private network in the cloud

- VPC - virtual private cloud
  - a private network in the cloud
  - You borrow EC2 instances, S3 services, RDS services
  - connect them with stronger security boundary, from another cloud users, Internet users
- compare with private network
  - you buy multiple computers, connected through VPN
  - VPN end-points use encryption protocols
  - All your traffic (incoming/outgoing) from/to the computers flows through VPN end-points, enhancing security
  - Your VPN client checks all your network port, blocks unintended network traffic

BTW. NAT and home network

- If you're using Wireless Access Point, your notebook has a private IP address (192.168.x.x)
  - Now, you will get to know this is not the public IP address
  - No, 127.0.0.1. That's the localhost, meaning yourself
- Then, how can I use private IP address?
  - Your Wireless AP provides NAT service
- Network Address Translation service
  - translate your private IP/port into public IP/port
    - subnetting your public local network
    - All your traffic flows into the default gateway, which is the Wireless AP
      - 192.168.0.1
    - Translate local IP/port to public IP/port, recalculate checksum

DHCP. dynamic host configuration protocol

- Numerous client devices incoming/outgoing
  - IP allocation/deallocation is very inconvenient, tedious
- A network server that
  - looks for a newcomer,
  - allocates a new IP address from a managed pool
  - assigns IP, the client can configure the IP addr, default gateway, DNS server
  - after some time, the borrowed address should be explicitly extended; otherwise reclaim the borrowed IP address
  - Yes, your Wireless AP runs DHCP server

TCP: the transport protocol

- Port, the communication end point
- Connection establishment
- Reliable communication
- Efficient communication
- Congestion control
- Flow control

The Internet and computer networks

- Connecting computers
  - Specifically, IP network
- Principles of computer networks
  - Not trustworthy, distributed
    - Interworking with another computer, which is not under our control
  - Protocol based
    - Peers should understand what you will do
  - Should be scalable
    - Connect computers in all over the world
  - End-to-end argument

Scaling with the Billions of nodes

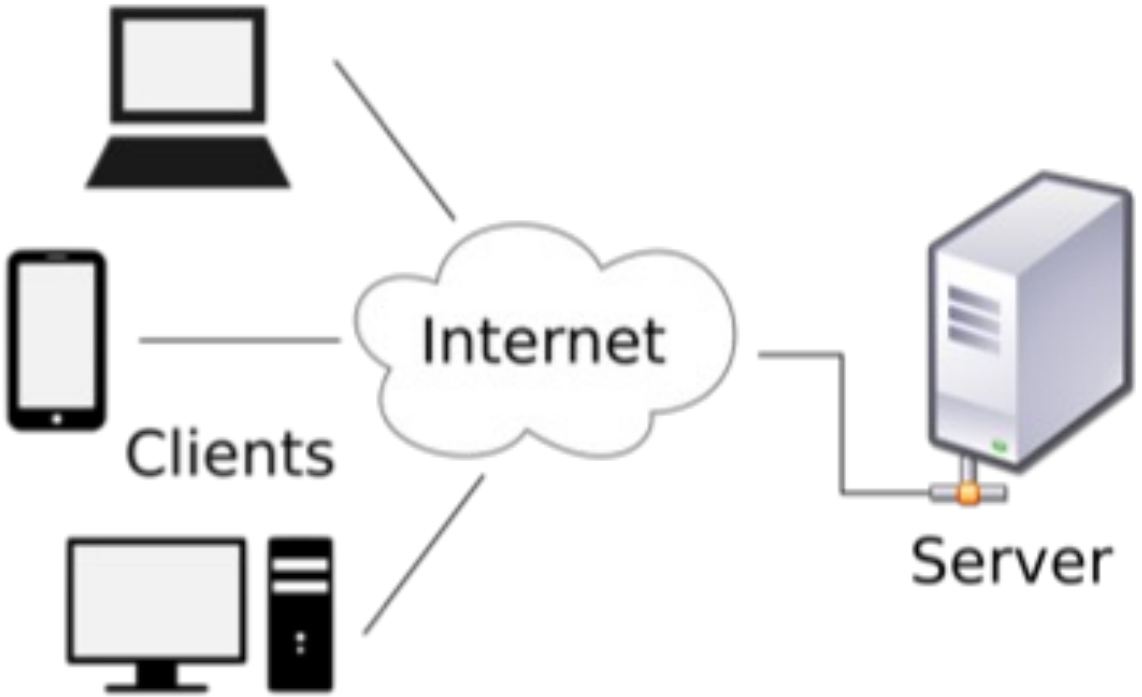
- Some considerations at global scale
  - Traffic control
  - Reliability
  - Fault-tolerance
- End-to-end argument
  - IP network: network of networks
    - Intermediary network can be down
    - Intermediary network would not be under our control
    - Intermediary network functions would not be available

End-to-end arguments

- Before the IP network, there were several networks
  - ATM, X.25
- Network layer did many things
  - Many functions such as congestion control, connection setup, error check, etc.
- IP focuses only two things (addressing, routing)
  - Because different network would not provide the same function (even fails)
  - Go back to simplest principle, data delivery
  - The rest of functions moved out to higher layer, end point of communication

Server-client model of the Internet

- Server: receive requests from many client computers
- Client: send requests to the server



If you google sth in the Web browser
The web browser is the client,
and some computer in the google
is the server.

The client sends query text to the google server.
The server finds the answer.
Then, the server responses with the result to the client.

OK, server-clients model needs

- Port: The communication end point
  - A server handles multiple requests from different clients
- In the old days, a link is dedicated to a connection
  - IP network?
    - No, simply forward packets to the next hop
    - No connection (session) at the network layer
  - Then, how can we maintain connection?
    - Multiple connections?

Port: communication end point

- A server handles multiple connections
  - A connection is maintained by port
- When a packet is delivered to a network card
  - IP has done all its duty!
  - But, which application (process) is responsible for this packet?
    - Look at the port of the packet
    - Send the packet to the process who opened the port

The Internet services and port numbers

- BTW, how can a client knows the port number of the server?
  - Well-known ports: 80, 25, 22, etc.
  - Your web browser sends google's web server the server port number is 80.
- What if…
  - All the clients in the world want to use 80 port?
  - After the connection establishment, the server uses different port for each connection

TCP, the transport protocol

- TCP connection setup
- Initiated from the client request
- A server opens well-known port
- Setup connection, and begins with private port
- Once the connection is established, server and client use per-connection ports pair
- Connection break down is necessary

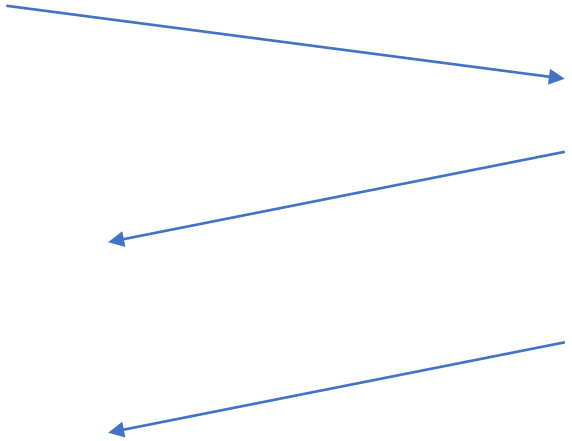
I. Establishing TCP session:
SYN-SYNACK exchange

- A client initiates connection with SYNc
  - The server replies with SYNACKc
- After receiving SYNACKc, the client is aware of the server is ready

- The server sends SYNs, to initiate backward session
  - The client replies with SYNACKs
- After receiving SYNACKs, the server is aware of the client is ready

Piggybacking SYNACKc on SYNs

- Client
- Connect
- Send SYNc



- Server
- Listen

- Recv SYNc
- Send SYNACKc

- Recv SYNACKc

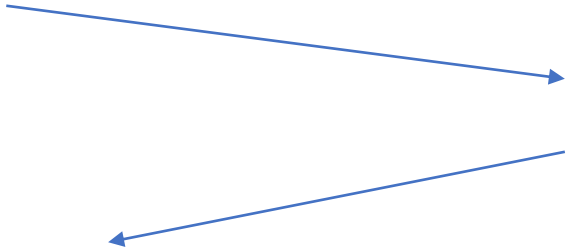
- Send SYNs

- Recv SYNs
- Send SYNACKs
- Established

- Recv SYNACKs
- Eatablished

Piggybacking SYNACKc on SYNs

- Client
- Connect
- Send SYNc



- Server
- Listen

- Recv SYNc
- Send SYNACKc + SYNs

- Recv SYNACKc + SYNs
- Send SYNACKs
- Established

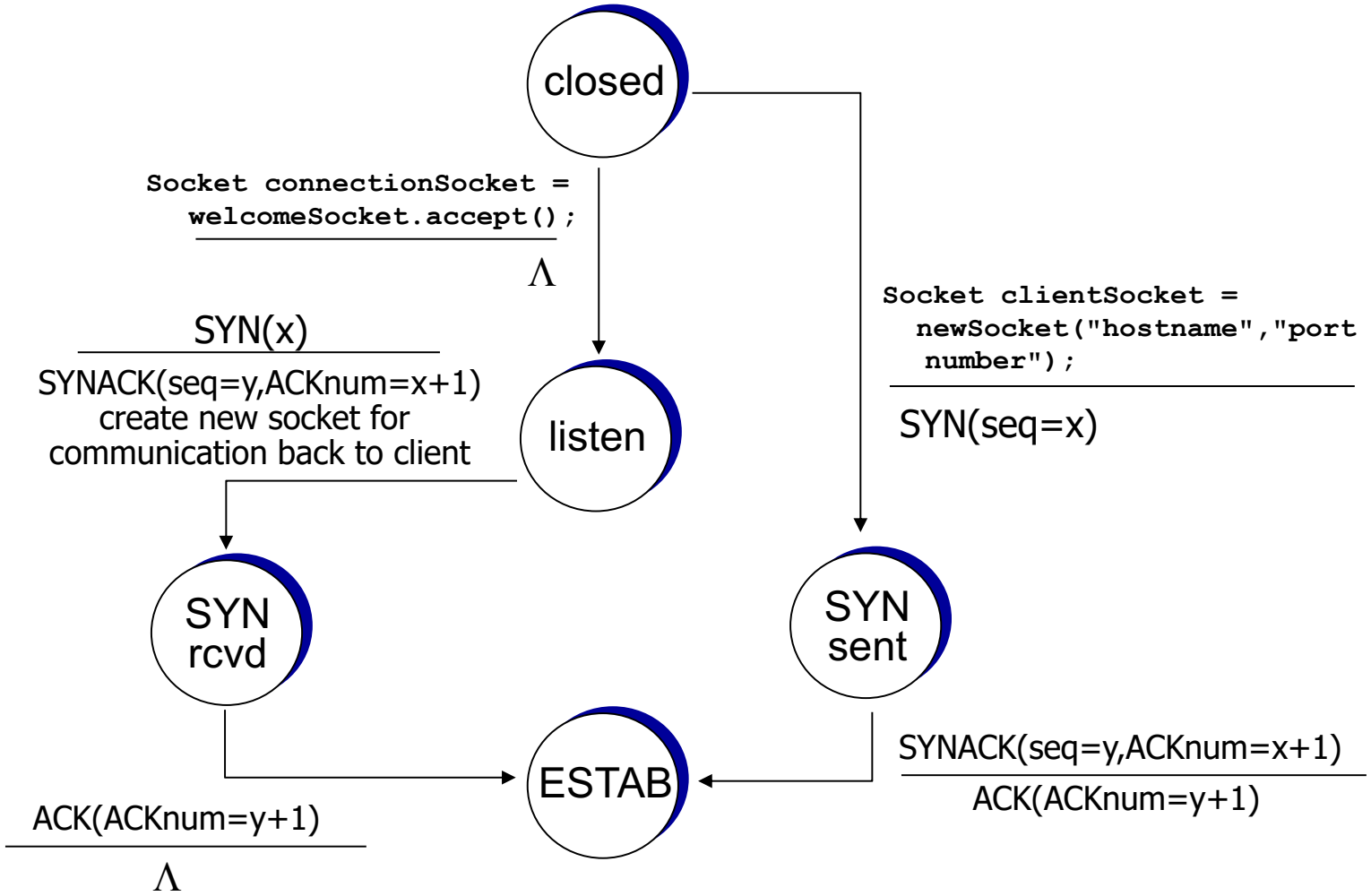
- Recv SYNACKs
- Eatablished

Meet the three-way handshaking!

- A client initiates connection with SYNc
  - The server replies with SYNACKc
- After receiving SYNACKc, the client is aware of the server is ready

- The server sends SYNs, to initiate backward session
  - The client replies with SYNACKs
- After receiving SYNACKs, the server is aware of the client is ready

TCP 3-way handshake: FSM (finite state machine)



Side note: SYN flooding

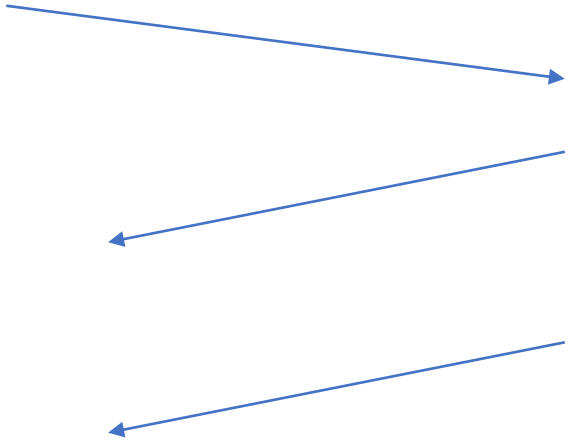
- When a server receives SYN, it prepares a new port (and new connection)
  - Reserve some memory, threads, etc.
  - i.e. there are some overhead
- What if…
  - A bad guy sends many SYN packets (SYN packets only)?
  - The server continuously allocates additional resources
  - What if the server consumes all the resources in handling SYN packets?
    - No effective connections available
    - That's what we call DoS (Denial of Service) attack

Connection closing

- FIN-FINACK exchange
- Initiate close req. from either the server or the client
  - Usually, the client begins; but the server can begin

FIN/FINACK exchange

- Client
- Connect
- Send FINc



- Server
- Listen

- Recv FINc
- Send FINACKc

- Recv FINACKc

- Send FINs

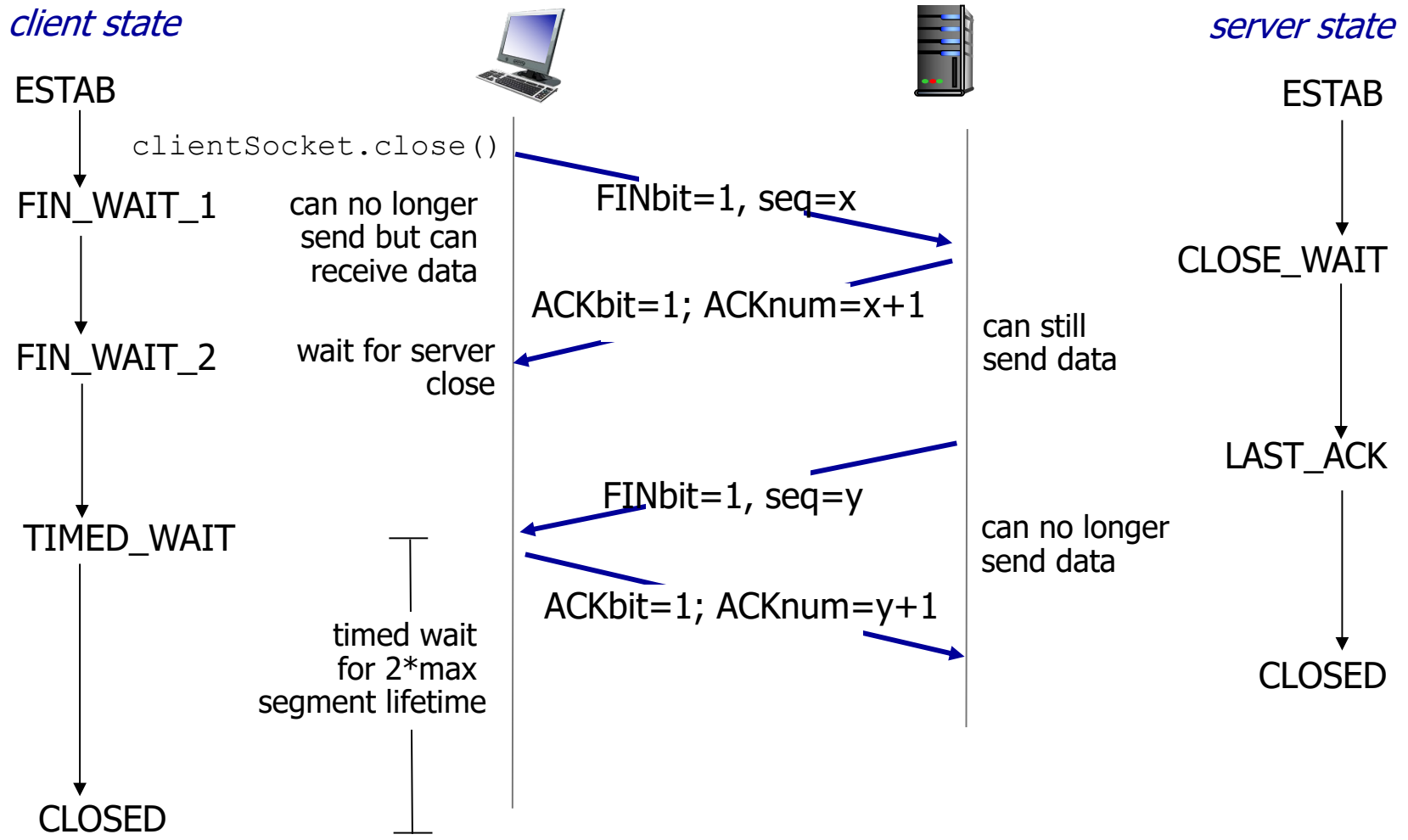
- Recv FINs
- Send FINACKs
- Established

- Recv FINACKs
- Eatablished

Why is that so complex?

- After sending FIN, one cannot send data anymore
  - Either the server or the client
- Why not piggyback? FINACKc-FINs
  - You can
  - The server can send FINs before the client sends FINc

TCP: closing a connection



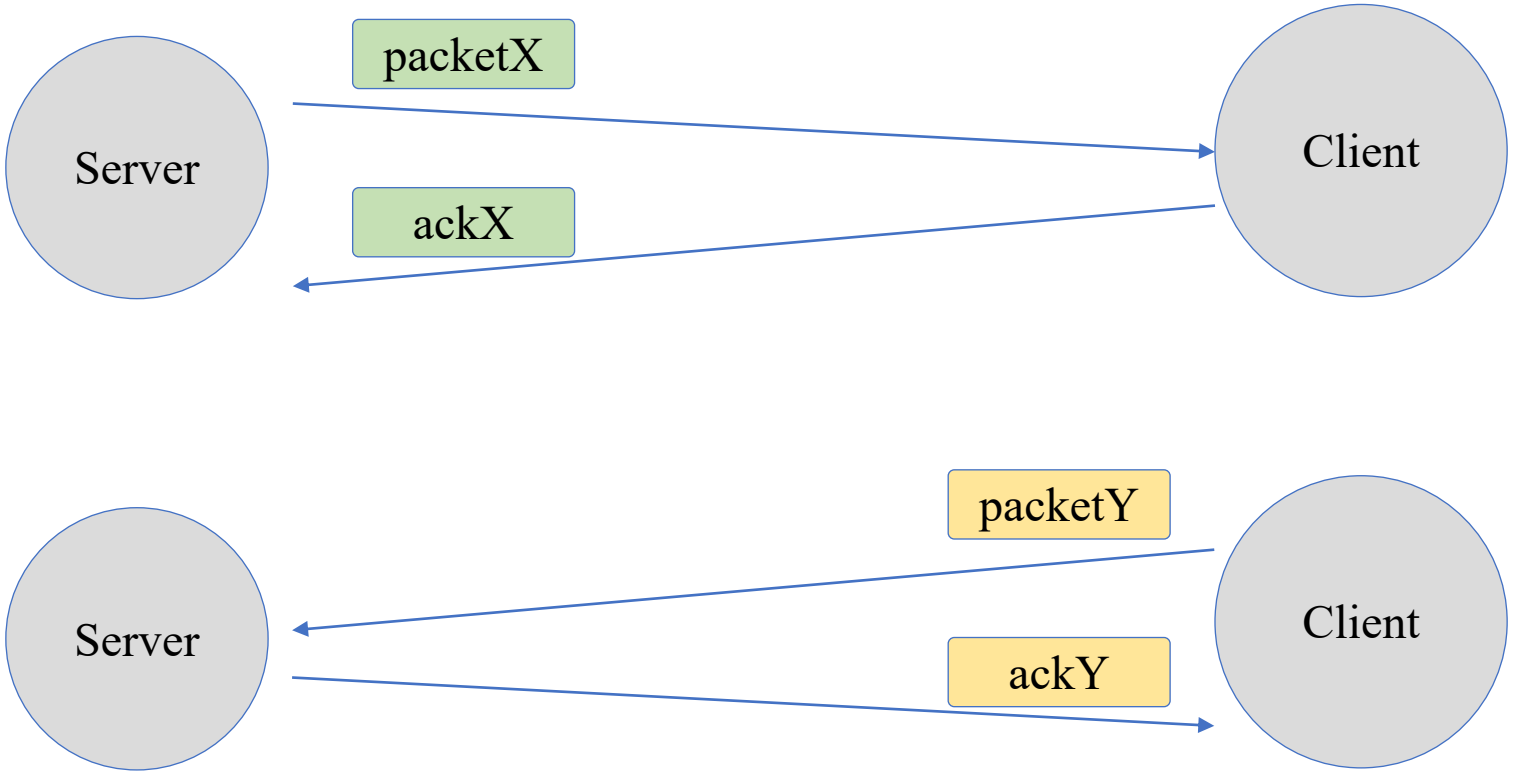
II. TCP session: reliable ordered byte stream

- What's RELIABLE byte stream?
  - If you send data, you can be sure that the data is delivered to the destination
- How?

TCP session: reliable byte stream

- What's RELIABLE byte stream?
  - If you send data, you can be sure that the data is delivered to the destination
- How?
  - Receive ACK message
- Send data_x → receive ack_x
  - Either the server or the client

Acknowledgement



What if a packet is lost?

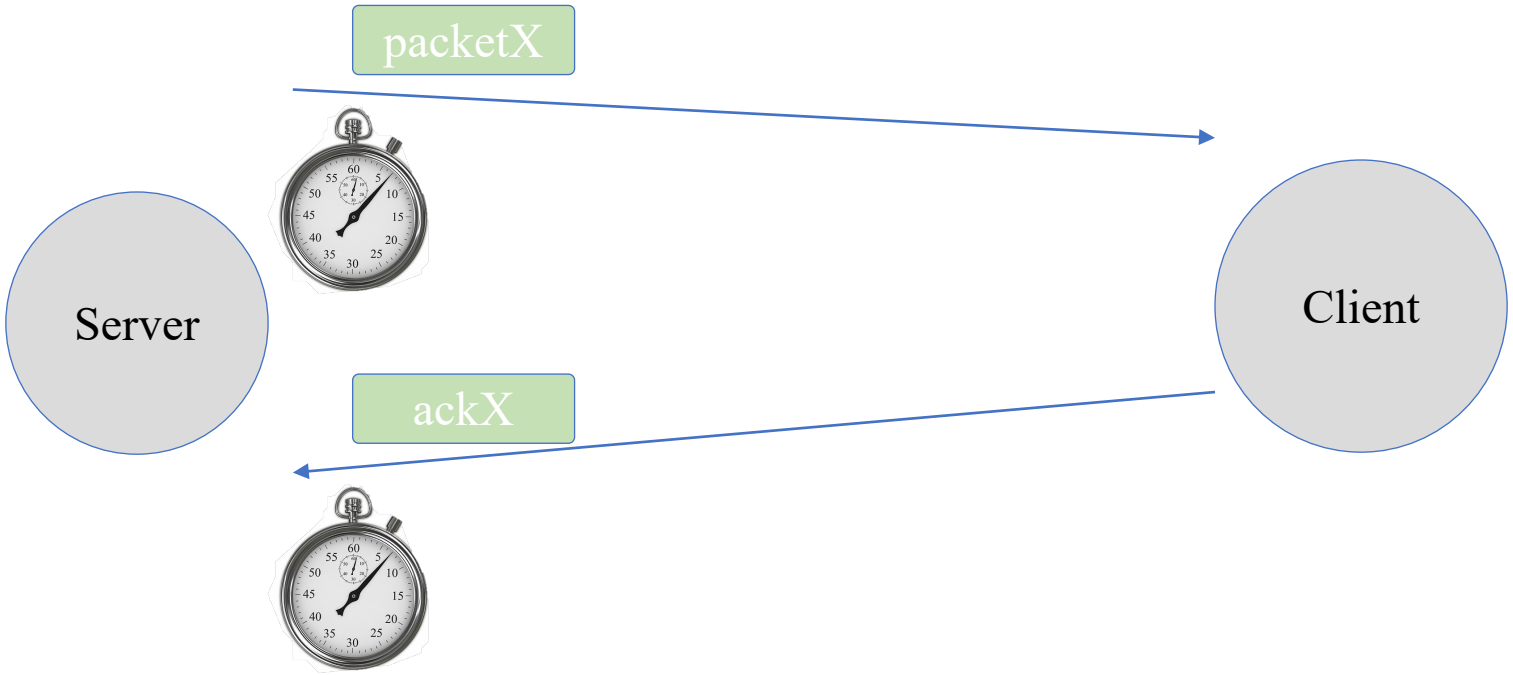
- If data packet is lost
  - The receiver cannot get data packet, so that the receiver cannot send ACK
  - Thus, the sender cannot get ACK
- If ACK is lost
  - Although the receiver sent ACK, but ACK is lost in the middle
  - Thus, the sender cannot get ACK

➔ The sender cannot get ACK

- If the sender does not get ACK, the sender cannot make sure the packet is arrived or not

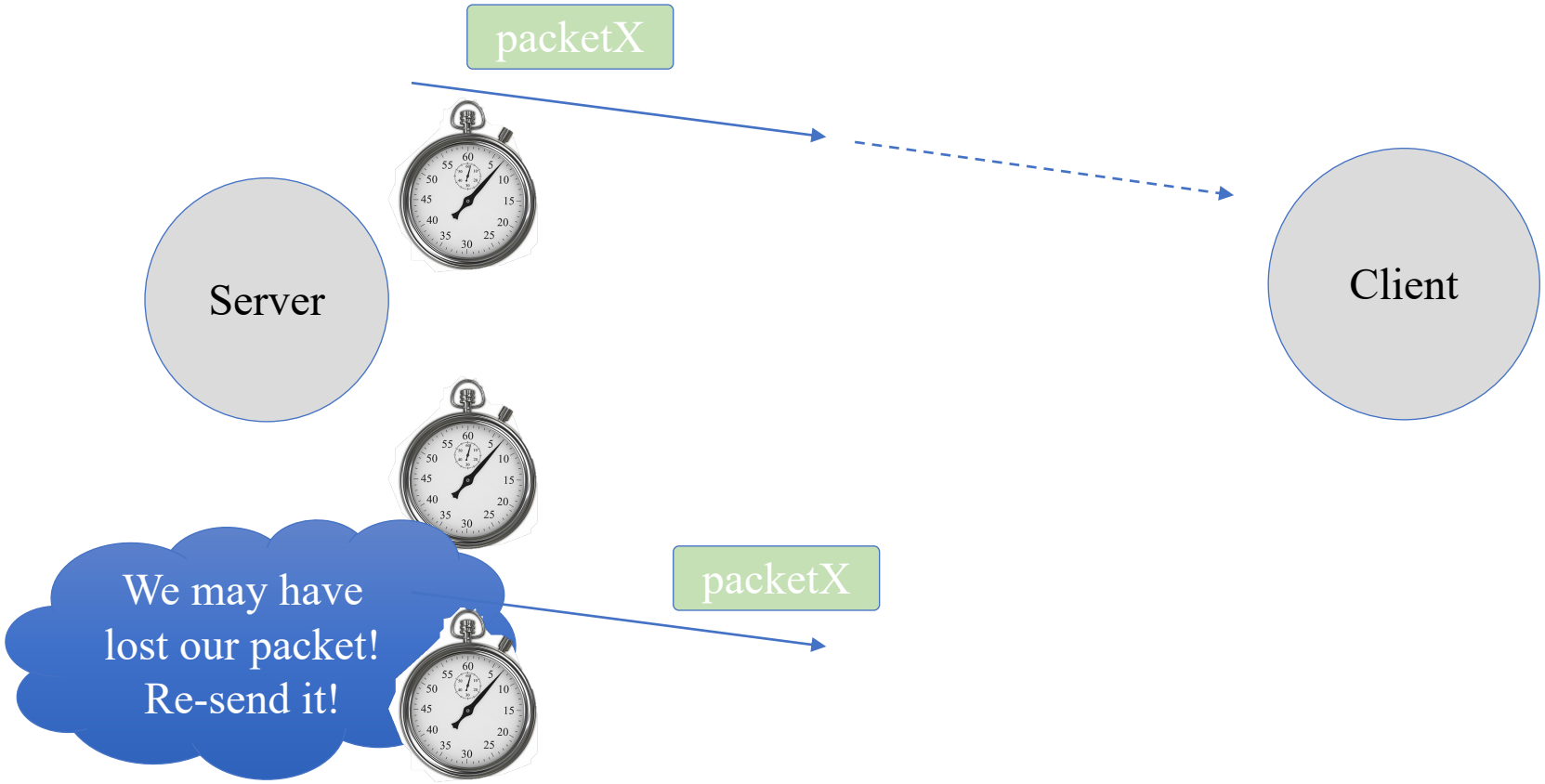
TCP made it reliable

- Re-transmit



TCP made it reliable

- Re-transmit



Timer length?

- How long do we need to wait?
  - Too small: packet would arrive in near future
  - Too large: lost detection is late, slows down further progress, retransmission
- Use heuristic
  - RTT (Round-trip time)
  - Min (2*RTT, 1ms)

III. Sending rate of TCP

- One Ethernet frame is 1500 bytes long
- Let's say RTT is 10ms (from DKU to google server in US West)
- Performance?
  - How long do we need to wait for sending 150KB jpg file?
  - Q) calculate the throughput (Kbps)

Sending multiple packets at a time!

- Assume you have 10Gbps Ethernet network card, that can handle every 1500 bytes-long packets in 1.2 microseconds
  - 1 Microsecond = 10^{-6} second
- How quickly can you send 150K jpg file in theory?

Sending multiple packets at a time!

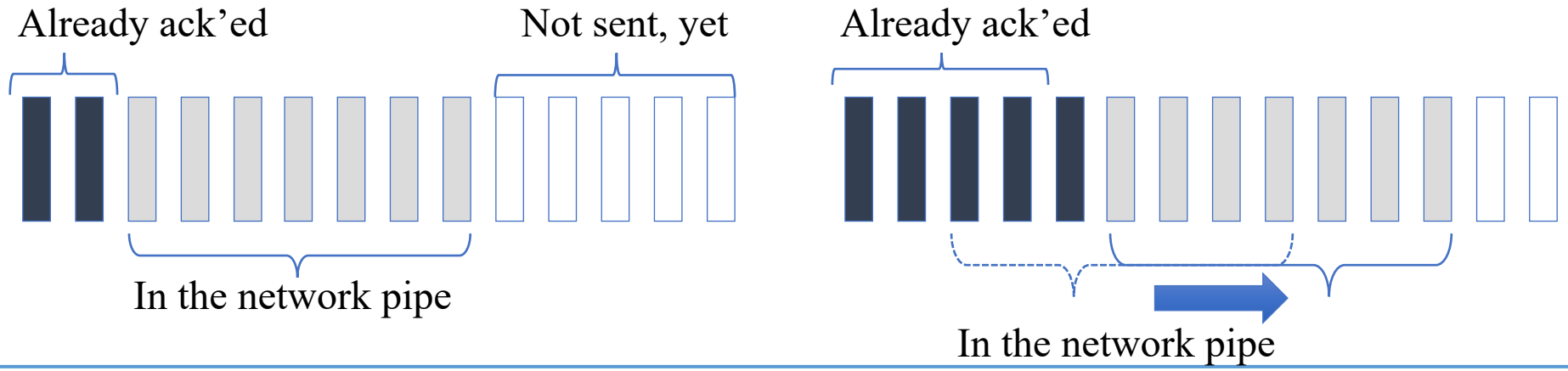
- Assume you have 10Gbps Ethernet network card, that can handle every 1500 bytes-long packets in 1.2 microseconds
  - 1 Microsecond = 10^{-6} second
- How quickly can you send 150K jpg file in theory?
  - 10ms + 120us
  - Why?

Yes, TCP does this!

- WARNing, it is going to be dirty!
- Pipeline!
  - While the first packet is on the way, send the second, and third packets

Sliding Window

- Sliding pipeline
  - We want to fully utilize network pipe
  - Once a packet gets out of the pipe, push another packet
  - Keep the pipeline full
- Like this,



Analogy: pipeline in computer arch.

- What was the difficulty?
  - Dependency! Pipeline hazard
    - What if…
      Some instruction in the middle is dependent upon previous ones?
- What if…
  - A previous packet is lost?
  - Multiple previous packets are lost?

TCP-lost packets handling

- Lost packets detection
- Retransmission
- Adjust sending rate

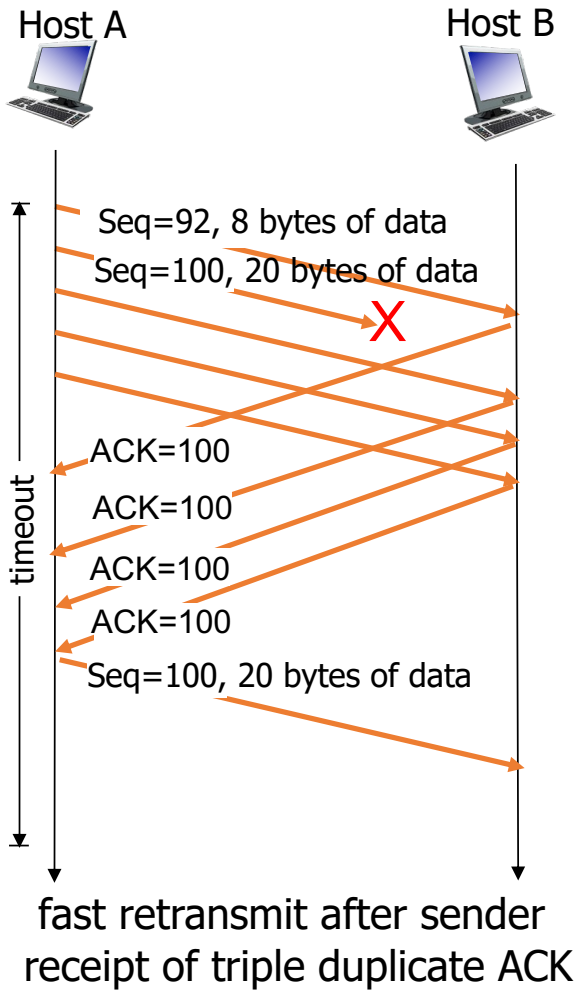
Lost packets detection...

- How to detect packet loss?
  - Receiving per-packet ACK
  - Better way?
  - Clue… Multiple packets in the pipeline

Detecting packet loss

- The receiver remembers the last packet in sequence
  - Send ACK for the last-seen packet
  - The sender will see the same ACKs
    - ‘triple dup ACKs’ → diagnose as the subsequent packet loss
    - Trigger (fast) retransmit before timeout!
    - Send from there, immediately!

TCP fast retransmit



fast retransmit after sender receipt of triple duplicate ACK

Addendum: What if multiple losses?

- Think about the cases when the windows size is 4, and two packets are lost
  - Fixed sliding window size
  - How can we open the sliding window further before timeout?
    - TCP-newReno
    - TCP-Sack

IV. Congestion control of TCP

- What's the optimal sliding window size?
- What if…
  - All the computers send data at maximum rate?
  - The internet traffic would be congested
  - The internet collapse
- Router will drop packets if
  - The queue is full, and additional packet incomes
- How to avoid congestion?

End-to-end control of the Internet traffic

- Congestion avoidance
  - If packet is dropped, it means router queue is full
    - We don't know which router, but something on the path (should be the bottleneck router) is congested
  - Decrease the sending rate by half
    - Relax the congestion on the path
- If not congested, opportunistically increase sending rate
  - Until the network is congested
- Each node
  Additively increase, multiplicatively decrease
  the sending data rate
  - To reach the global fairness [there is a proof for this!]

Starting window size

- One packet
  - Linearly increase → too slow to find congestion point
  - Multiplicatively increase until the first packet drop occurs
- Fast increase of initial sending rate
  - Due to slow-start
  - So it is called as ‘slow-start’

Putting them all together

- Adapting the sending rate
  - To fully utilize network bandwidth on the path
  - While achieving global fairness

Flow control

- Limiting the sending rate according to the recipient's buffer size
- Let the sender know the available buffer size (max window size)

They're all implemented in OS kernel

- That's what we call TCP/IP stack
  - With all the exception handling
  - All the timer handling
  - All the sending rate control
  - All the ack/selective retransmission
  - All the buffer management for user data into packets

Academic research article that brings the Internet world
V. Jacobson. 1988. Congestion avoidance and control.
In Symposium proceedings on Communications architectures and protocols (SIGCOMM '88),
Vinton Cerf (Ed.). ACM, New York, NY, USA, 314-329. DOI=http://dx.doi.org/10.1145/52324.52356
[paper]

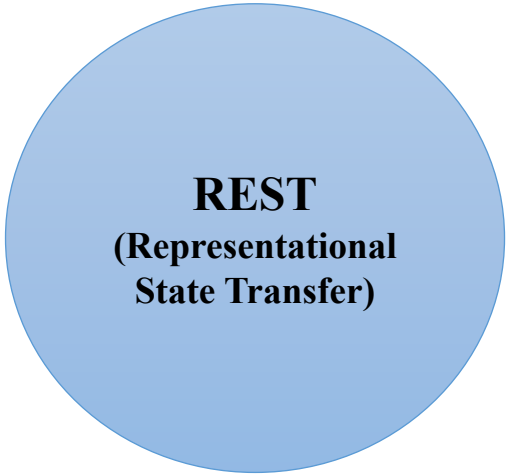
Some web technologies

Web tech.

- GUI: browser renders the page on a screen
- Web server: business logic, data presentation
- DBMS / DB server: data storage

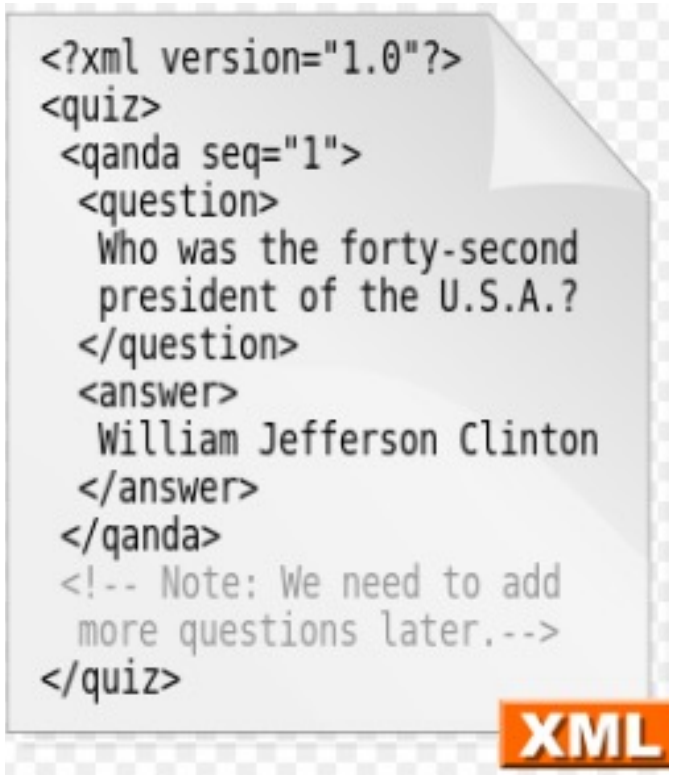
Services Tech.

- SOAP and REST
  - SOAP: XML based network data communication protocol
    - a standard
  - WSDL
  - XML schema
  - SOAP
  - UDDI



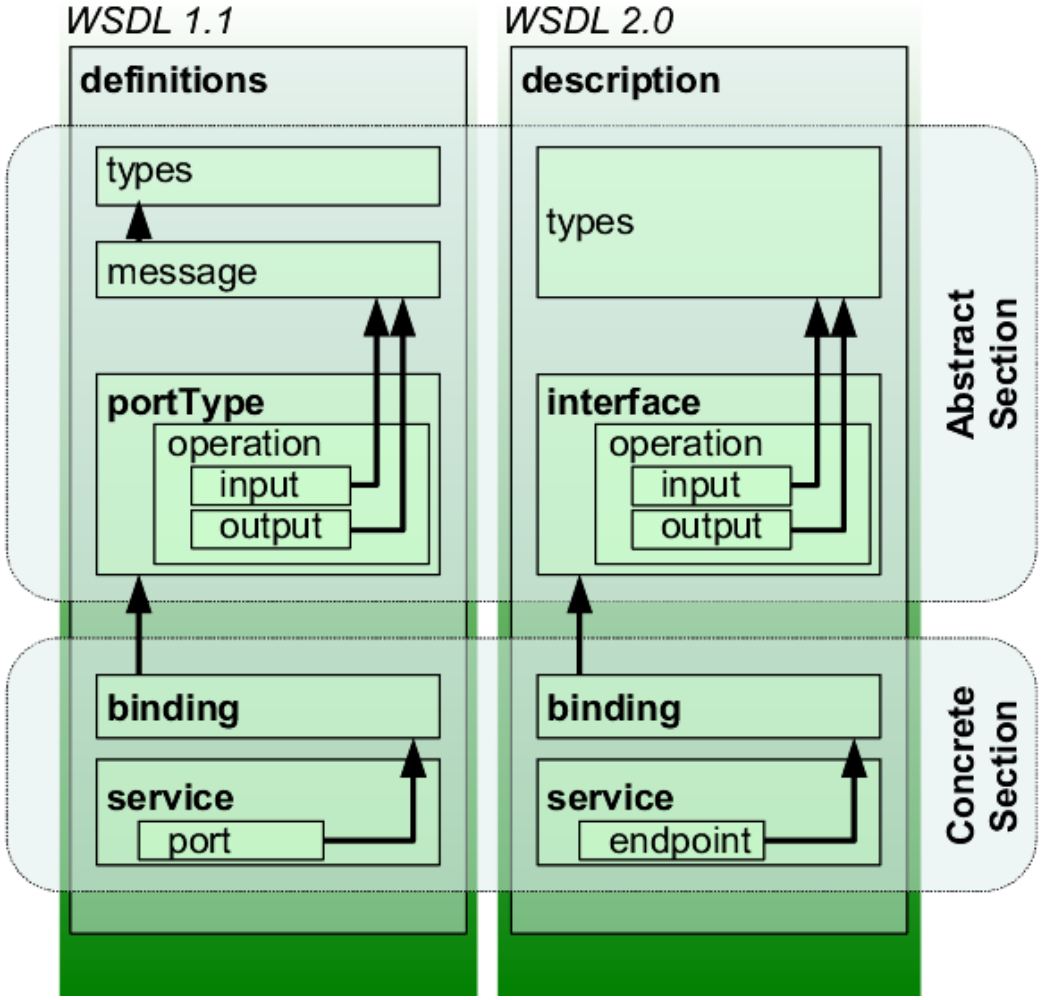
WSDL **web service description language**	define interface, service logic, etc.
XML Schema definition language	define message types of web services
SOAP	message exchange, remote procedure calls
UDDI	Universal Description. Discovery, and Integration of services

XML: representation of data / services



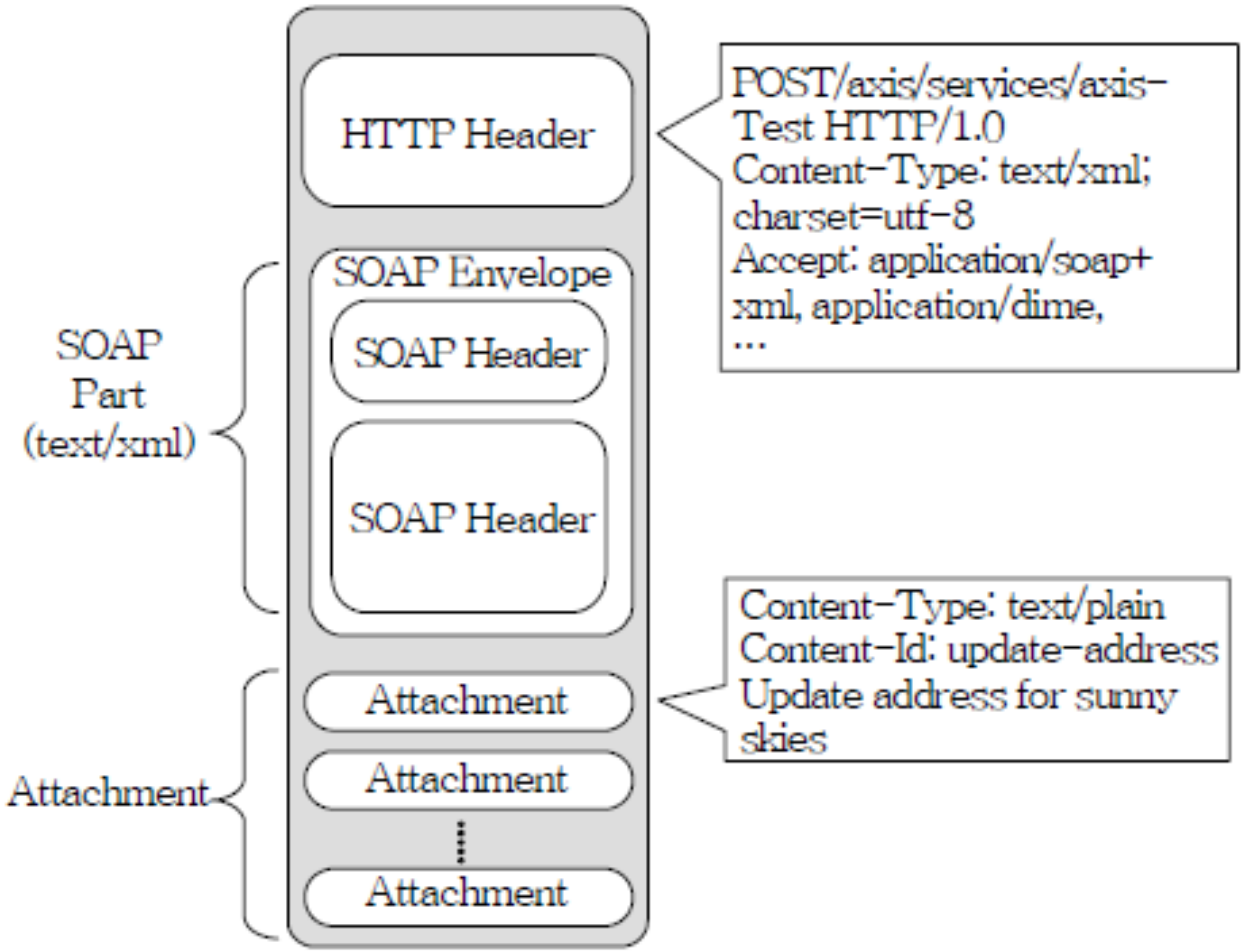
1. XML(Extensible Markup Language) is a standard markup language for defining another markup languages
2. adv. over HTML: intercommunications over systems on the Internet
3. not only for humans, but also for communication among the computers (computer-understandable)

WSDL(Web Services Description Language)



- Language that defines a web service (in XML)
- including
  - resource location, service provider, message format, protocols, etc.

SOAP: simple object access protocol



- runs over HTTP
  - proxy-free, firewall-free (in many cases)
- support multi-protocols (HTTP, SMTP, etc.)
- platform, programming language-independent
- simple and extensible (Multipart MIME), and supports attachments

UDDI: universal description, discovery and Integration

- A standard for
  - to search for a service aka. google for searching web
  - to register and integration of services

  aka. webinfo/portal for DKU online service
- example) web service: data store
- UDDI for service registration
  - to be searched by some other user
- UDDI has WDSL
  - service access format (protocol description) in XML
  - how to access the data objects in the data store
- SOAP to effectively access to the files

SOAP: standard way to make a web service

- standardized by W3C
- defines a general way of making a new web service
  - SOAP
  - WSDL
  - UDDI
- but, complicated

REST: an alternative

- Representational State Transfer
  - Non-complicated web service architecture
  - client-server architecture
  - state-less, thus light-weight
- Uses HTTP
  - GET/PUT to simplify SOAP

Distributed Systems

Motivations for distributed systems

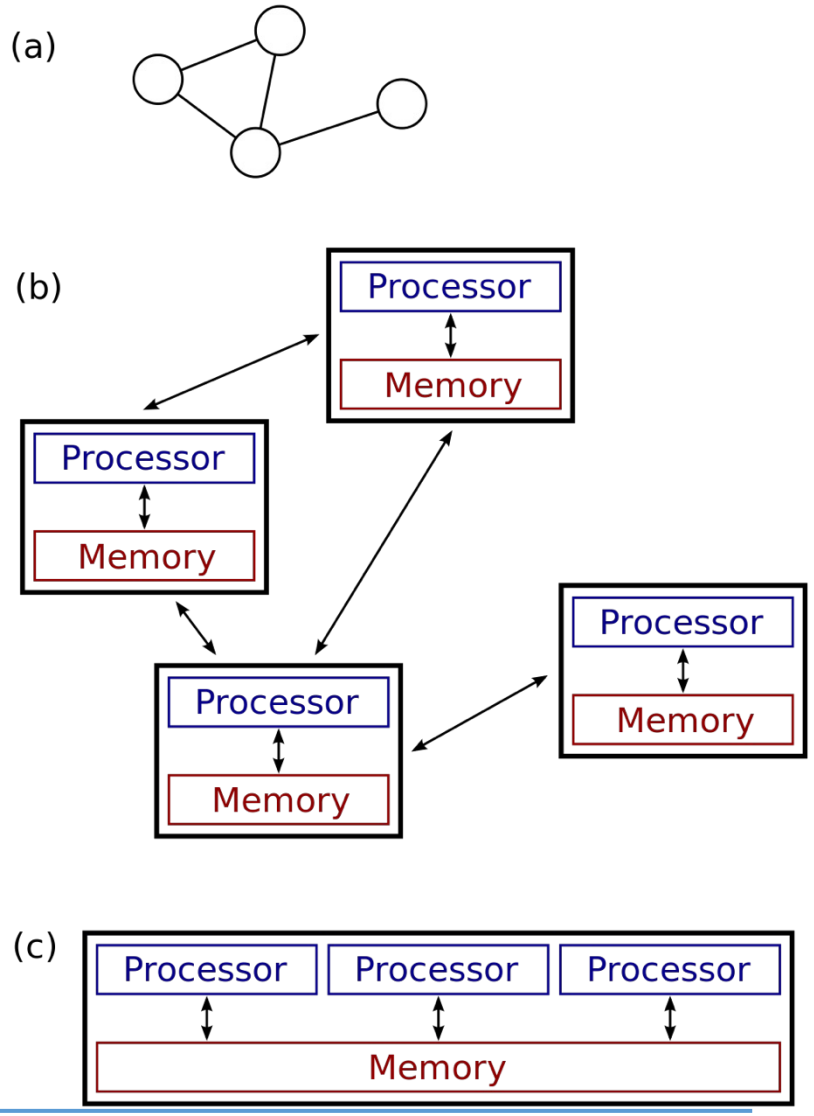
- Having *large* system with *fault-tolerance* and *availability and reliability*
- Fault-tolerance
  - Even though a subsystem fails, the rest of it could manage the failure, and survive
  - Measure for reliability (how much can we use a system reliably?)
    i.e.) how could a system survive when a part of it fails?
- Availability
  - Providing multiple replicas to deal with failures
  - Measure for service continuity,
    i.e.) how much a system is available to provide service?
- Large data handling
  - Having large data much more than one system's capability

Into a distributed system

- DNS
  - Stores a mapping for (domain name, IP address)
  - Distributed over the Internet
  - If one fails, the other could provide the service
  - As large as Internet scale
- Cloud storage (dropbox, onedrive, etc.)
  - Stores a mapping for (remote file, local file)
  - Access file on the Internet server
  - You can have multiple local copy
- What facebook, google uses are set of computers; not a super computer
  - Making a larger one with smaller computers

Distributed systems

- Consists of multiple computers
- Goals
  - For scaling-out: handle data more than a computer can do
  - For efficiency: multiple computers work concurrently
  - For reliability: more reliably operate under harsh environment
- Challenges
  - Communication among the nodes (with who and how?)
  - Heterogeneity among the nodes (Each of them can use different HW, OS, SW)
  - Consistency among replicas (How to keep consistency?)
  - Some of them can fail (How to survive?)



Reliability and distributed systems

- Some computers may fail during operation
  - How to isolate failure? / How to keep consistency? / How to recover from failure?
- Make some duplicate copy & update from multiple sites
- Two-phase commit
  - Commit-request phase (voting phase)
  - Commit phase (completion phase)
- Assumes commit manager
  - Commit req / ack

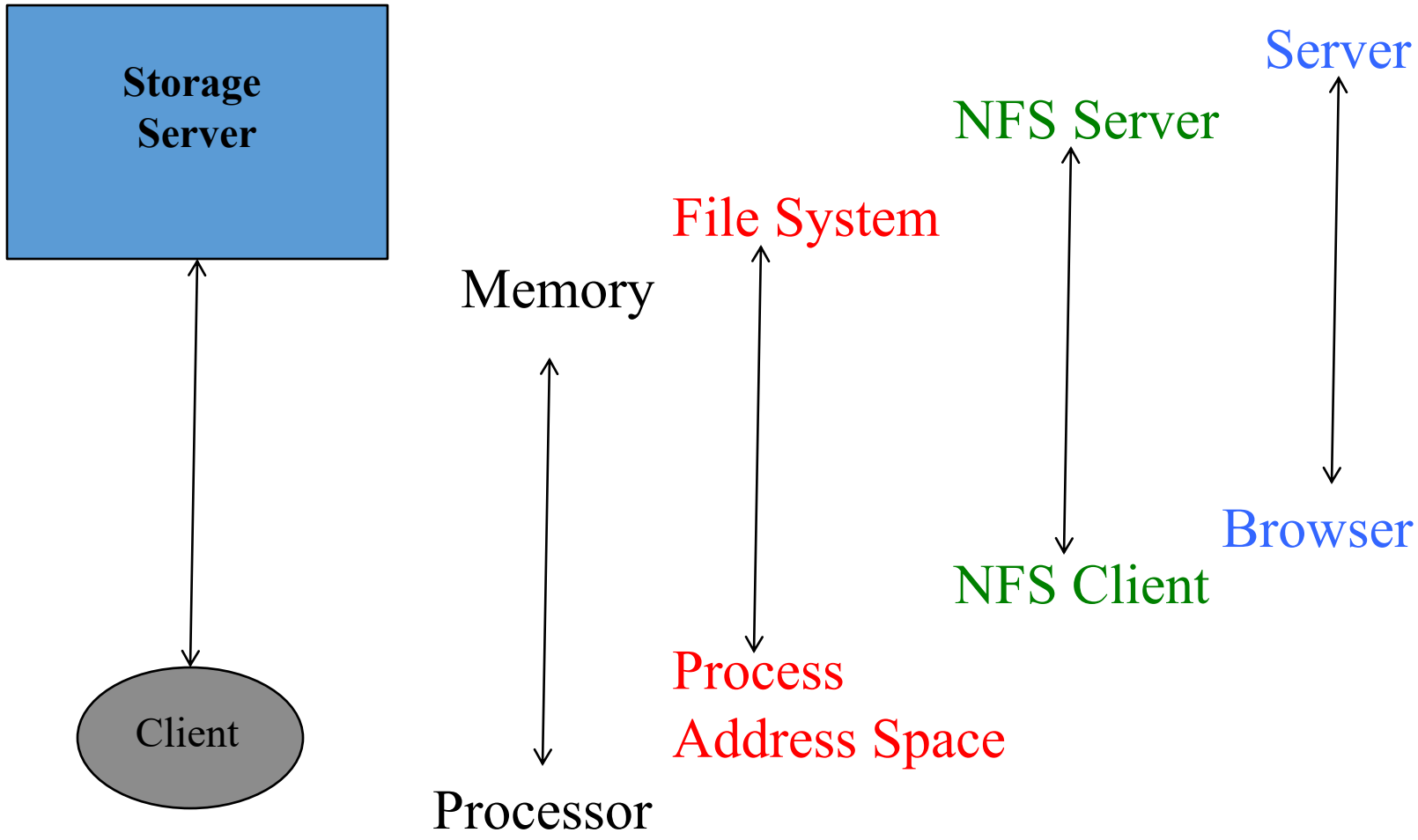
What is the key to performance AND reliability?

Replication

What is the source of inconsistency?

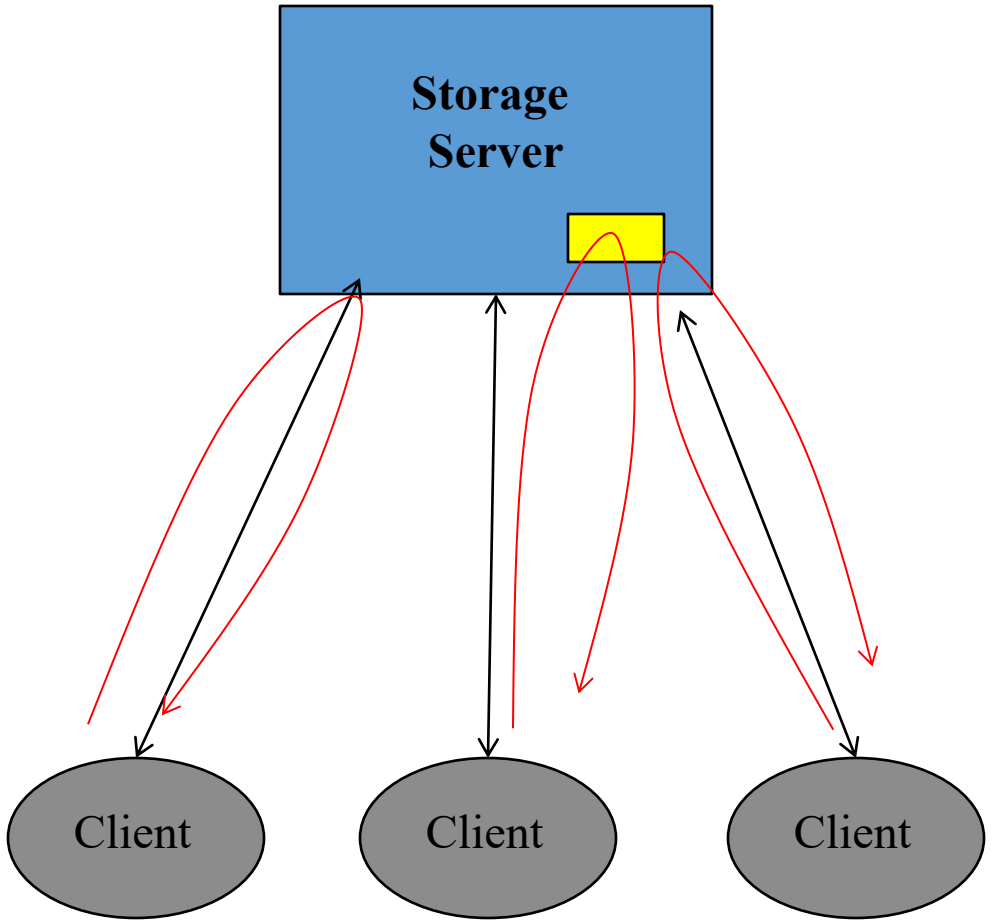
Replication

Any Storage Abstraction



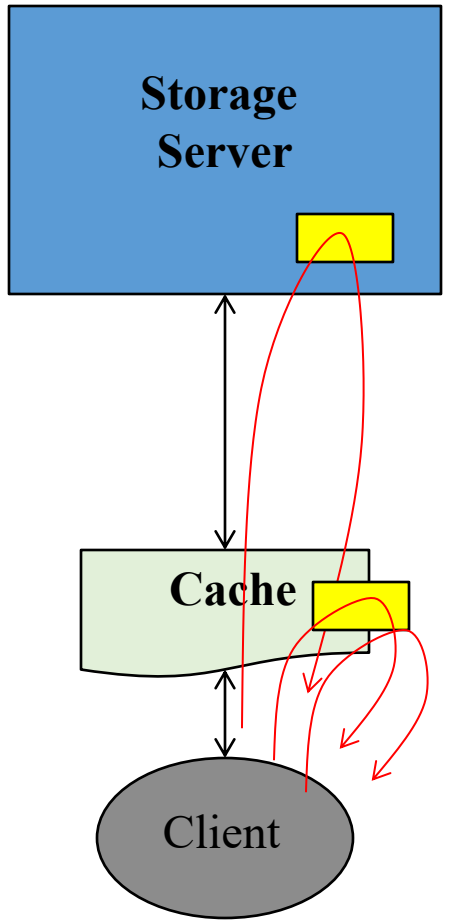
Multiple Clients access server: OK

- But slow



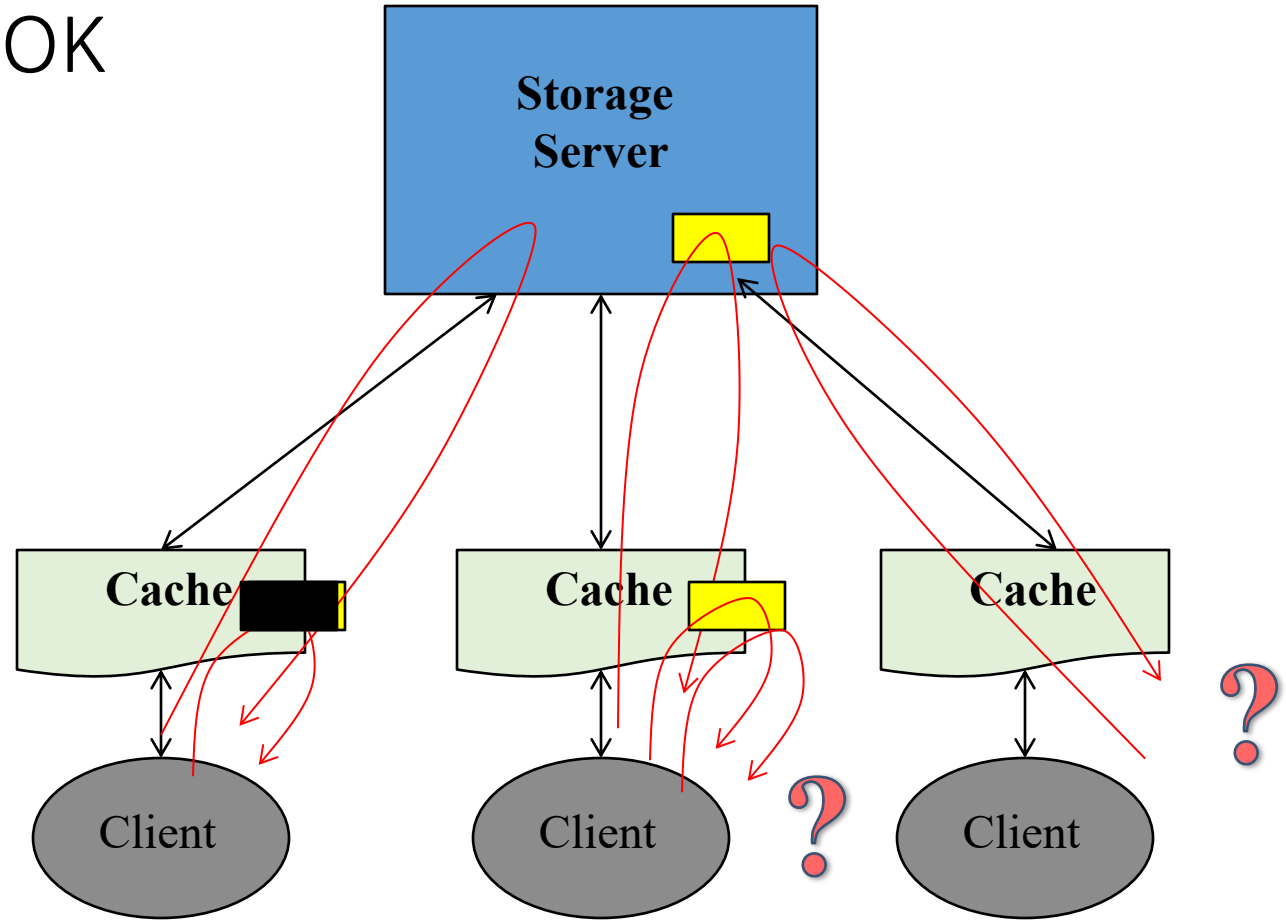
Multi-level Storage Hierarchy: OK

- Replication with in storage hierar chy
to make it fast



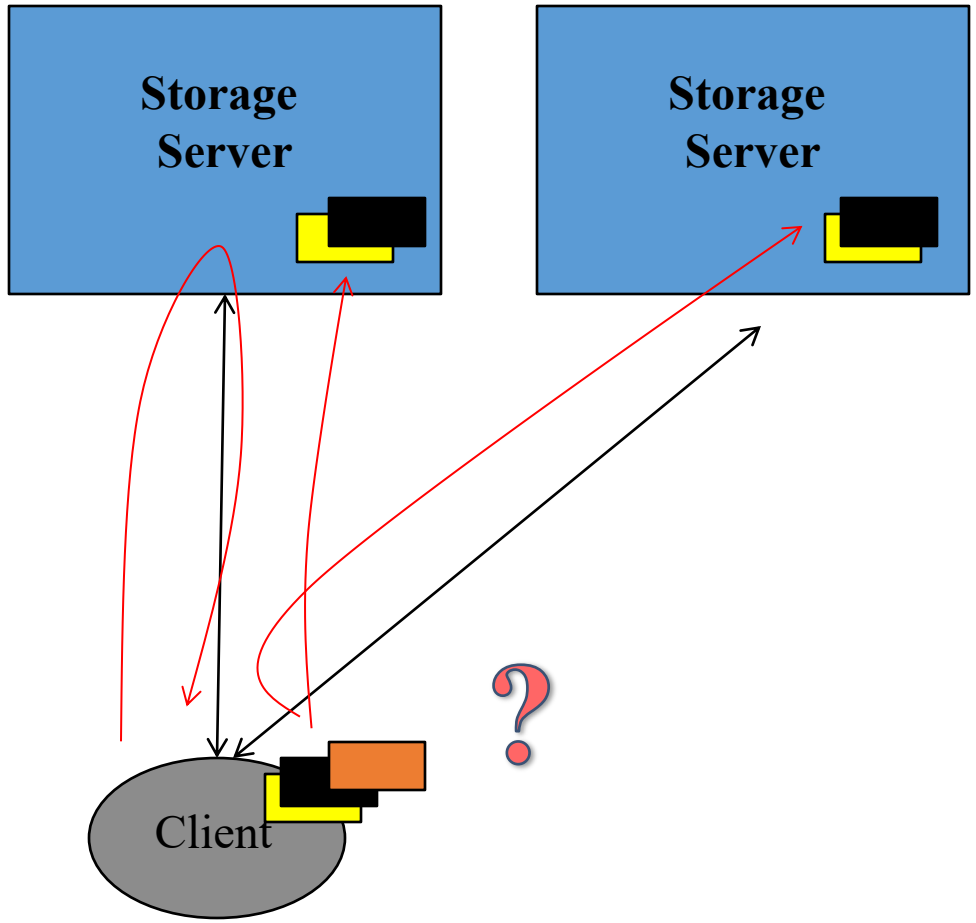
Multiple Clients and Multi-Level

- Fast, but not OK



Multiple Servers

- What happens if we cannot update all th e replicas?
- Availability => Inconsistency

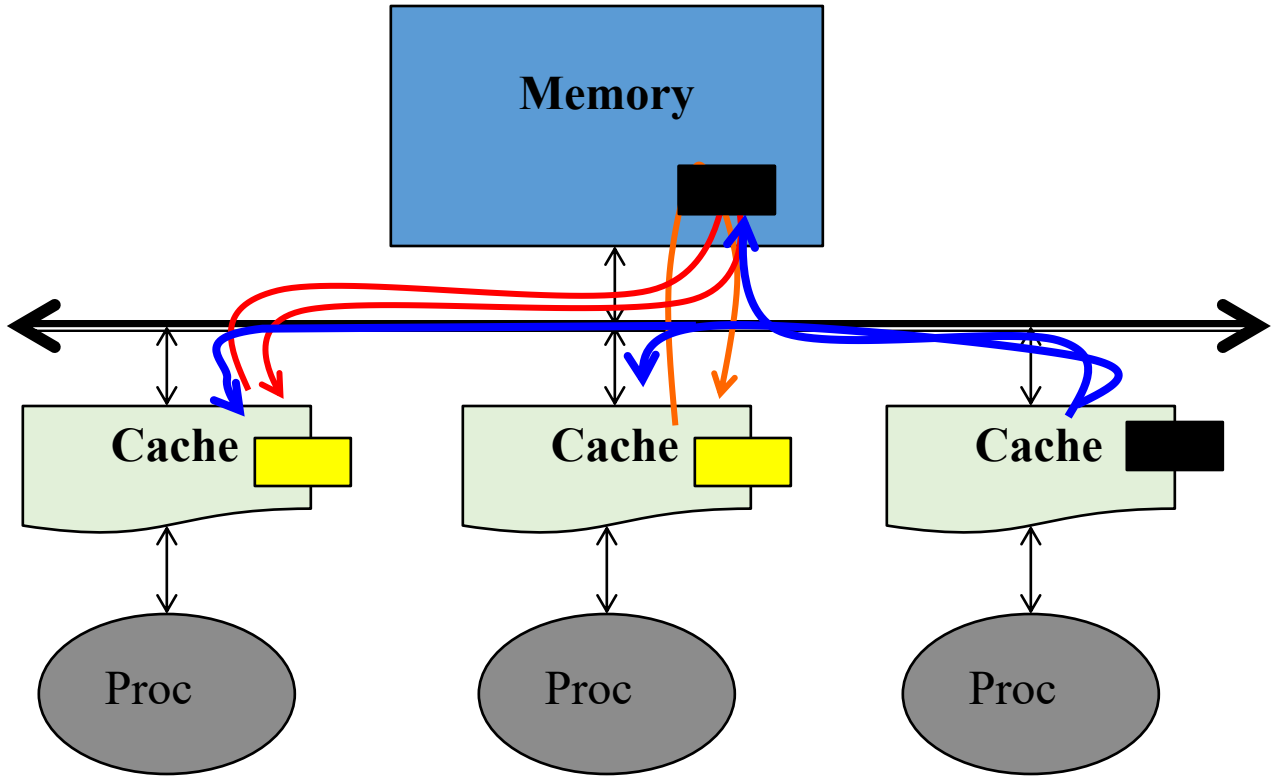


Basic solution to multiple client replicas

- Enforce single-writer multiple reader discipline
- Allow readers to cache copies
- Before an update is performed, writer must gain exclusive access
- Simple Approach: invalidate all the copies then update
- Who keeps track of what?

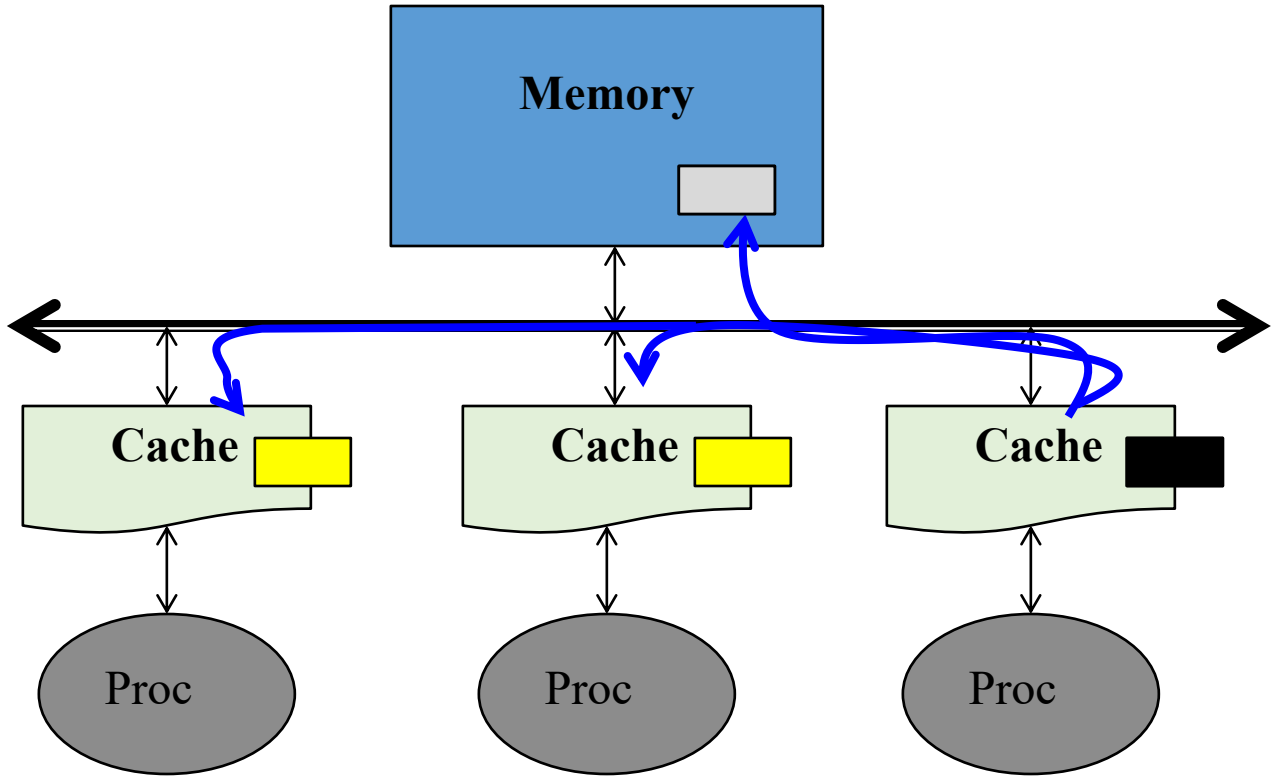
The Multi-processor/Core case

- Interconnect
  is a broadcast medium
- All clients can observe
  all writes and invalidat
  e
  local replicas (write-thr
  u
  invalidate protocol)



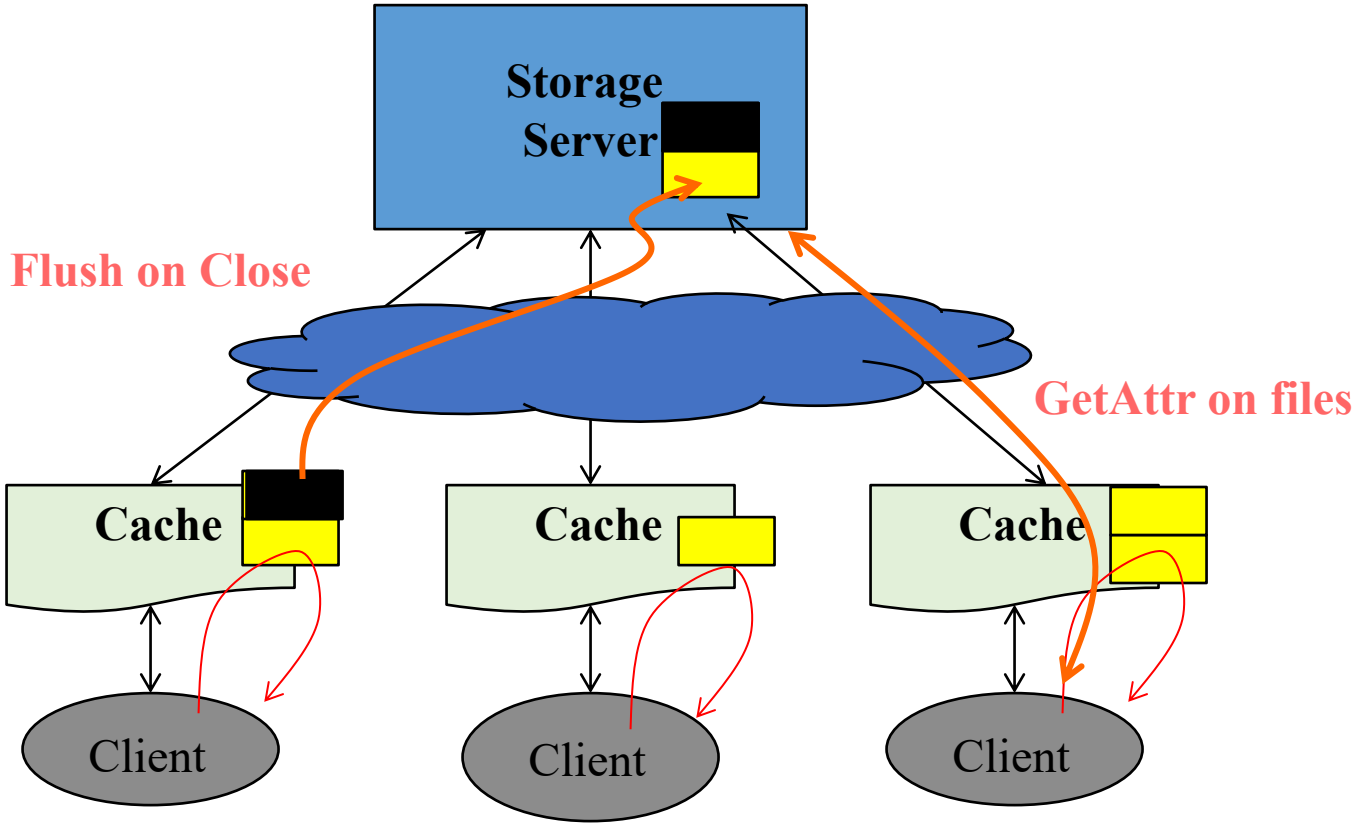
The Multi-processor/Core case

- Write-Back via read-exclusive
- Atomic Read-modify-write



NFS “Eventual” Consistency

- Stateless server allows multiple cached copies
  - Files written locally (at own risk)
- Update Visibility by “flush on close”
- GetAttributes on file ops to check modify since cac he



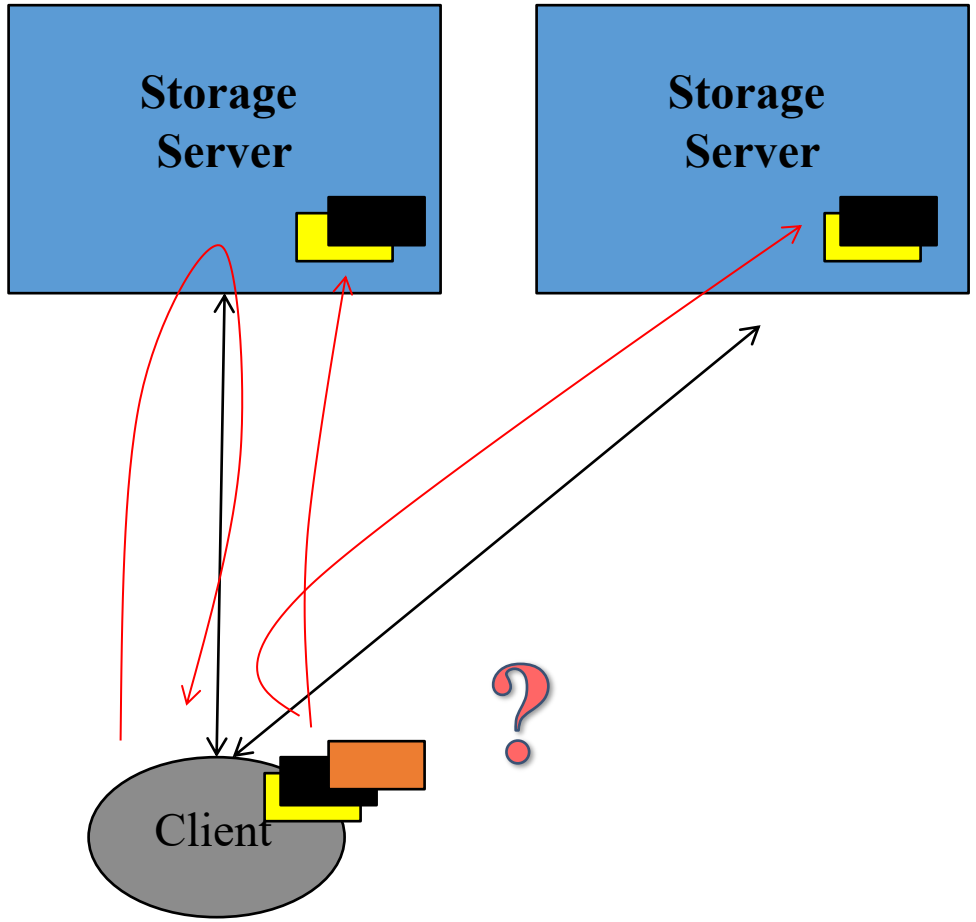
Other Options

- Server can keep a “directory” of cached copies
- On update, sends invalidate to clients holding copies
- Or can send updates to clients
- Pros and Cons ???

- OS Consistency = Architecture Coherence
  requires invalidate copies prior to write
- *‘Write buffer’* has be to be treated as primary copy
  - like transaction log

Multiple Servers

- What happens if cannot update all the replicas?
- Availability => Inconsistency



Durability and Atomicity

- How do you make sure transaction results persist in the face of failures (e.g., server node failures)?

- Replicate store / database
  - Commit transaction to each replica

- What happens if you have failures during a transaction commit?
  - Need to ensure atomicity: either transaction is committed on all replicas or none at all

Two Phase (2PC) Commit

- 2PC is a distributed protocol

- High-level problem statement
  - If no node fails and all nodes are ready to commit, then all nodes COMMIT
  - Otherwise ABORT at all nodes

- Developed by Turing award winner Jim Gray (first Berkeley CS PhD, 1969)

2PC Algorithm (agreement algorithm)

- One coordinator
- N workers (replicas)

- High level algorithm description
  - Coordinator asks all workers if they can commit
  - If all workers reply "VOTE-COMMIT", then coordinator broadcasts "GLOBAL-COMMIT",
    Otherwise coordinator broadcasts "GLOBAL-ABORT"

  - Workers obey the GLOBAL messages

Detailed Algorithm

Coordinator Algorithm

Coordinator sends VOTE-REQ to all workers

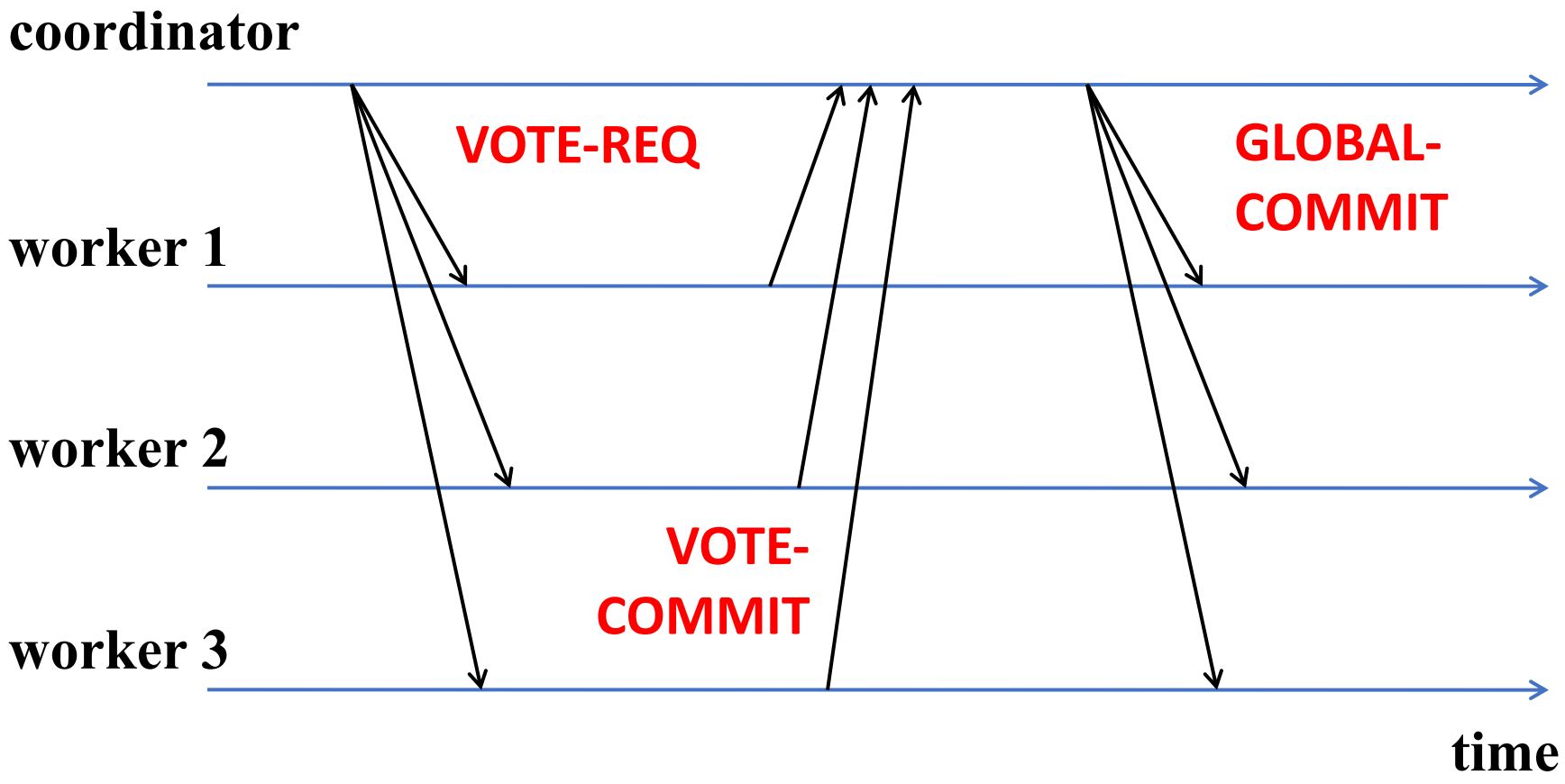
- If receive VOTE-COMMIT from all N workers, send GLOBAL-COMMIT to all workers
- If doesn't receive VOTE-COMMIT from all N workers, send GLOBAL-ABORT to all workers

Worker Algorithm

- Wait for VOTE-REQ from coordinator
- If ready, send VOTE-COMMIT to coordinator
- If not ready, send VOTE-ABORT to coordinator
  - And immediately abort

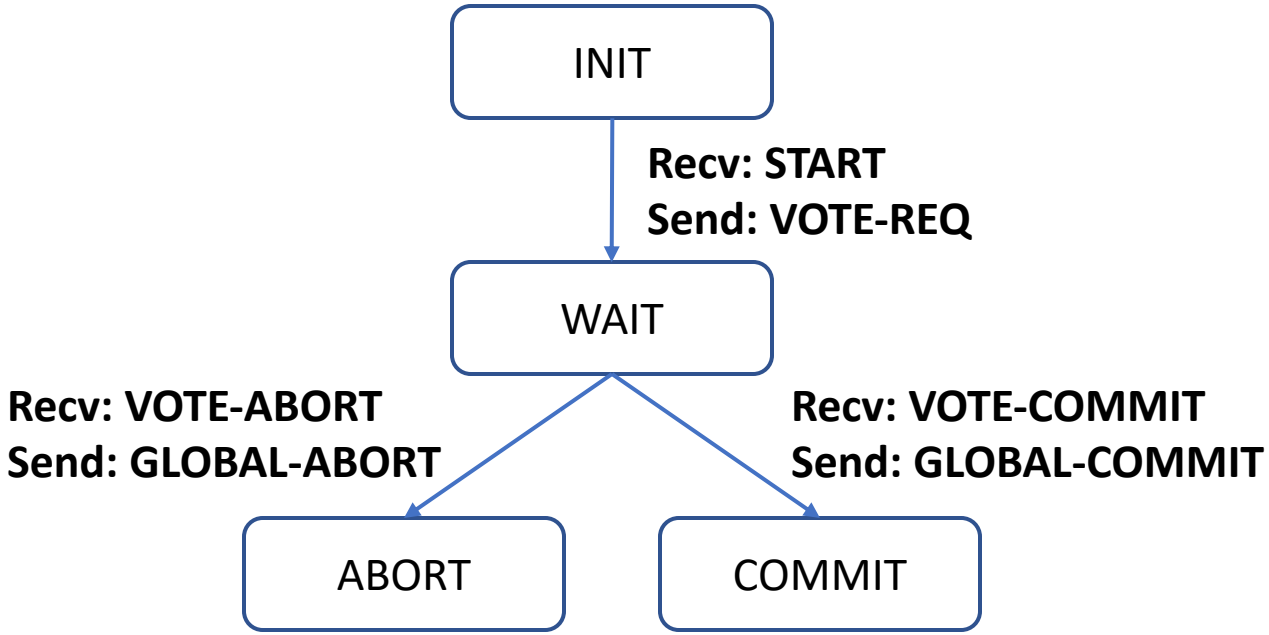
- If receive GLOBAL-COMMIT then commit
- If receive GLOBAL-ABORT then abort

Failure Free Example Execution

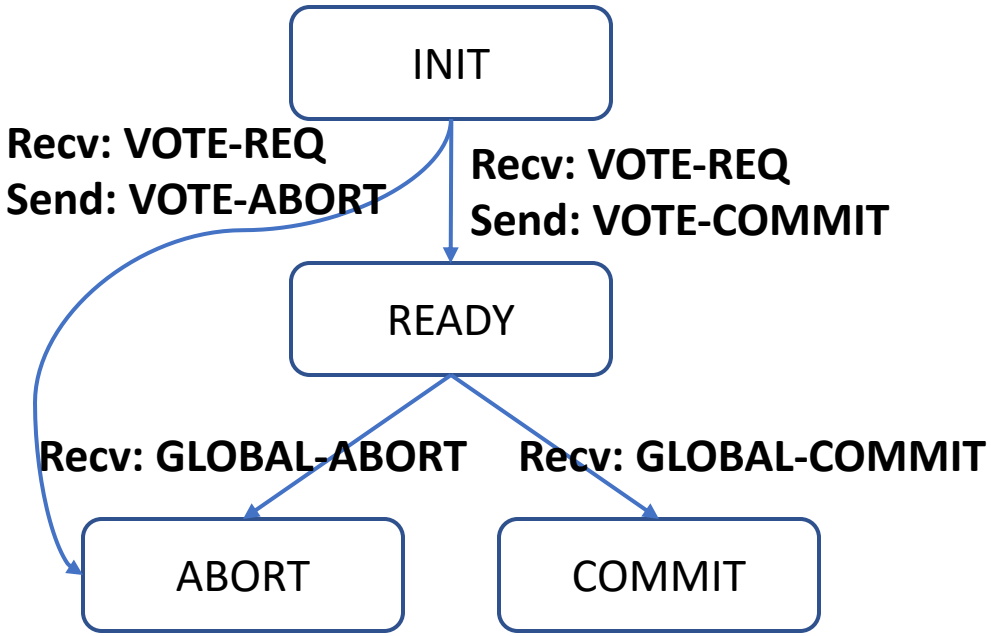


State Machine of Coordinator

- Coordinator implements simple state machine



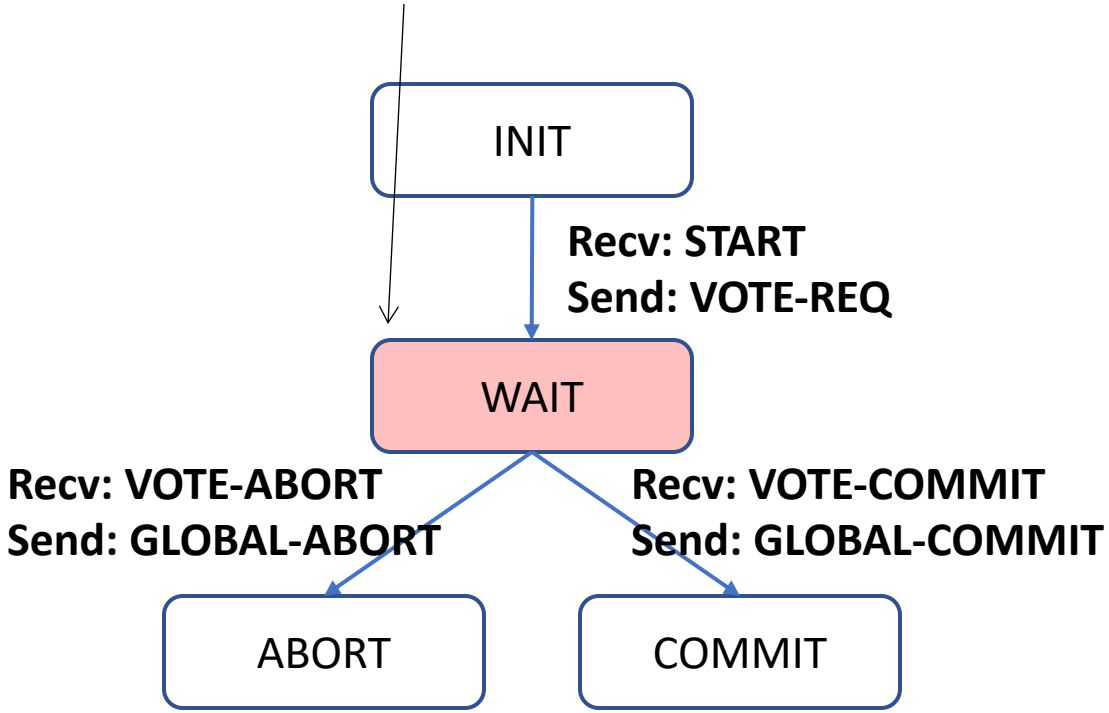
State Machine of Workers



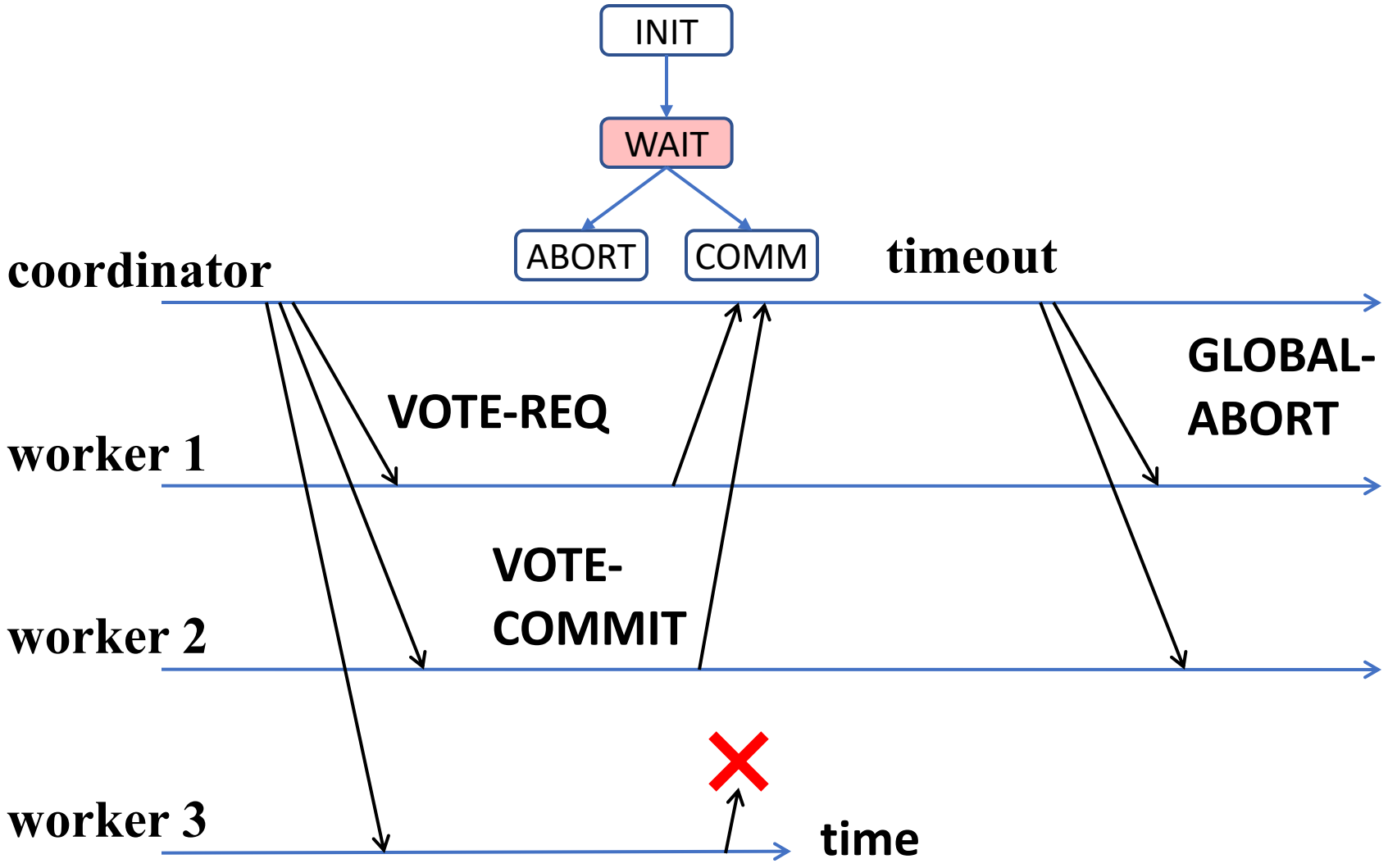
Dealing with Worker Failures

- How to deal with worker failures?
  - Failure only affects states in which the node is waiting for messages
  - Coordinator only waits for votes in “WAIT” state

- In WAIT,
  if doesn’t receive N votes,
  it times out and sends
  GLOBAL-ABORT

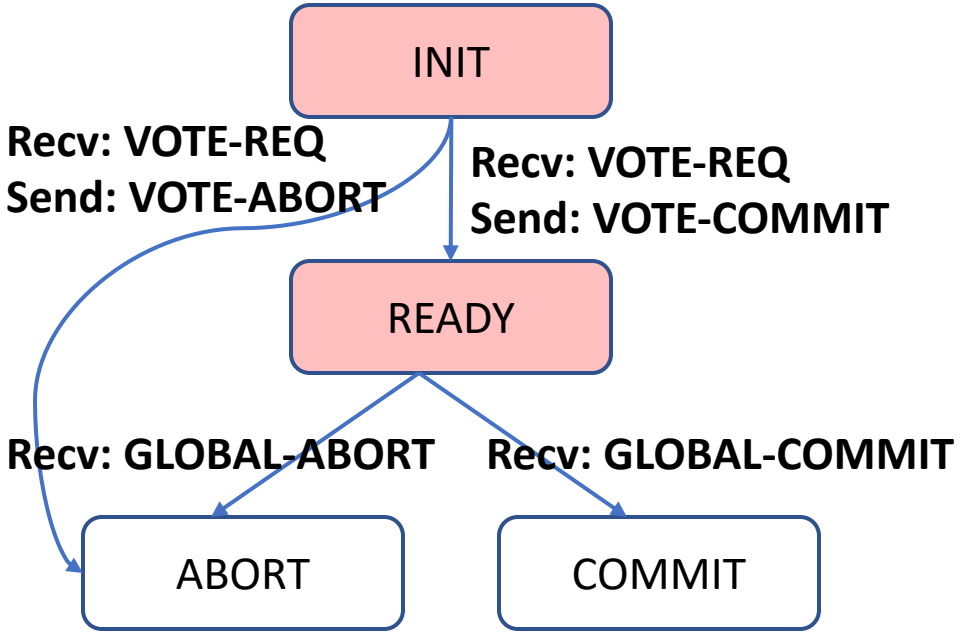


Example of Worker Failure

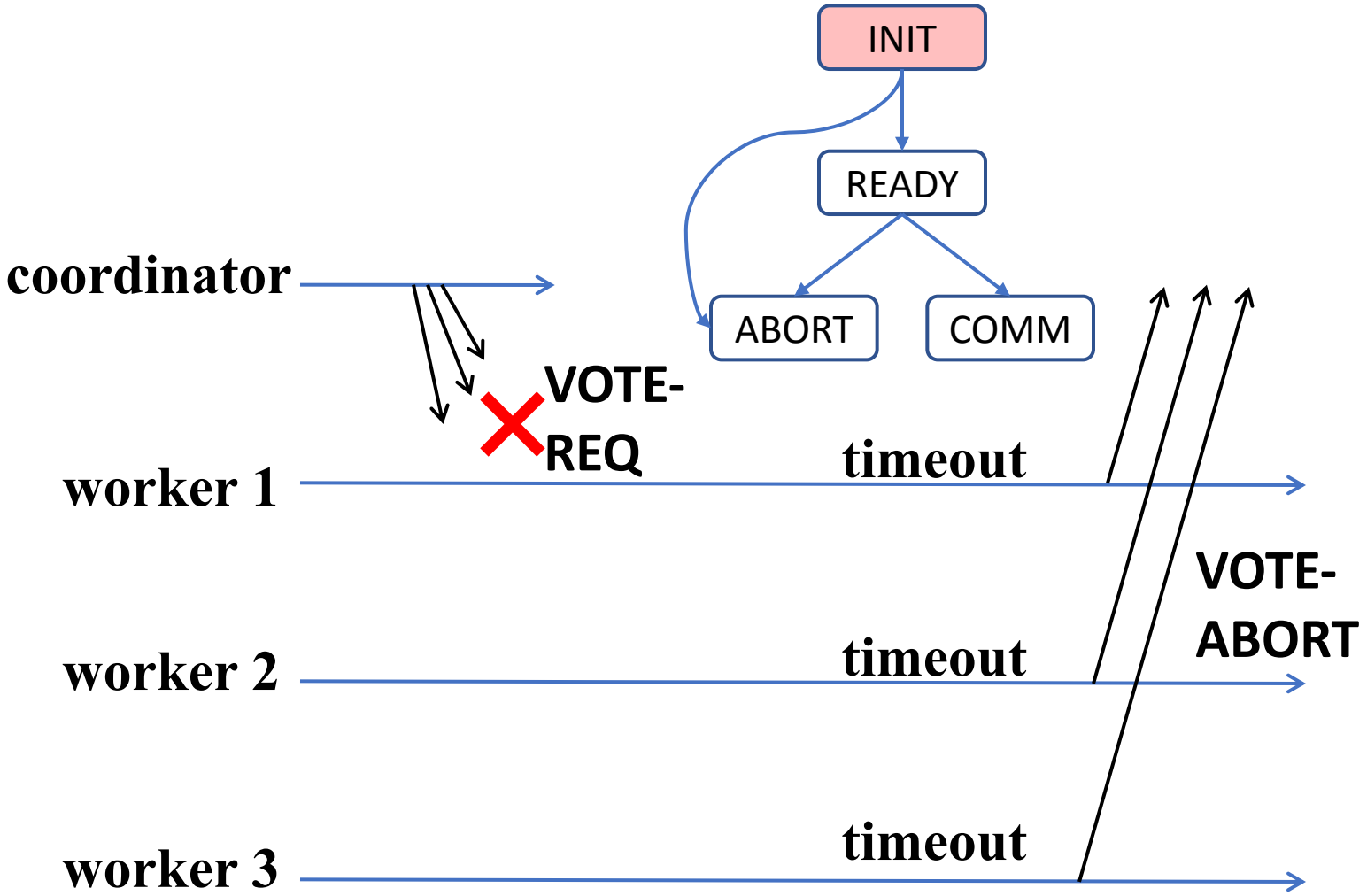


Dealing with Coordinator Failure

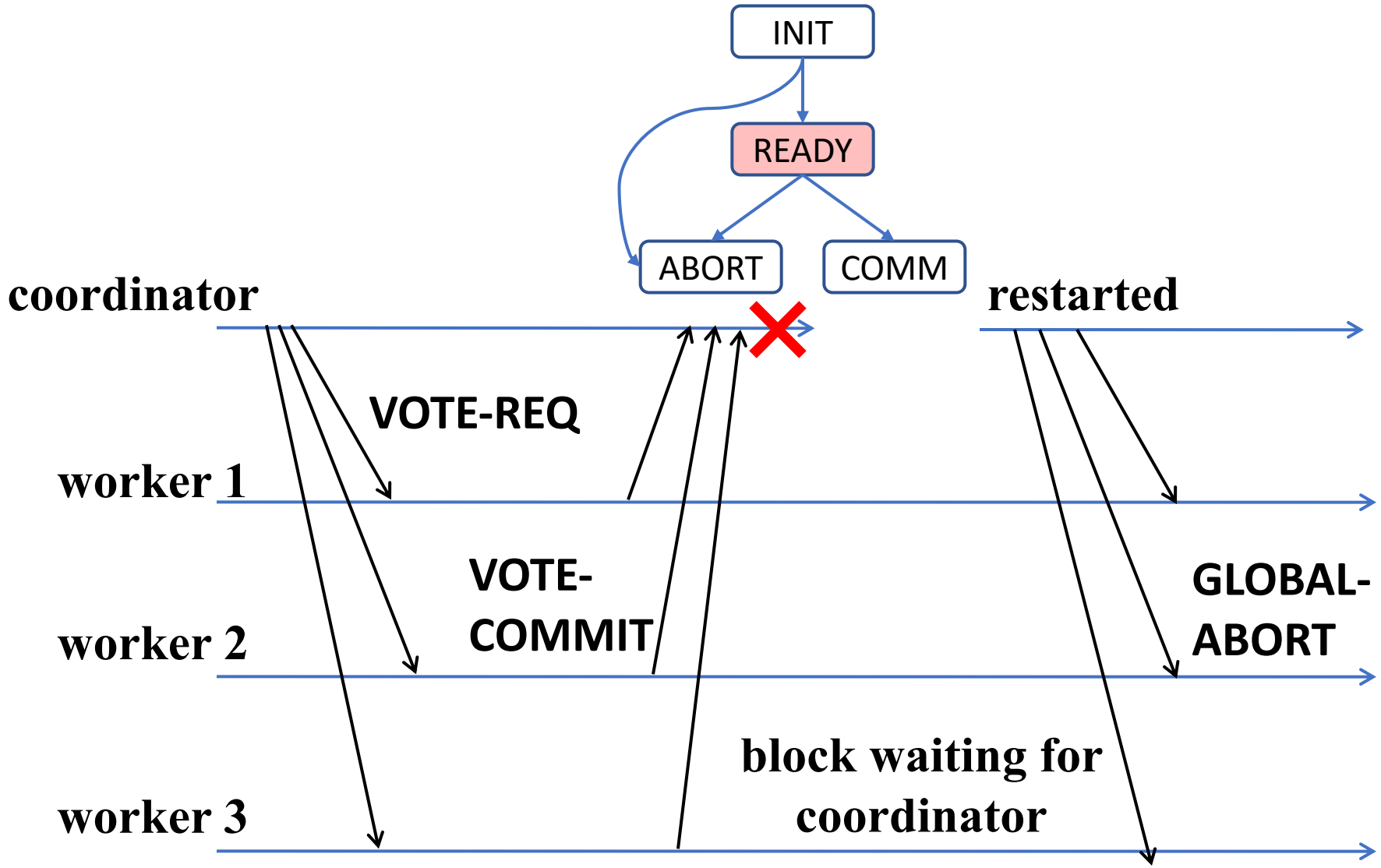
- How to deal with coordinator failures?
  - worker waits for VOTE-REQ in INIT
    - Worker can time out and abort (coordinator handles it)
  - worker waits for GLOBAL-* message in READY
    - If coordinator fails, workers must **BLOCK** waiting for coordinator to recover and send GLOBAL_* message



Example of Coordinator Failure #1



Example of Coordinator Failure #2



Durability

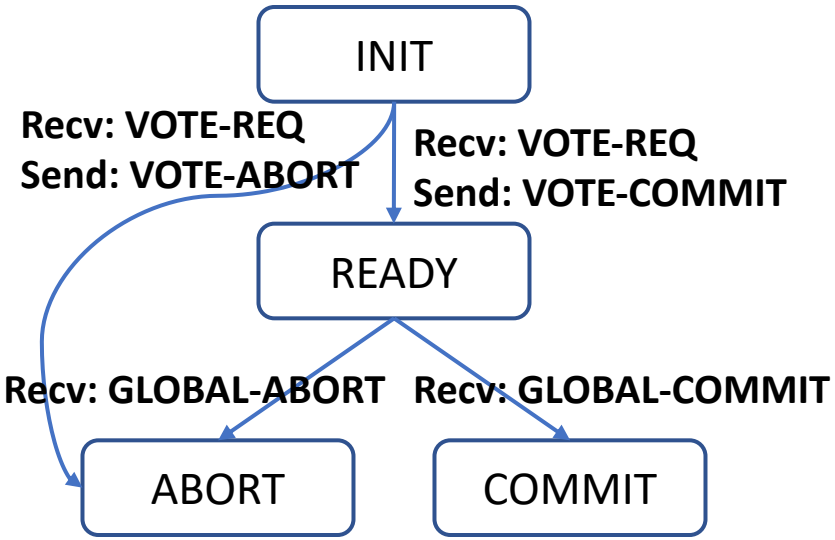
- All nodes use stable storage* to store which state they are in

- Upon recovery, it can restore state and resume:
  - Coordinator aborts in INIT, WAIT, or ABORT
  - Coordinator commits in COMMIT
  - Worker aborts in INIT, ABORT
  - Worker commits in COMMIT
  - Worker asks Coordinator in READY

* - stable storage is non-volatile storage (e.g. backed by disk) that guarantees atomic writes.

Blocking for Coordinator to Recover

- A worker waiting for global decision can ask fellow workers about their state
  - If another worker is in ABORT or COMMIT state then coordinator must have sent GLOBAL-*
  - Thus, worker can safely abort or commit, respectively
  - If another worker is still in INIT state then both workers can decide to abort
  - If all workers are in ready, need to **BLOCK** (don't know if coordinator wa nted to abort or commit)



Communication between computers

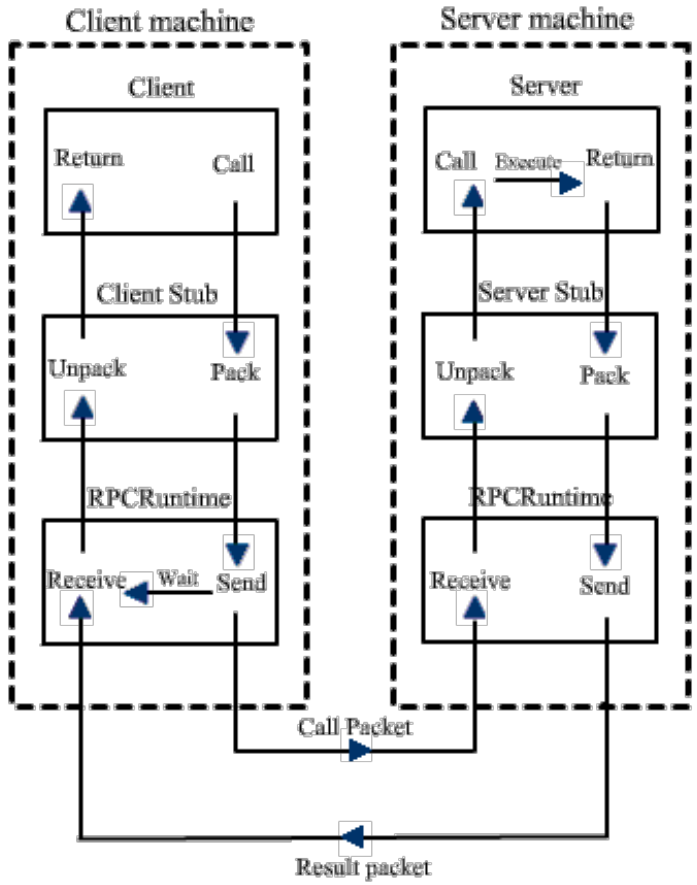
- Use Internet (TCP/IP), why not?
- Congestion control / flow control
  - Sliding window, advertising window
  - Sending more than one packet / RTT
- Reliable data stream transmission over unreliable links
  - Ack, sequence numbering
  - Retransmission with timeout
  - Check the packets reception in order
- A distributed system is a network program that runs on multiple computers
  - Enlarging the boundary of OS
    - File systems, networking, process, threads, virtual memory, etc.
- Now, you can have PB+ data on the network, larger than your local storage

To deal with heterogeneity,

- Heterogeneity could be
  - Endianness, data packing / presentation in hardware
  - Software interface (system calls, API)
- RPC (remote procedure call)
  - Run a procedure in remote computers, as if the computer is a local one
  - More functional than simple message passing
  - More efficient than using a shared memory over the networks
- Tightly coupled with programming language
  - Stub code generation at compile time
    - Static, interface used (compiled) at both sides
  - Binding at run time
    - Server can be changed even during the run-time

Remote Procedure Call

- A. Birrell & B. Nelson, Implementing remote procedure calls, ACM Trans. of Computer Systems 1984
- Two sites (server / client)
  - Client sends a procedure ID (function ID)
  - Client sends arguments and corresponding data
  - Server receives RPC messages, execute the procedure
  - Server sends back the result and corresponding data
  - Client receives the result as return value
- Client stub / server stub
  - Marshall data: Transparently sending data if arguments have pointers
- Server / client share call-able functions table or list
  - Use the same header files, to have the identical interface
  - Call-able server pool is exported before use



Implementation of RPC Mechanism

Network File System and RPC

- Design and Implementation of Sun Network Filesystem, USENIX 1985
- Remote file server provides OS interface, such as open, read, write, close
  - When a (client side) user requests to open a file, the remote computer operates, and return result
- Uses XDR (external data representation) for machine, OS independence
  - Defines size, byte order, alignment for basic typ
- Some more considerations
  - Crash recovery
  - Transparent access and interface to local UNIX file systems
  - Reasonable performance

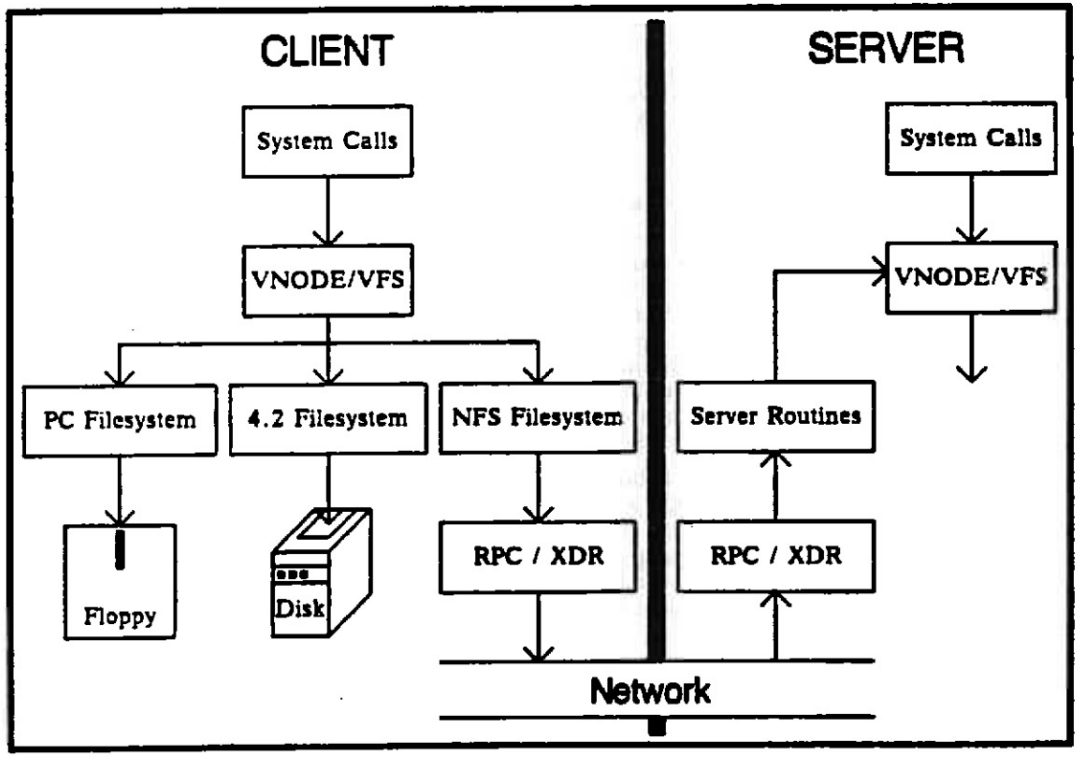
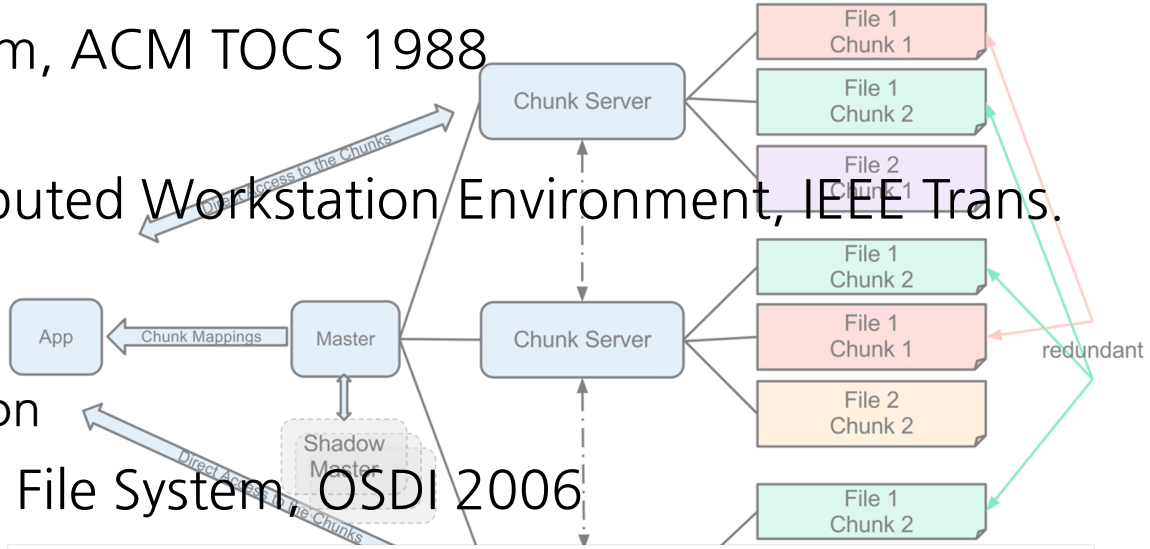
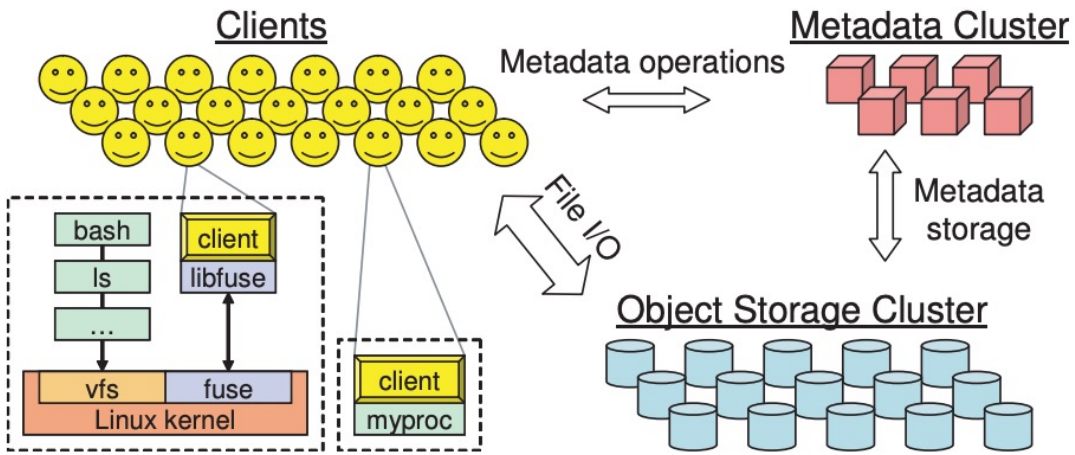


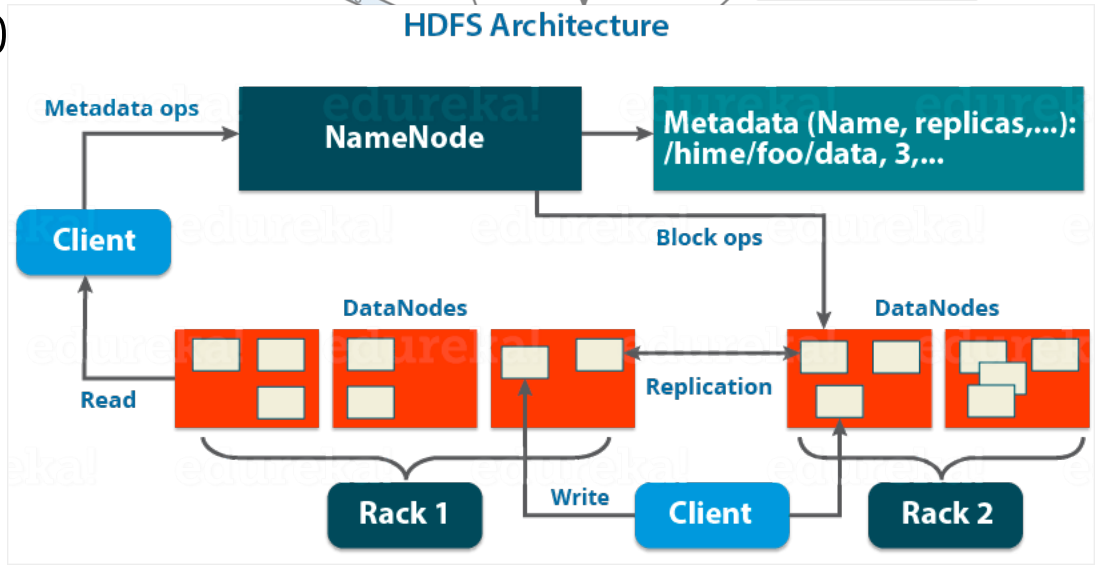
Figure 1

More distributed file systems

- Scale and Performance in a distributed file system, ACM TOCS 1988
  - Andrew file system by CMU
- Coda: A Highly Available File System for a Distributed Workstation Environment, IEEE Trans. on Computers 1990
- The google file system, ACM/Usenix SOSP 2003
  - Linux-based distributed file system implementation
- Ceph: A Scalable, High-Performance Distributed File System, OSDI 2006
- The Hadoop Distributed File System, MSST 2010







General view of distributed file systems

- Having replicas
  - Caching in the local machine
  - Redundant copy for availability, fault-tolerance
- One of them works as master, many workers
  - Manage mapping / assign jobs to computers
    - Scale out workload / load balancing among workers
  - Metadata / data separation
    - name nodes, metadata cluster
    - Chunk server, data nodes, object storage cluster
- Using VFS (virtual file system)
  - Identical operation interface with local files
  - Uses RPC/XDR for low-level compatibility

Moving Computation is Cheaper than Moving Data

- You can access data in remote site; or
- You can move to remote site, instead of moving data
  - You: computation / application
- Which do you think is cheaper?

Map-reduce by google

- MapReduce: Simplified Data Processing on Large Clusters, OSDI 2004
- Handling data for google data center
- Large-scale data processing (not only storing), but processing with thousands of nodes
  - Hello world example in map-reduce world, Word count on distributed server farm
  - Distribute file on network
  - -- Map phase
  - Count the words in local chunks
  - Stores a local map (Key, Value) for (word, count) in distributed file system
  - -- reduce phase
  - Reducer fetches (Key, Value) from distributed file system
  - Accumulate final (Key, Value)

So, easy. What's contribution?

- It can scale out. 1800+ nodes are used for text mining
- Deals with continuous input data stream
- Deals with partial failure, dynamic binding
  - There is a job tracker, which assigns mapper and reducers
- Deals with non-uniform data
  - Good for data processing from string input (such as logs)
- A developer can program mapper / reducer
  - Mapper can choose what to use in the system, refining data
  - Reducer gathers result, combining partial results

Hadoop, Spark, etc.

- Have some attention / interest on technology
- Very interesting era, reaching out industrial effort into daily innovation
- Regarding AI/ML, many knows Alphago; and recent huge trend in AI/ML applications
- However, they're based on the systems work
  TensorFlow: A System for Large-Scale Machine Learning, OSDI 2016 by google brain

Summary for distributed systems

- A separate semester-long topic
- Some systematic operations to make a larger system than single computer
- Picking a master
- Making replicas
- Keeping consistency
- Allow partial failures
- Allow transactions
- Distributed processing model / inter-communication model

Into the Cloud computing

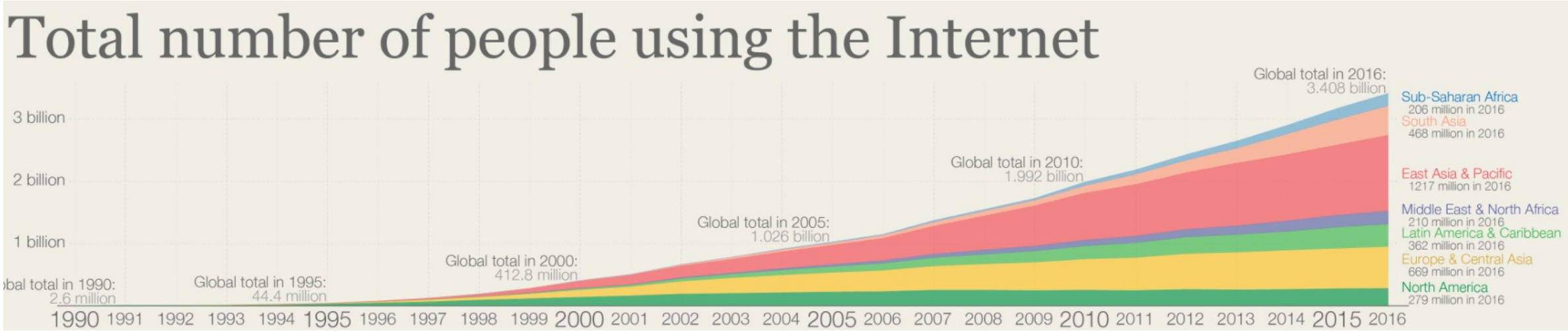
Cloud computing is at hand

- All students here have used cloud computing! (everyday)
  - Dropbox, Onedrive, google, naver, facebook, Office, etc.
  - Without knowing it (because it is server-side technology)
    - Likewise, you've used OS, before knowing what it actually does
- We all use it, and cloud computing is de facto standard in servers
  - why?

- BTW. We will focus on what/why/how questions
  - What it is / what it is not
  - Why we need it / what happens if we don't have it
  - How can we implement it
  - Logical reasoning, based upon the understandings of technical abstractions required

A motivation: Internet service at scale

- Internet populations: 3.4B users
  - https://ourworldindata.org/internet



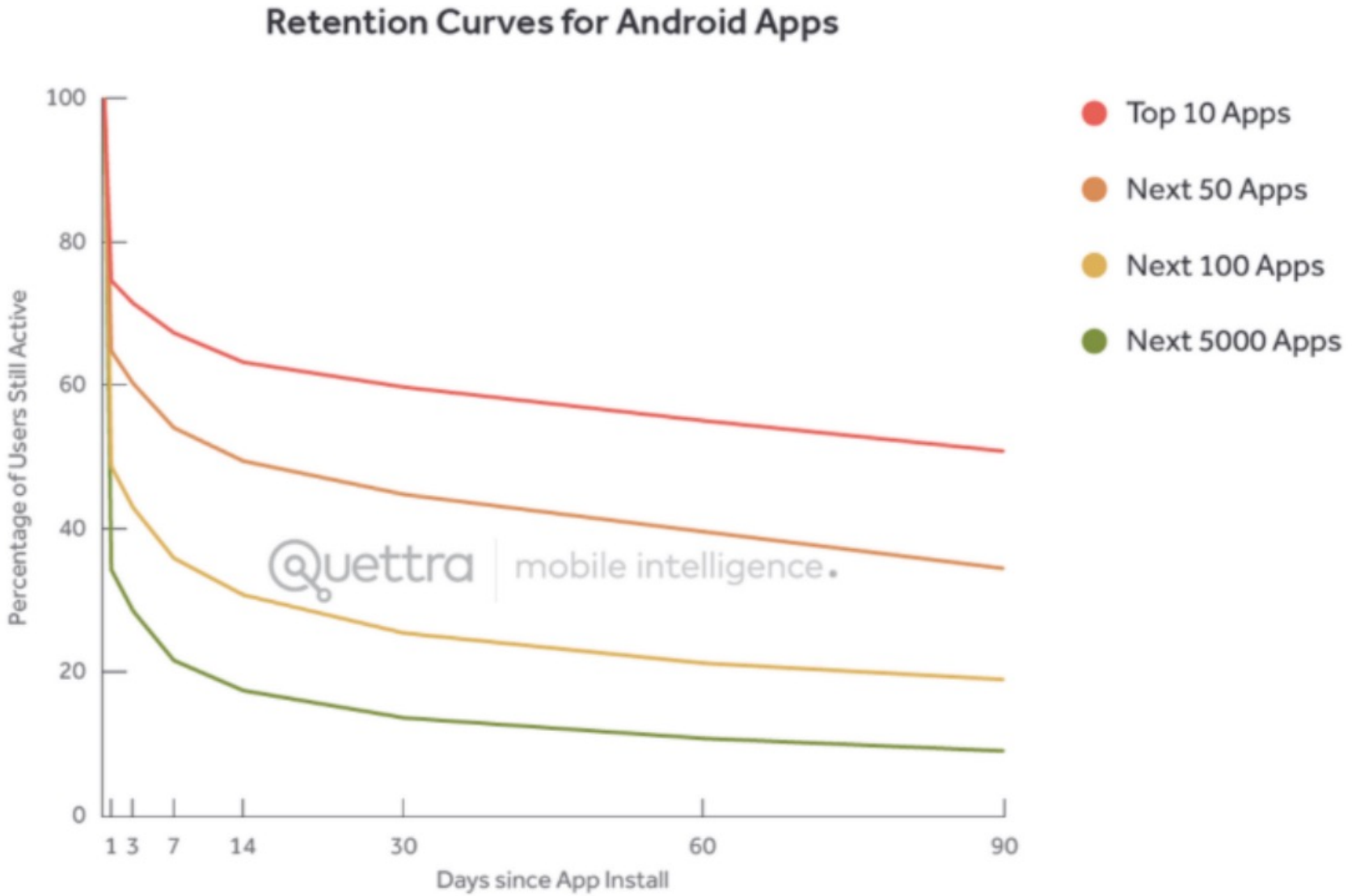
- Thinking about a querying system (google/naver) for 100 users
  - Web server - a threaded programming dealing with 100 connections
  - How about 1M users?
  - How about 3Billion users?
  - ➔ has to be scalable (be able to deal with larger clients' requests)

Two ways of scaling

- Having a more powerful server, having more servers
  - A.k.a. vertical / horizontal scaling
  - Super-computers / cluster system
- Pros/cons of different scaling
  - Cost-effectiveness
  - Fault-tolerance,
  - Consistency,
  - Communication cost,
  - Easy of programming
- Which is better?
  - Your conclusion?

Limitation of having more servers

- Buying more and more servers,
  would not make sense in normal business env.
  - IT service has ups and downs
  - What if… you buy 100 more servers, and your service drops after 15 weeks?
- Total Cost of Ownership (TCO)
  - Cost of owning servers for operation
  - Not decreases, even if the service discontinued
    - Energy, management & operation cost, floor plan, etc.
  - Why not borrow servers?
    - A.k.a. borrowing books from library, instead of purchasing books



Source: Quettra

Web hosting service

- If you have 100 server systems, utilizing only 60% of them
  - How about lent your servers to some other who need it?
  - Get some amount of money for managing it
  - → Now, you're hosting servers
- How about hosting a server with some basic software stack?
  - Linux, Apache web server, Mysql database, Php engine installed (LAMP)
  - With some more money
- That's what web hosting biz operators did
  - 카페24, 가비아, godaddy, inmotion

And more?

- What's server?
  - Is that a hardware? Or a software stack?
- Webservers, still underutilized…?
  - How about packing them into a software stack?
  - Running two apache servers on a single machine?
- Virtual machines and containers
  - A logical computer
  - Consolidating multiple VMs' workloads

Now, server is a flexible computing unit

- Could mean single hardware
  - Hardware + software
  - Part of hardware + software stack
  - Multiple hardware
  - Multiple hardware + software stack, if you want
- ➔ Flexible utilization of hardware (elastic utilization)
  - Scalable to the given workload
  - Could be over the limitation of a physical server
  - Could be fault-tolerant
  - Pay per-use model (utility computing)
  - Use computer as a service (charge, as much as you use)

Our approach to Cloud computing

- All students here have used cloud computing! (everyday)
  - Dropbox, Onedrive, google, naver, facebook, Office, etc.
  - Without knowing it (because it is server-side technology)
    - Likewise, you've used OS, before knowing what it actually does
- We all use it, and cloud computing is de facto standard in servers
  - why?

- BTW. We will focus on what/why/how questions
  - What it is / what it is not
  - Why we need it / what happens if we don't have it
  - How can we implement it
  - Logical reasoning, based upon the understandings of technical abstractions required

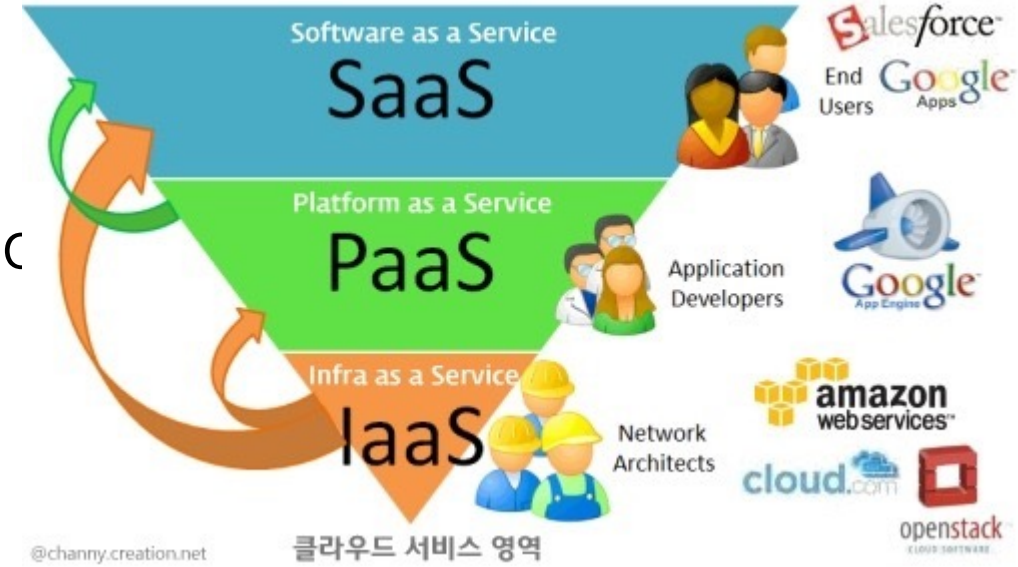
Cloud computing Architecture & services

Cloud computing: as a service

- Cloud makes a computing environment ‘as-a-service’
  - Because you borrow computing resources from the ‘service provider’
    - There is a service provider, who makes the cloud
    - There is a service user (customer or client), who uses the cloud computing resources
    - There are some resources that you can borrow
- What you can borrow from a cloud service provider
  - Some hardware disk capacity, cpu time, network bandwidth
  - Some mixture of the above - infrastructure
  - Some software of operating application services
    - web server, database server, operating systems
  - Some mixture of the above - platform
  - Some high-level abstraction of the computing - software

As-a-service

- SAAS
  - Software as a service
  - Office365, salesforce, WordPress
  - You are the end user, and a software develc
- PAAS
  - Platform as a service
  - LAMP web platform, web hosting
  - You need to manage and operate the service
- IAAS
  - Infrastructure as a service
  - Dropbox, google drive
  - You need a platform to drive hardware



출처 : http://blog.creation.net/506

Players in cloud computing

- Cloud Service Provider
  - Who makes the cloud computing infrastructure
- Cloud Consumer/ Cloud Service Owner (service vendor)
  - Who uses the cloud service
- Cloud networks (Cloud Carrier)
  - Who connects the cloud provider and the client devices
- Resource admin / Auditor
  - Who monitoring the resource utilization, auditing
- Trust Boundary
- Service brokers
  - Service discovery / distribution, inter-connects service providers to span services over the pl atforms

Essential characteristics of cloud computing

- On-demand self-service
  - Auto-scaling as much as the system demands
- Broad network access
  - Wide range of network access coverage
- Resource pooling
  - Provisioning physical computing resources, reducing the run-time allocation overhead
- Rapid elasticity
  - Instant Capacity, performance provisioning
- Measured service
  - Traceability for utilizing resources

Ultimate advantage of cloud computing

- Cost-efficiency
  - Pay per-use service model
- Flexibility
  - Scale as much as you need (not more or less)
- Availability
  - Migrate resources to some other geographical region from natural disaster s
- Fast deployment
  - Instant platform/service provisioning

Service model

SaaS

PaaS

IaaS

Hosted applications

Development tools, database management, business analytics

Operating systems

Servers and storage

Networking firewalls and security

Data center Physical plant or building

More cloud service models

- Publicly available cloud service vs. cloud service for a specific organi zation
  - Kakao, KT, SKT have their own private cloud
  - You can make your own cloud service; when do you need it?
- Hybrid cloud
  - Mixture of public / private cloud
- Virtual private cloud
- Multi-cloud

Cloud architecture & SW architecture

Systems design

- What is a (computer) system?
  - Components / elements
  - Relationship (interaction) among elements
  - Goals / purpose
  - Boundary (environment)
  - Component can be a system
- Define the following systems
  - Network system
  - File system
  - Operating system
  - Virtual memory system
  - Cloud computing system

System Design

- Software design has some patterns (frequently appearing common properties)
  - Modular design
  - Object-oriented design
  - Relational database design
  - Layered design
  - Component design

Software architecture?

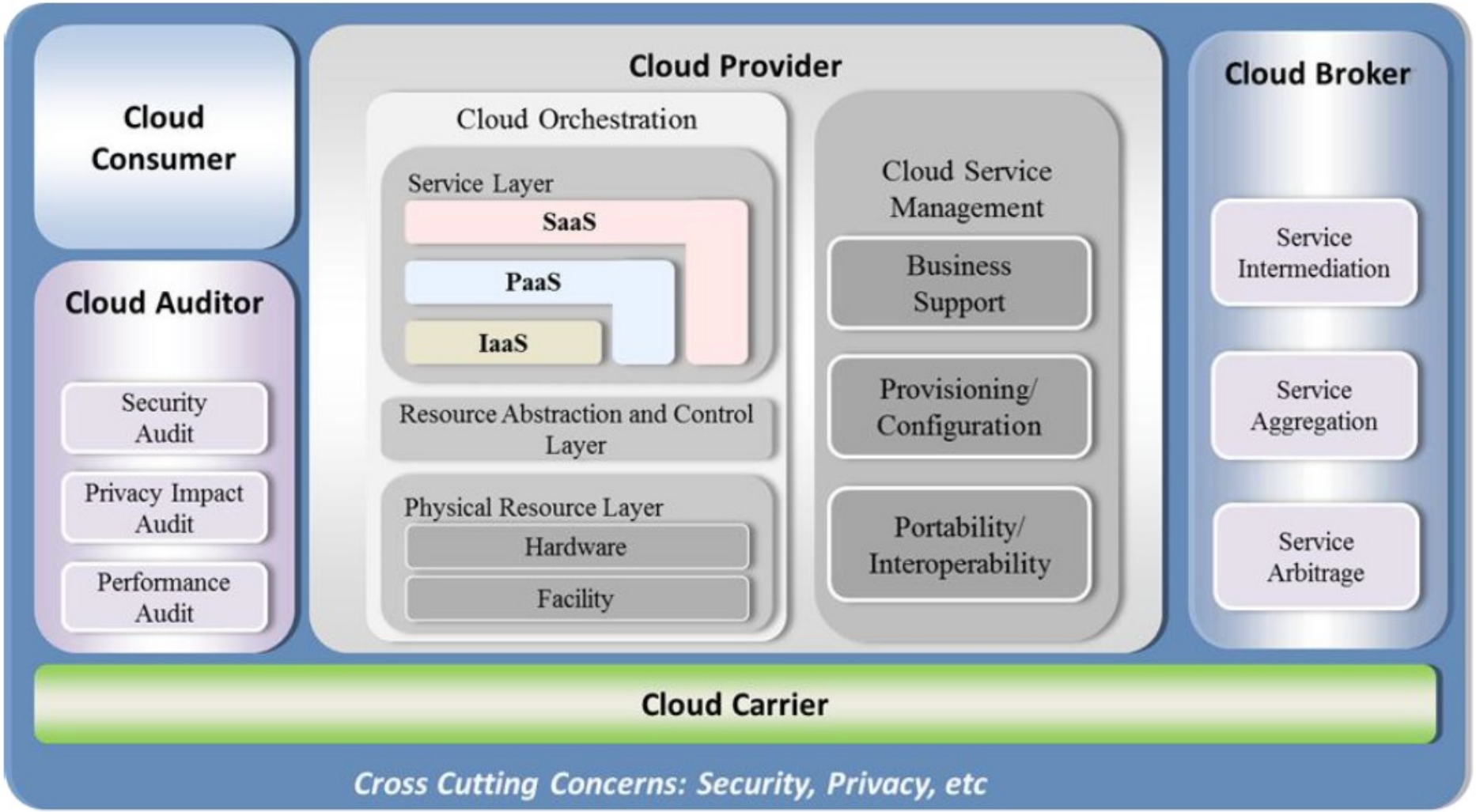
- Defines the structural property
  - Elements, relationship among elements
- Defines the interaction between elements
  - Interfaces among elements
- Defines the roles and responsibility
  - Boundary of elements
- Defines the dependency
  - Inter-dependent changes of a partial changes

➔ Learn UML

- How architecture is presented in programmers' language
- Class diagram, usage model, sequence chart, and design patterns

A reference architecture of cloud computing

- From NIST CCRA
  - Consumer
  - Auditor
  - Provider
  - Carrier
  - Broker



Cloud service provider is responsible for

- Service orchestration
  - Top layer: Service enabler
    - Define cloud service, service enabler
  - Middle layer: resource abstraction and controller
    - Provide abstraction to cloud resource, control access t o the resource
    - Hypervisor, virtual machine, virtual storage, virtual n etwork, management console
  - Low layer: physical resource pool
    - Provision physical computing resources and facilities
- Service management
  - Business support
  - Provisioning/configuration
  - Portability / inter-operability
- Privacy & Security

Cloud service broker

- Matchmaking between the service provider and the service consumers
- Service directory, discovery
  - Maintains the list of available services
- Service integration, inter-operation
  - Combine multiple services, spans over platforms
- Service provisioning
  - Guaranteed service for requested QoS
- Administration
  - Auditing, Reporting, security, support, billing, etc.
- Should prepare common interface for different services

Cloud carrier

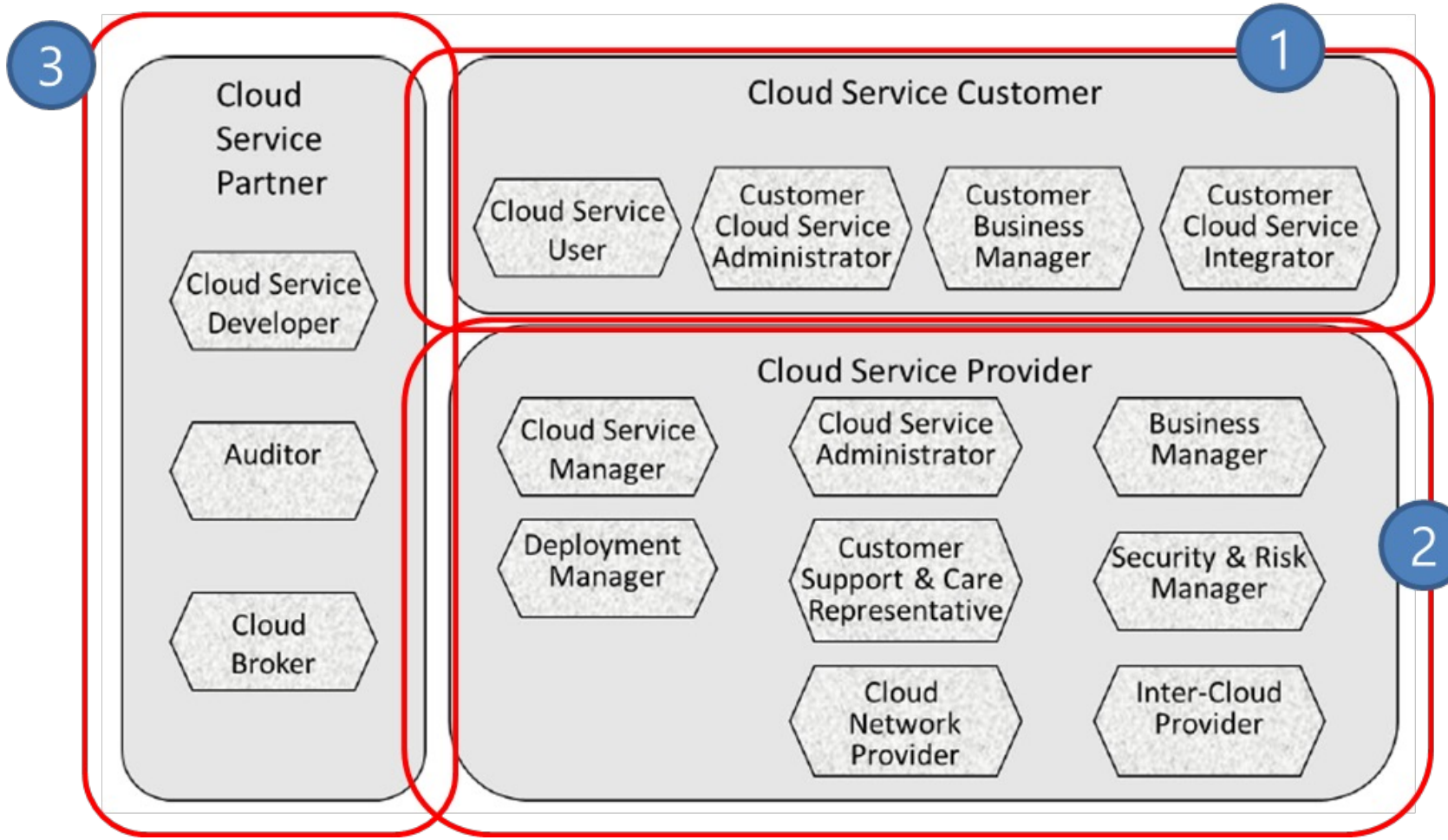
- Network support for service user and cloud provider

Security and privacy

- Collaborative responsibility
  - Cloud user and provider share the responsibility
- Physical security should also be considered
- Personal Information(PI), Personal Identifiable Information (PII) should be managed
  - Data life cycle management policy should be applied

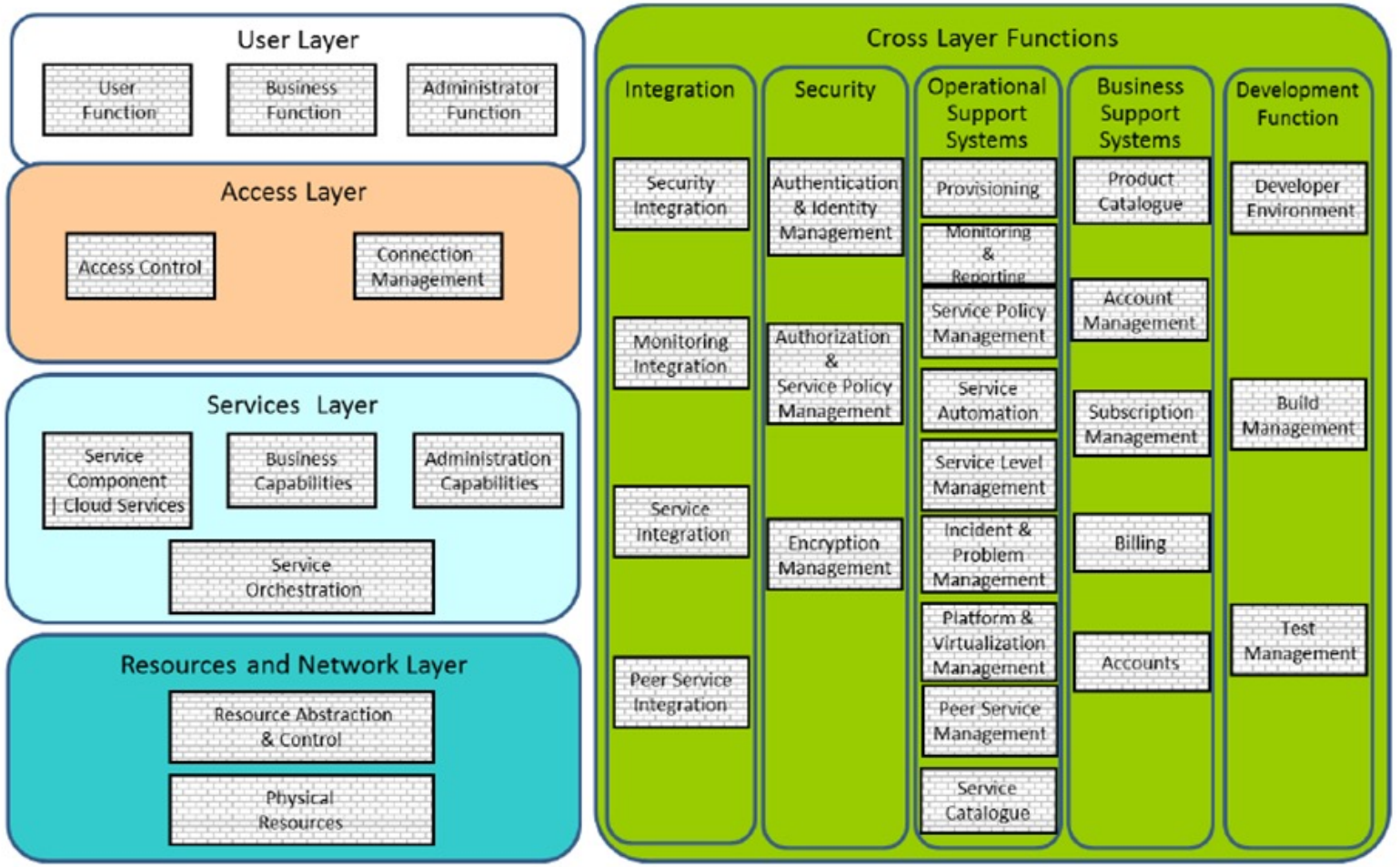
Another reference architecture

- From ISO/IEC
  - Cloud consumer
  - Cloud provider
  - Cloud service partner



Functions in ISO/IEC reference architecture

[클라우드컴퓨팅 참조 모델]



Cloud functionalities perspective

사용자 레이어 (User Layer)	•클라우드 서비스 고객을 표현 •클라우드 아키텍처 기능 레이어 사이의 상호작용 지원
접근 레이어 (Access Layer)	•서비스 레이어에서 사용할 수 있는 능력에 대한 수동 또는 자동 접근을 위한 인터페이스 제공 •암호화처리, 요청 무결성 검사
서비스 레이어 (Service Layer)	•클라우드 제공자에 의해 제공되는 서비스의 구현 포함 •소프트웨어 컴포넌트(하이퍼바이저, 호스트 운영체제, 디바이스 드라이버 제외)
자원과 네트워크 레이어 (Resources and Network Layer)	•물리적 자원이 거주하는 장소 •서버, 네트워킹 스위치와 라우터, 스토리지 장치와 같은 데이터 센터 •호스트 운영체제, 하이퍼바이저, 디바이스 드라이버, 일반적 시스템 관리 소프트웨어
크로스 레이어 기능 (Cross Layer Functions)	•지원능력 제공을 위한 레이어 상의 컴포넌트들과 상호작용을 하는 일련의 기능적 컴포넌트들을 포함 •운영 지원 시스템 기능, 비즈니스 지원 시스템 기능, 보안 시스템 기능, 통합 기능, 개발 지원 기능

A closer look

크로스 레이어 기능 (Cross Layer Functions)	•지원능력 제공을 위한 레이어 상의 컴포넌트들과 상호작용을 하는 일련의 기능적 컴포넌트들을 포함 •운영 지원 시스템 기능, 비즈니스 지원 시스템 기능, 보안 시스템 기능, 통합 기능, 개발 지원 기능

- 통합 기능(integration)
- 보안 시스템 기능(security)
- 운영 지원 시스템(operational support systems)
- 비즈니스 지원 시스템(business support systems)
- 개발 지원 기능(development function)

A more closer look

크로스 레이어 기능 (Cross Layer Functions)

- 지원능력 제공을 위한 레이어 상의 컴포넌트들과 상호작용을 하는 일련의 기능적 컴포넌트들을 포함
- 운영 지원 시스템 기능, 비즈니스 지원 시스템 기능, 보안 시스템 기능, 통합 기능, 개발 지원 기능

보안 통합(security integration)

- 보안 통합 컴포넌트 인증, 권한 부여, 암호화, 무결성 검증을 포함하는 보안 기능과 이와 관련된 정책 메커니즘에 대한 통합을 제공

모니터링 통합(monitoring integration)

- 접근 레이어, 서비스 레이어, 자원 및 네트워크 레이어의 컴포넌트로부터 운영 지원 시스템의 모니터링 및 보고 기능으로의 연결을 제공

서비스 통합(service integration)

- 제공자의 환경에서 실행되는 서비스에 대한 연결

대등 서비스 통합(peer service integraion)

- 적절한 보안 및 사용에 대한 과금 기능을 가지고 통제된 형태로 대등 제공자들의 서비스를 연결

Next function

크로스 레이어 기능 (Cross Layer Functions)

- 지원능력 제공을 위한 레이어 상의 컴포넌트들과 상호작용을 하는 일련의 기능적 컴포넌트들을 포함
- 운영 지원 시스템 기능, 비즈니스 지원 시스템 기능, 보안 시스템 기능, 통합 기능, 개발 지원 기능

인증 및 식별성 관리(authentication and identity management)

- 인증 및 식별성 관리 컴포넌트는 사용자 식별과 사용자 인증에 대한 신임정보와 관련된 기능을 제공

권한부여 및 서비스 정책 관리(authorization and service policy management)

- 사용자의 특정 기능 또는 데이터 접근에 대한 권한부여의 제어 및 응용 기능을 제공
- 서비스와 관련된 보안 정책의 정의 및 응용을 제공

암호화 관리(encryption management)

- 암호화 관리 컴포넌트는 데이터의 암호화와 관련된 기능을 제공
- 암호화 키 관리와 암호화 방법 선택

Next function

운영 지원 시스템 (Operational Support Systems)

- 프로비저닝(provisioning)
- 모니터링과 보고(monitoring and reporting)
- 서비스 정책 관리(service policy management)
- 서비스 자동화(service automation)
- 서비스 레벨 관리(SLA: service level management)
- 사고와 문제 관리(incident and problem management)
- 플랫폼과 가상화 관리 (platform and virtualization management)

Next function

비즈니스 지원 시스템 (Business Support Systems)

제품 카탈로그(product catalogue)

- 카탈로그는 클라우드 서비스 고객들이 구입 가능한 가용 서비스 제안 리스트를 탐색, 클라우드 서비스 제공자의 직원들이 카탈로그의 내용을 관리하는 기능을 제공

계정 관리(account management)

- 클라우드 서비스 고객 관계를 관리

신청 관리(subscription management)

- 클라우드 서비스 고객으로부터의 특징 클라우드 서비스에 대한 신청을 처리

과금(billing)

- 측정과 등급 기능에 의해 생성된 클라우드 서비스의 사용에 대한 요금을 기반으로 청구서를 생성, 고객에게 전송

계정(account)

- 수입 및 지출 계정을 포함하는 원장과 일반 회계 기능과 관련된 기능

Next function

개발 기능
(Development Function)

개발자 환경(developer environment)

- 서비스 구현 소프트웨어의 개발을 지원하기 위한 기능을 제공

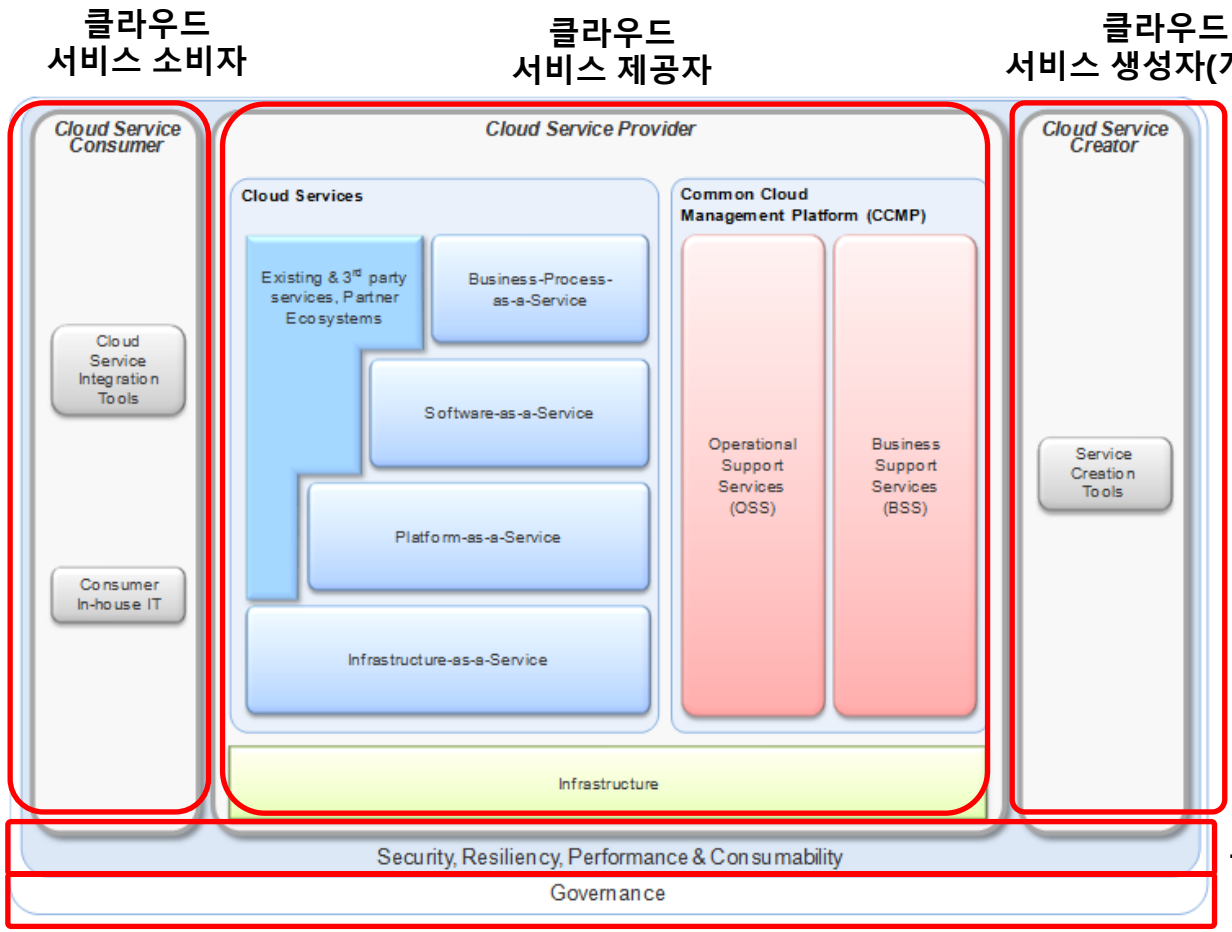
빌드 관리(Build Management)

- 클라우드 서비스 환경에서 전개하기 위해 클라우드 서비스 제공자에게 전달이 될 수 있는 전개-준비 소프트웨어 패키지 구축을 지원

테스트 관리(Test Management)

- 서비스 구현의 빌드에 대한 테스트 케이스의 실행을 지원

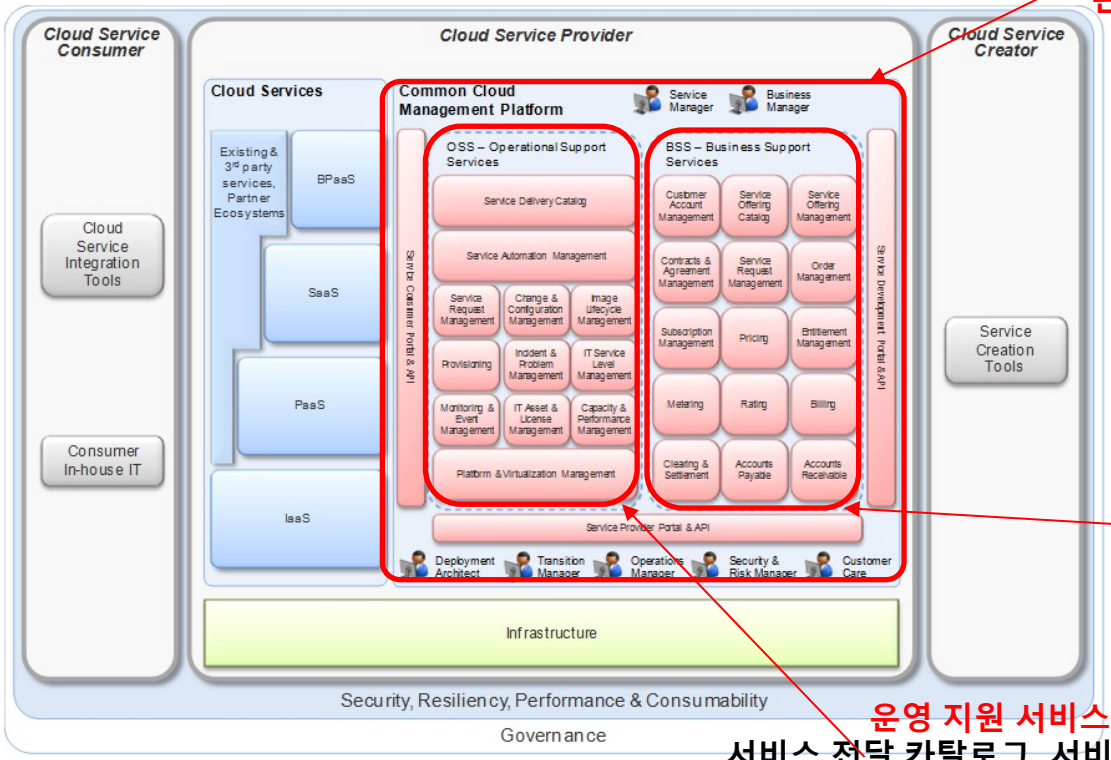
IBM cloud computing ref. arch.



IBM cloud computing ref. arch.

- IBM CCRA의 클라우드 관리 플랫폼의 세부 구조

공통 클라우드 관리 플랫폼



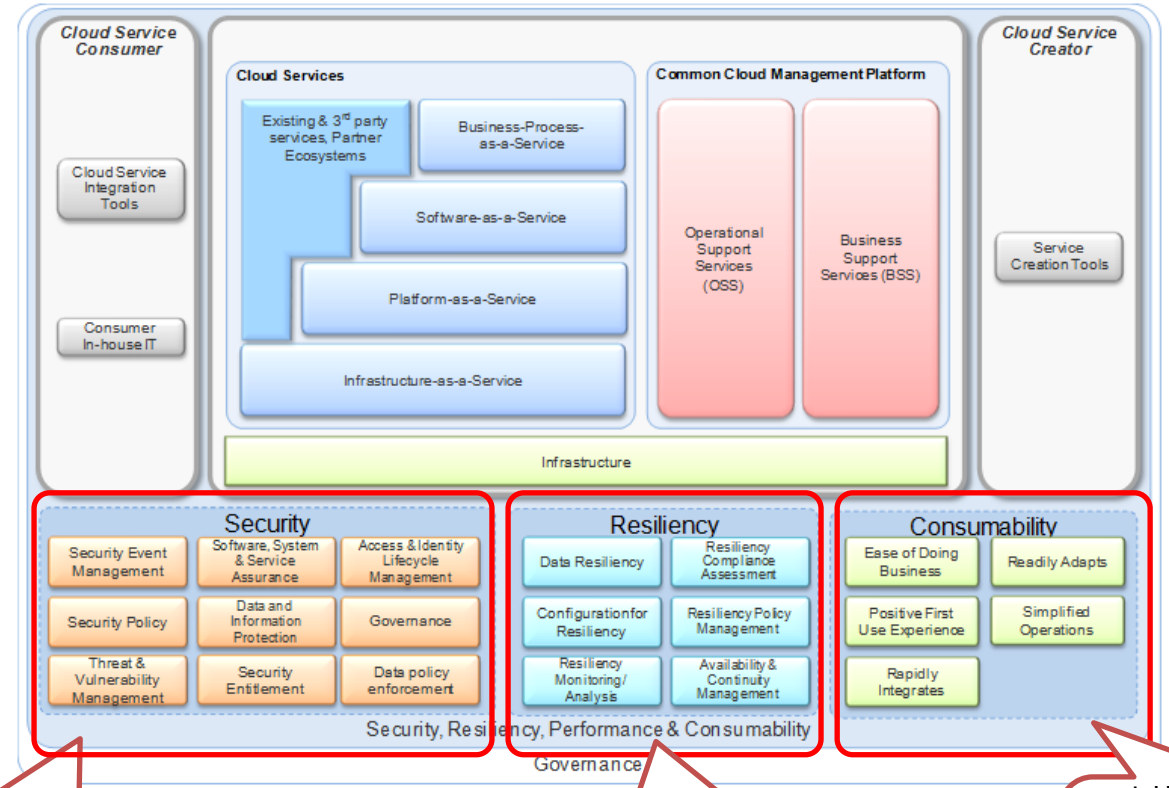
비즈니스지원 서비스
고객 계정 관리, 서비스 제공 카탈로그, 서비스 제공 관리, 계약과 동의 관리, 서비스 요청 관리 등

운영 지원 서비스(OSS)
서비스 전달 카탈로그, 서비스 자동관리, 서비스 요청관리, 변경 구성 관리, 이미지 라이클관리, 프로비저닝등

공통 클라우드 관리 플랫폼은 운영지원 서비스와 비즈니스지원 서비스로 구성

IBM cloud computing ref. arch.

- 보안, 탄력성, 성능, 소비가능성의 세부 구조



보안 : 소프트웨어, 시스템등외의 보안에 관련 구성

탄력성 : 데이터 회복성과 회복성에 관련된 정책, 모니터링, 분석 등등이 해당

소비가능성 : 비즈니스 수행의 용이성, 적응 용이성, 긍정적 최초 사용 경험, 단순화된 작동, 신속한 통합

Summary

- Cloud computing concept & architecture
  - Software architecture (from H/W to Service)
  - Server-side technology
  - Service oriented architecture
  - Consolidating multiple VM's workloads
  - Flexible utilization of computing resources
- Cloud Computing architecture
  - major functions: Services composition, inter-operation, resource mgmt, ser vice mgmt, monitoring, provisioning, security, etc.
  - layers, components, users

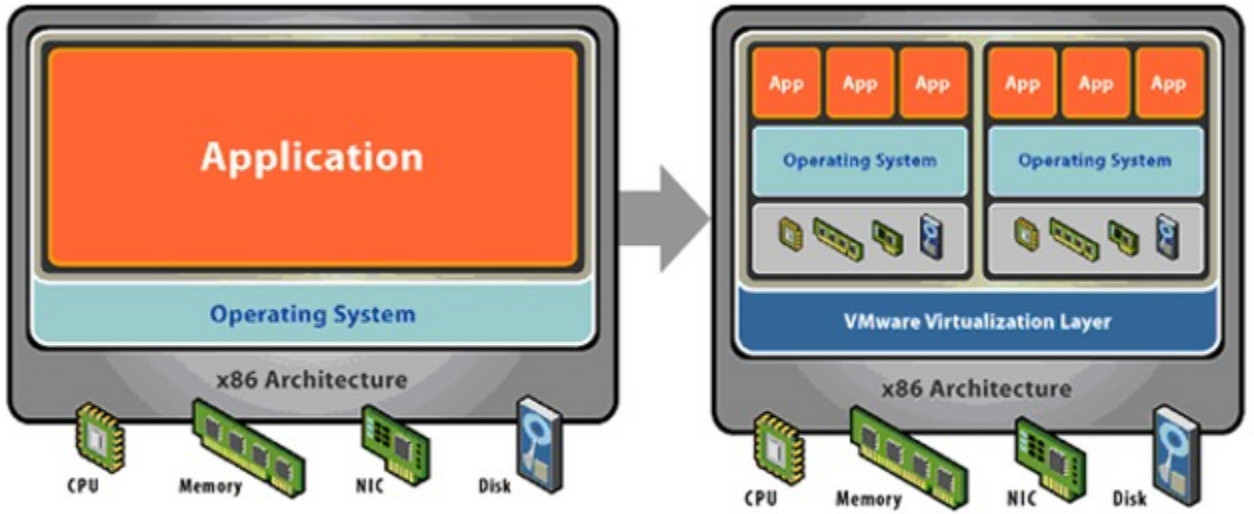
Cloud Technologies

Cloud Computing Characteristics

- on-demand self-service
  - increase service level whenever you need it, without human operation
- resource pooling
  - resource mgmt. for multiple users (multi-tenant), dynamic allocation accor ding to different users' demand
- measured service
  - measuring service usage
- broad network access
  - access to a service over diverse network access technologies
- rapid elasticity
  - computing power can be elastically provided

Some essential technology

- virtualization
  - illusion
  - multiple entries
  - logical separation
  - physical limitation

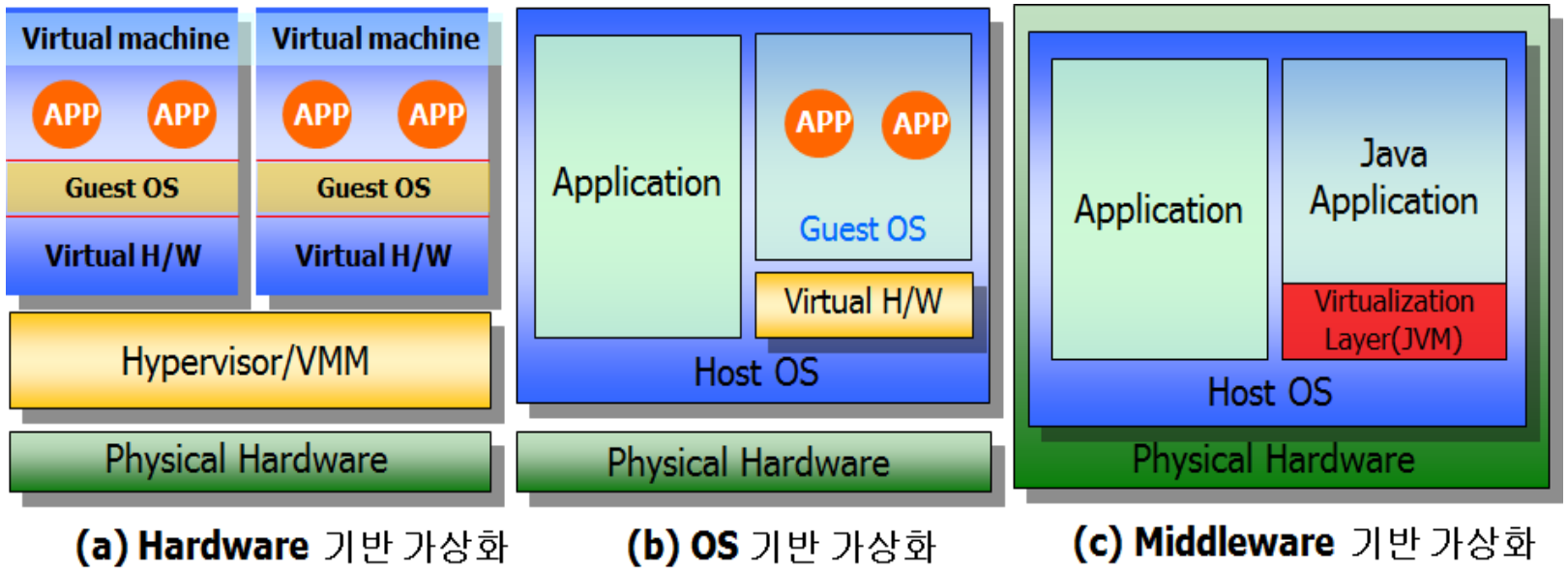


가상화 이전

가상화 이후

Virtual machines

- hypervisor
- virtual machine monitor
- guest OS
- host OS



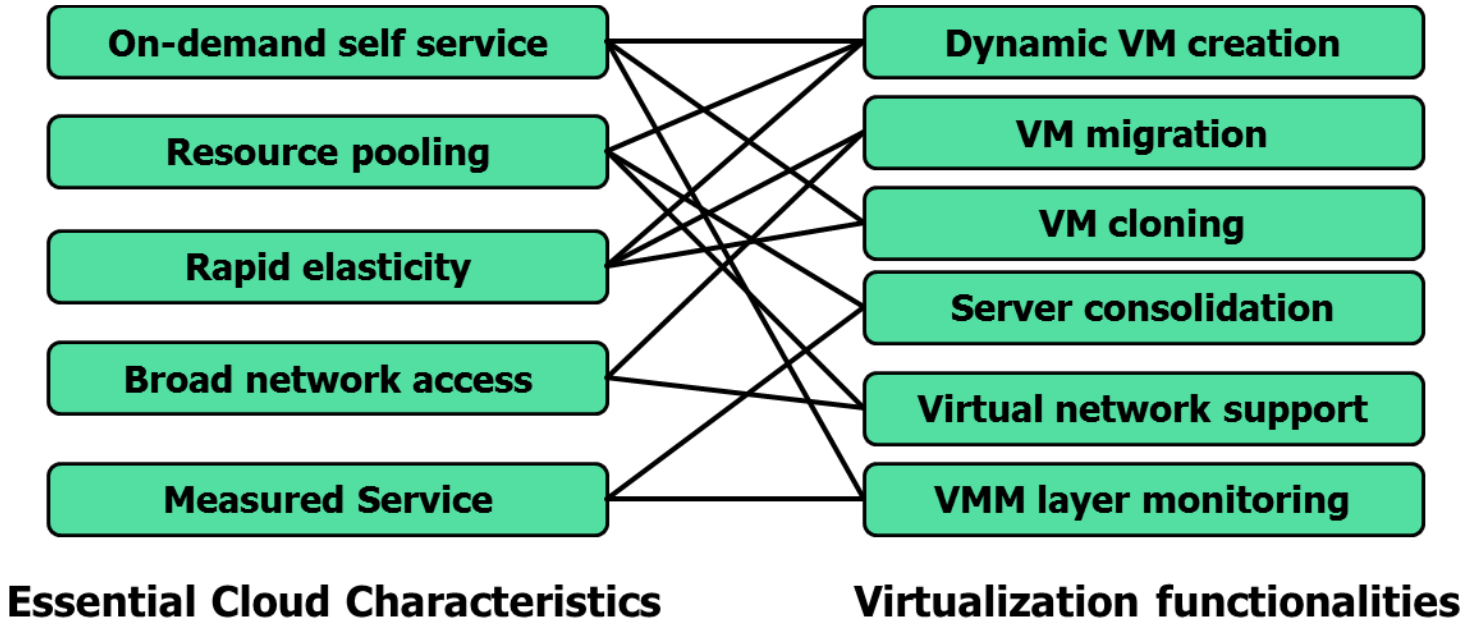
Key enabler of virtualization

- logical separation (isolation)
  - CPU
  - Memory
  - I/O devices (network, block)
- How?

Ultimate advantages of virtualization

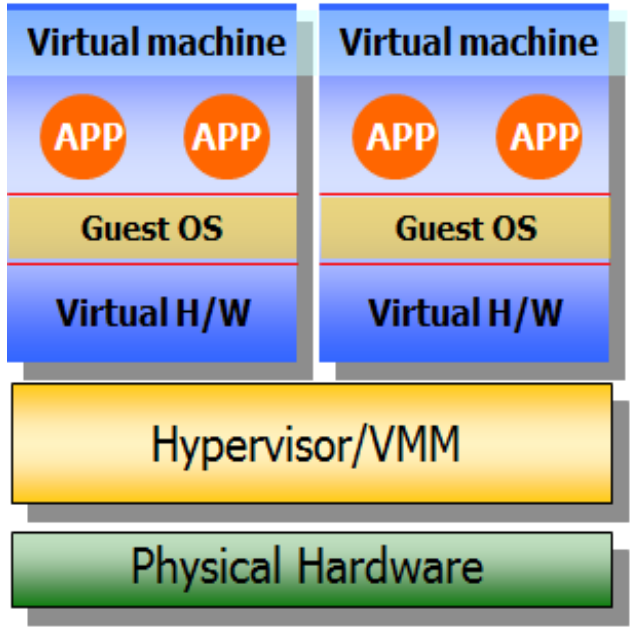
- Workload consolidation
  - webservers utilization??
  - consolidate 6~7 webservers into a single physical server
- workload isolation
  - multi-tenant support
  - performance isolation for a VM
- aggregated resource management
  - multiple physical resources --> logical unit
- Migration
  - migrate a VM over a network
- Emulation

Cloud computing and virtualization



Types of Virtualization

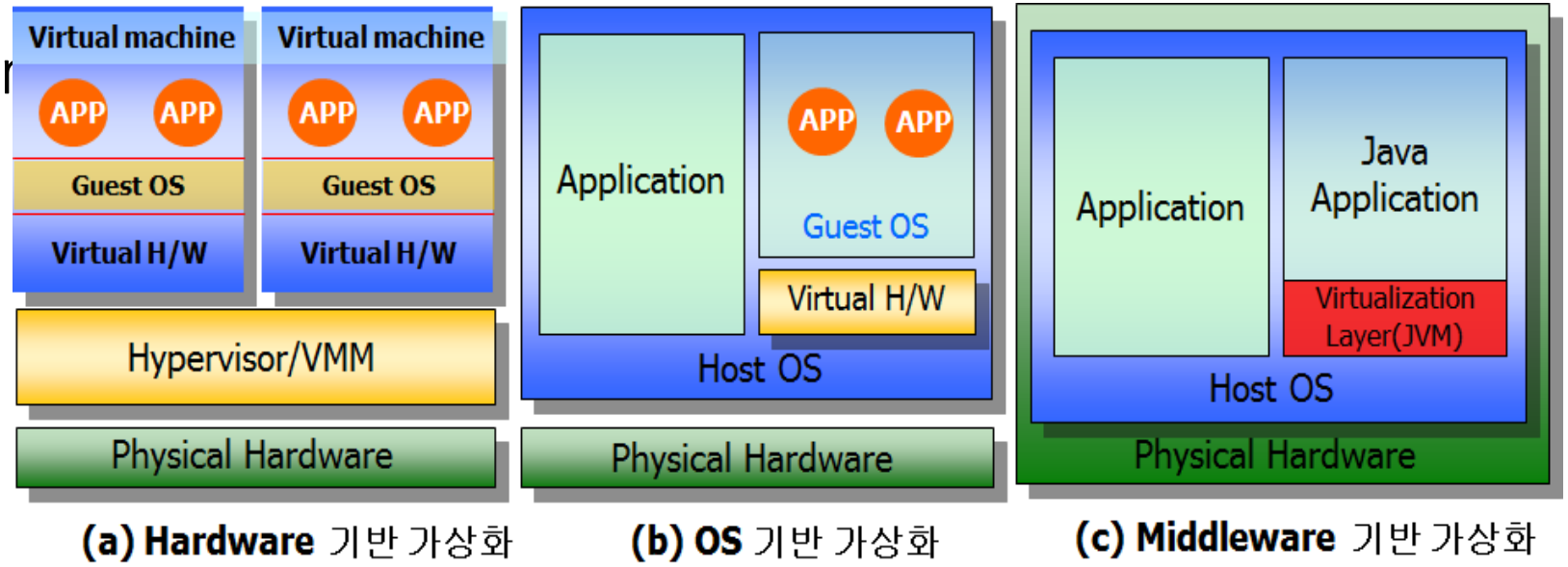
- HW support for Virtualization
- VT-X VT-d
  - CPU awares of the virtual machine / guest OS
  - VMCS (virtual machine control structure)
  - VMExit / VMEnter
  - hypervisor mode (ring 0)
- Nested Virtual memory
  - shadow page table
  - memory sharing
- SR-IOV
  - multiple queues for guest VM
  - Asynchronous I/O ring buffer



(a) Hardware 기반 가상화

Types of virtualization

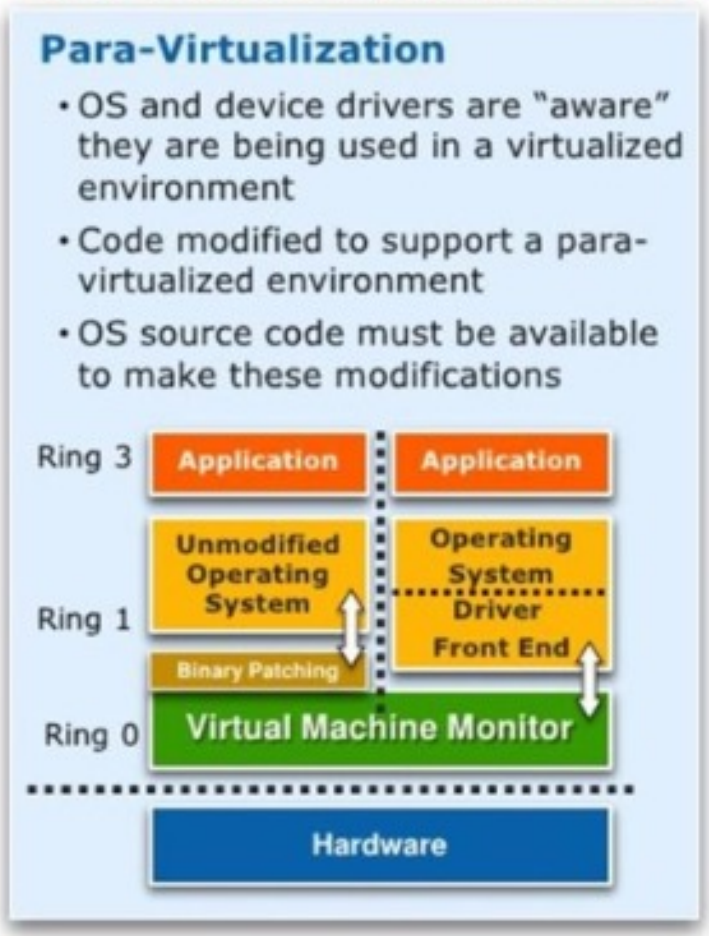
- Qemu / vmware / dosbox / boch / virtualbox
  - SW that emulates hardware operations
  - Android Studio → Android emulator
    - we can use ARM binary files (different ISA)
- How to emulate privileged operations?
  - Runs the code at user
  - OS Traps the instructio
  - deliver it to VMM

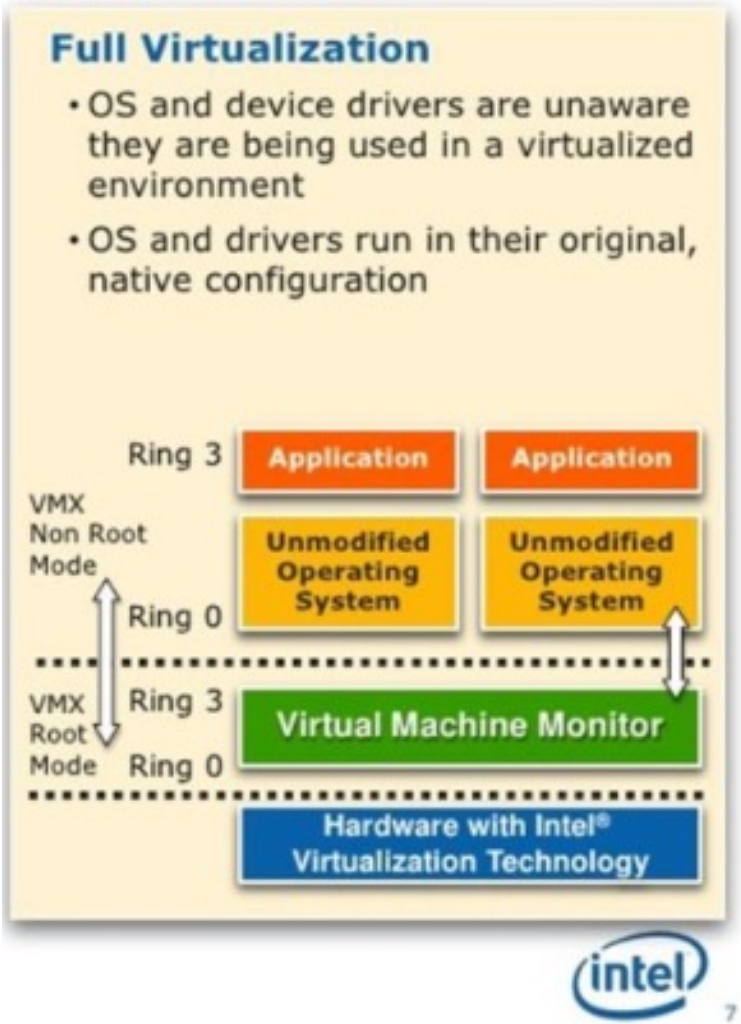


(a) **Hardware** 기반 가상화 (b) **OS** 기반 가상화 (c) **Middleware** 기반 가상화

Types of virtualization

- para-virtualization
  - memory update
  - frequent I/O jobs
- software optimization, rather than trap-and-emu
  - much better performance
  - Xen, used in AWS



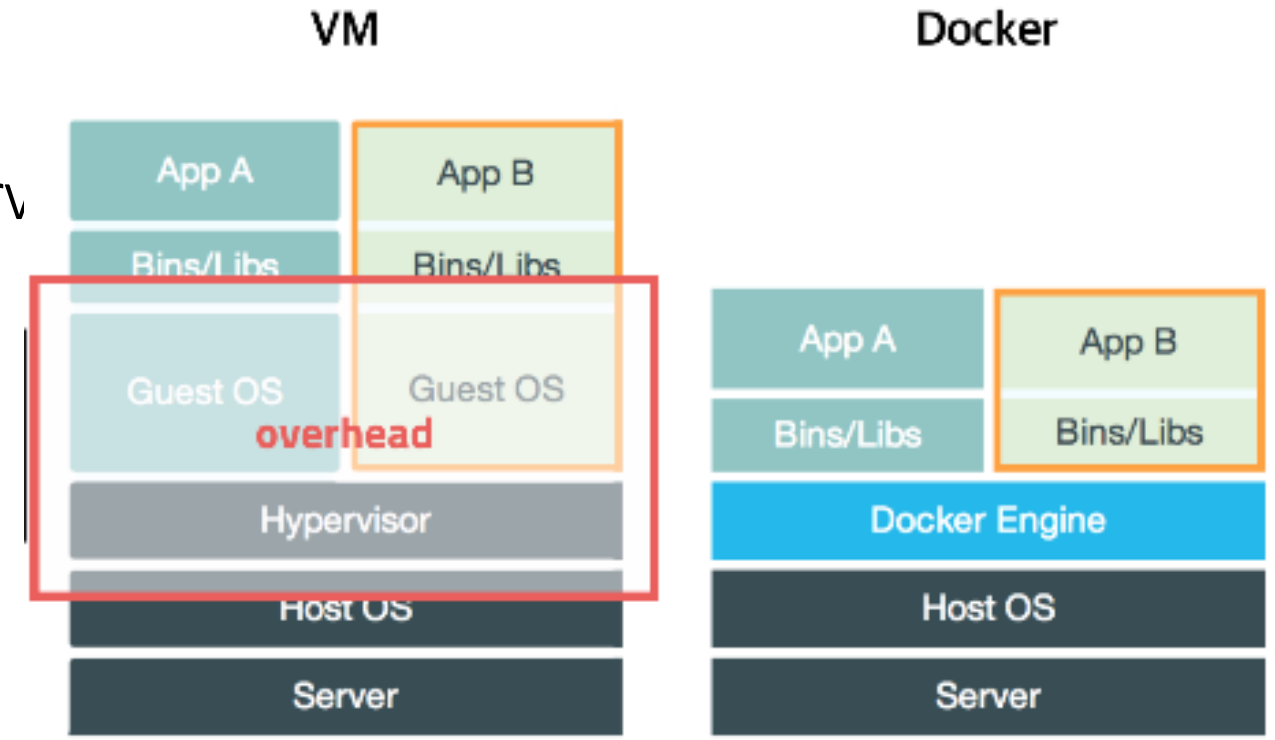


VM and migration

- Moving VM from a physical machine to another physical machine
- Why?
  - load balancing
  - availability
- stop-and-copy
- live VM migration
  - keep the VM running while we migrate the VM
  - pre-copy VM migration
  - post-copy VM migration
- SAN: shared storage / file system

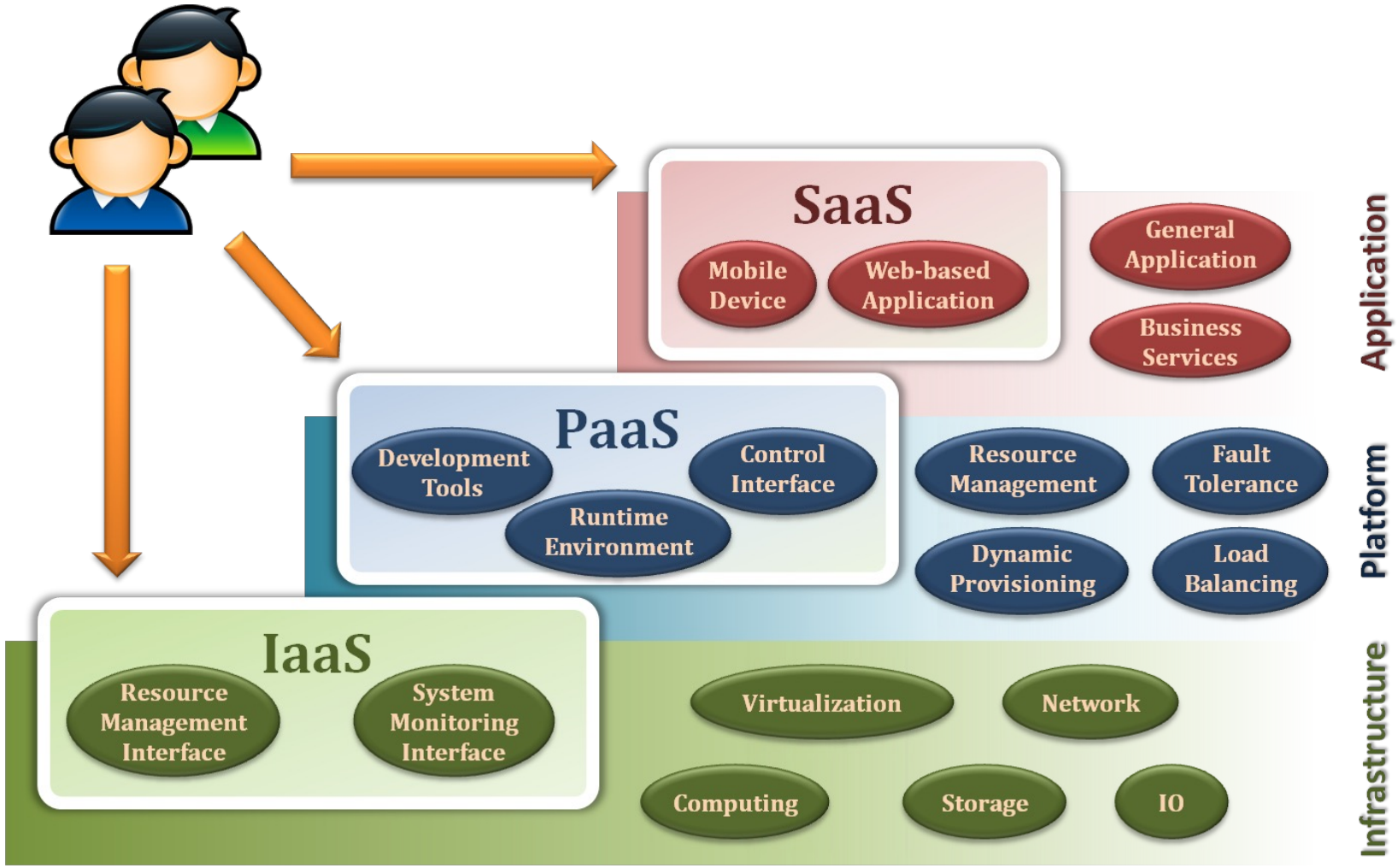
Containers

- Having the same kernel, but different name space & (QoS level)
  - cgroup, namespace in Linux kernel → Linux Container
- shared kernel, but isolated user space
- Make the VM small
  - VM with minimal libraries
    that only needed for the target serv
  - Micro-service architecture

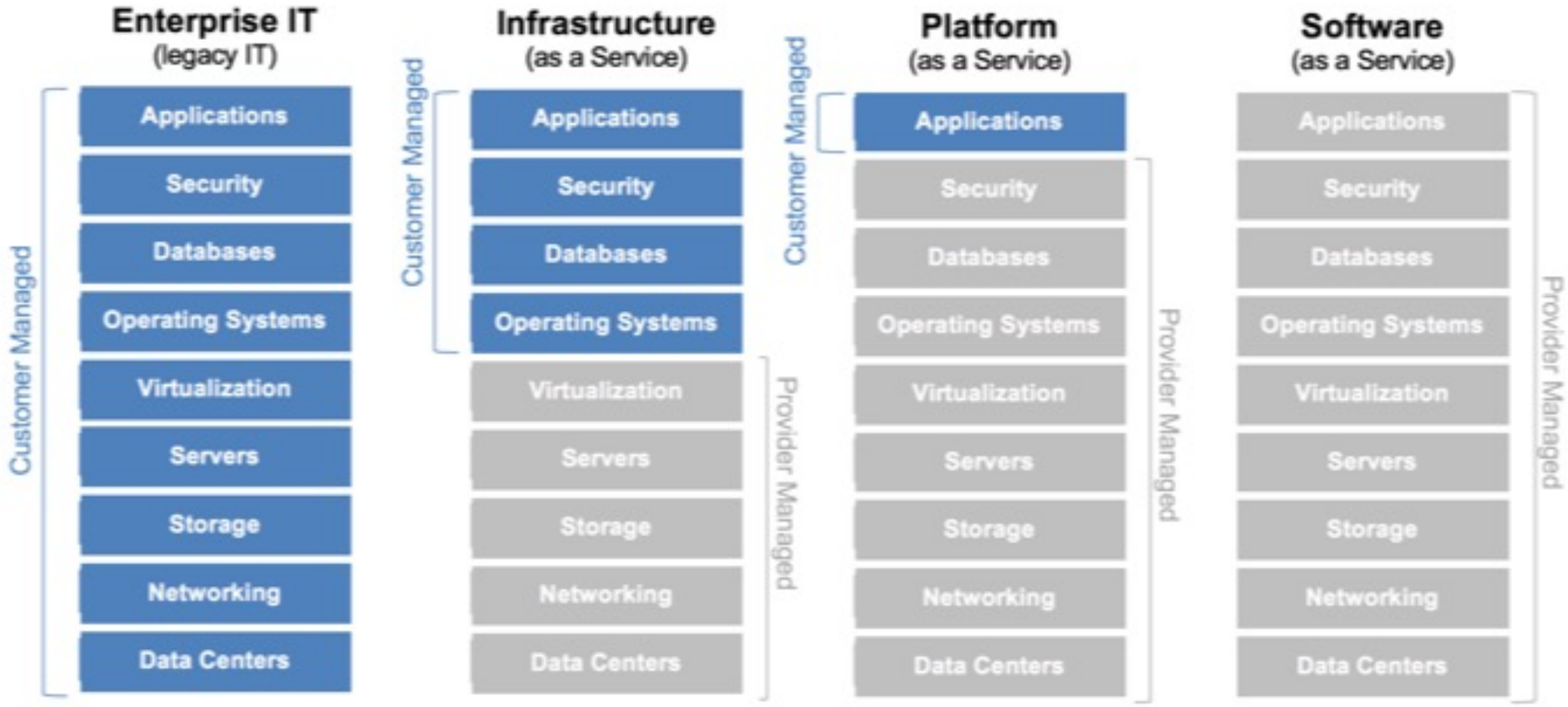


Cloud Service Models

Service model overview



SW stacks of service models



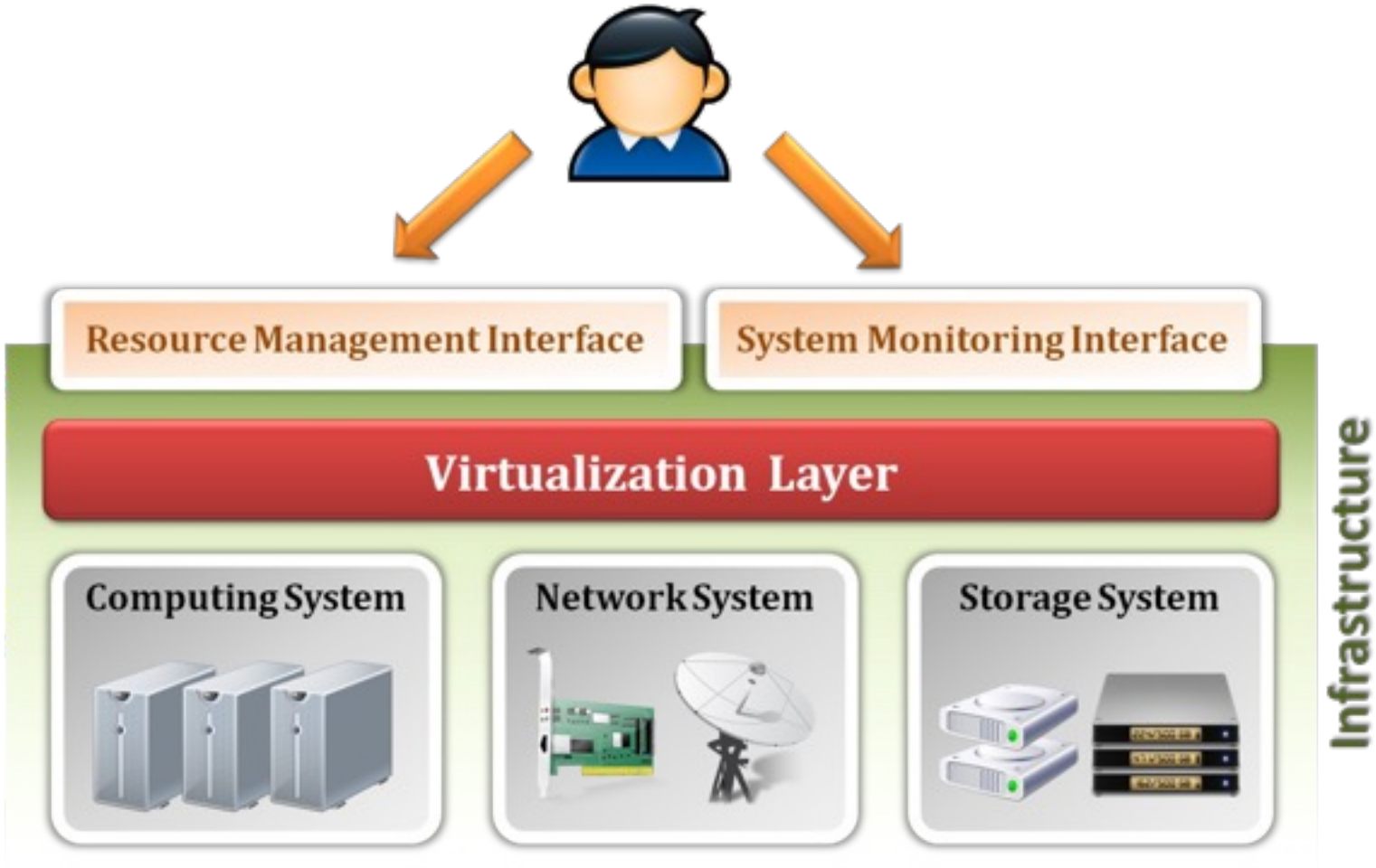
Infrastructure as a Service

- The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications.
- The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, deployed applications, and possibly limited control of select networking components .

- Examples :
  - Amazon EC2
  - Eucalyputs
  - OpenNebula
  - … etc

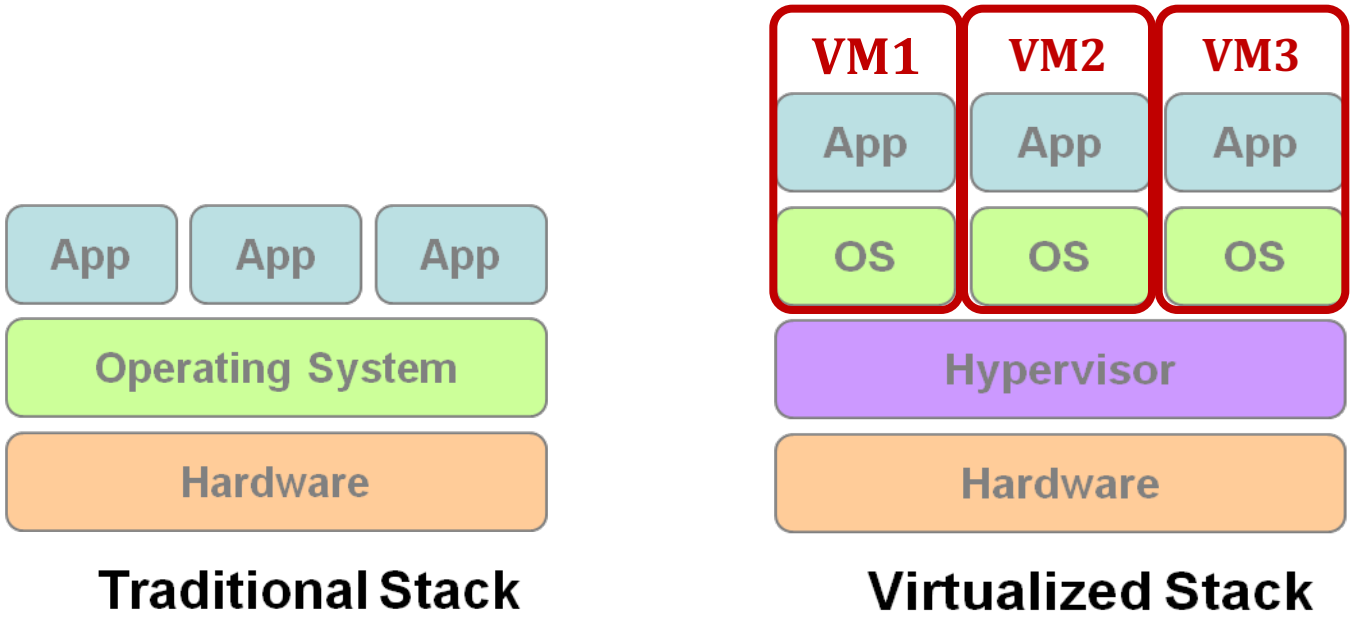
Infrastructure as a Service

- System architecture :



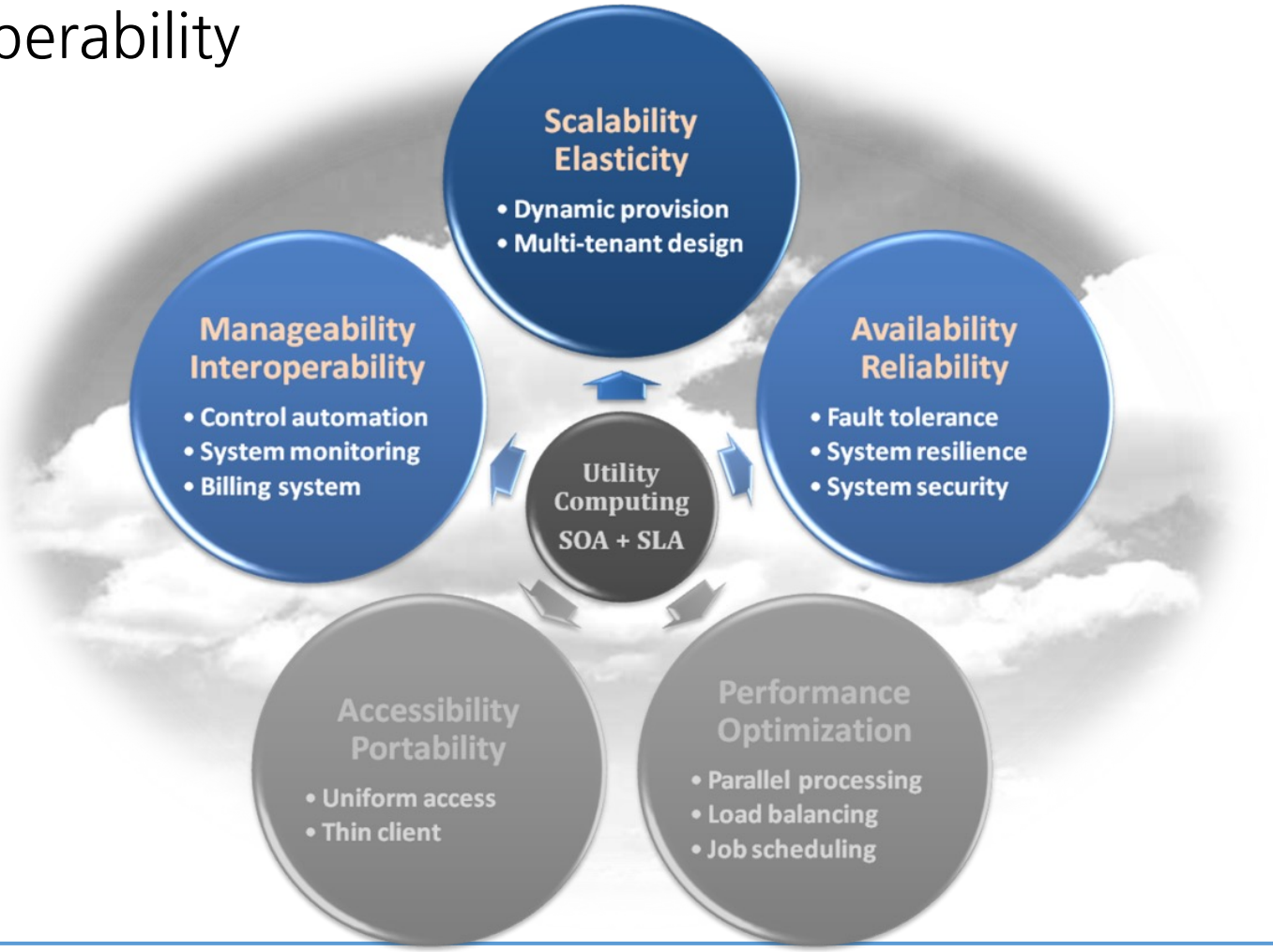
Infrastructure as a Service

- Enabling technique - *Virtualization*
  - Virtualization is an abstraction of logical resources away from underlying p hysical resources.
    - Virtualization technique shift OS onto hypervisor.
    - Multiple OS share the physical hardware and provide different services.
    - Improve utilization, availability, security and convenience.



Infrastructure as a Service

- Properties supported by virtualization technique :
  - Manageability and Interoperability
  - Availability and Reliability
  - Scalability and Elasticity



Infrastructure as a Service

- Provide service -**Resource Management Interface**
  - Several types of virtualized resource :
    - ***Virtual Machine*** - As an IaaS provider, we should be able to provide the basic virtual m achine operations, such as *creation*, *suspension*, *resumption* and *termination*, …etc.
    - ***Virtual Storage*** - As an IaaS provider, we should be able to provide the basic virtual sto rage operations, such as *space allocation*, *space release*, *data writing* and *data reading*, …etc.
    - ***Virtual Network*** - As an IaaS provider, we should be able to provide the basic virtual n etwork operations, such as *IP address allocation*, *domain name register*, *connection es tablishment* and *bandwidth provision*, …etc.

Infrastructure as a Service

- Provide service - **System Monitoring Interface**
  - Several types of monitoring metrics :
    - ***Virtual Machine*** - As an IaaS provider, we should be able to monitor some system states of each virtual machine, such as *CPU loading*, *memory utilization*, *IO loading* and *internal network loading*, …etc.
    - ***Virtual Storage*** - As an IaaS provider, we should be able to monitor some storage states of each virtual storage, such as *virtual space utilization*, *data duplication* and *storage device access bandwidth*, …etc.
    - ***Virtual Network*** - As an IaaS provider, we should be able to monitor some network states of each virtual network, such as *virtual network bandwidth*, *network connectivity* and *network load balancing*, …etc.

IaaS - Summary

- IaaS is the deployment platform that abstract the infrastructure.

- IaaS enabling technique
  - Virtualization
    - Server Virtualization
    - Storage Virtualization
    - Network Virtualization

- IaaS provided services
  - Resource Management Interface
  - System Monitoring Interface

Platform as a Service

- The capability provided to the consumer is to deploy onto the cloud infrastr ucture consumer-created or acquired applications created using programm ing languages and tools supported by the provider.
- The consumer does not manage or control the underlying cloud infrastruct ure including network, servers, operating systems, or storage, but has contr ol over the deployed applications and possibly application hosting environ ment configurations.

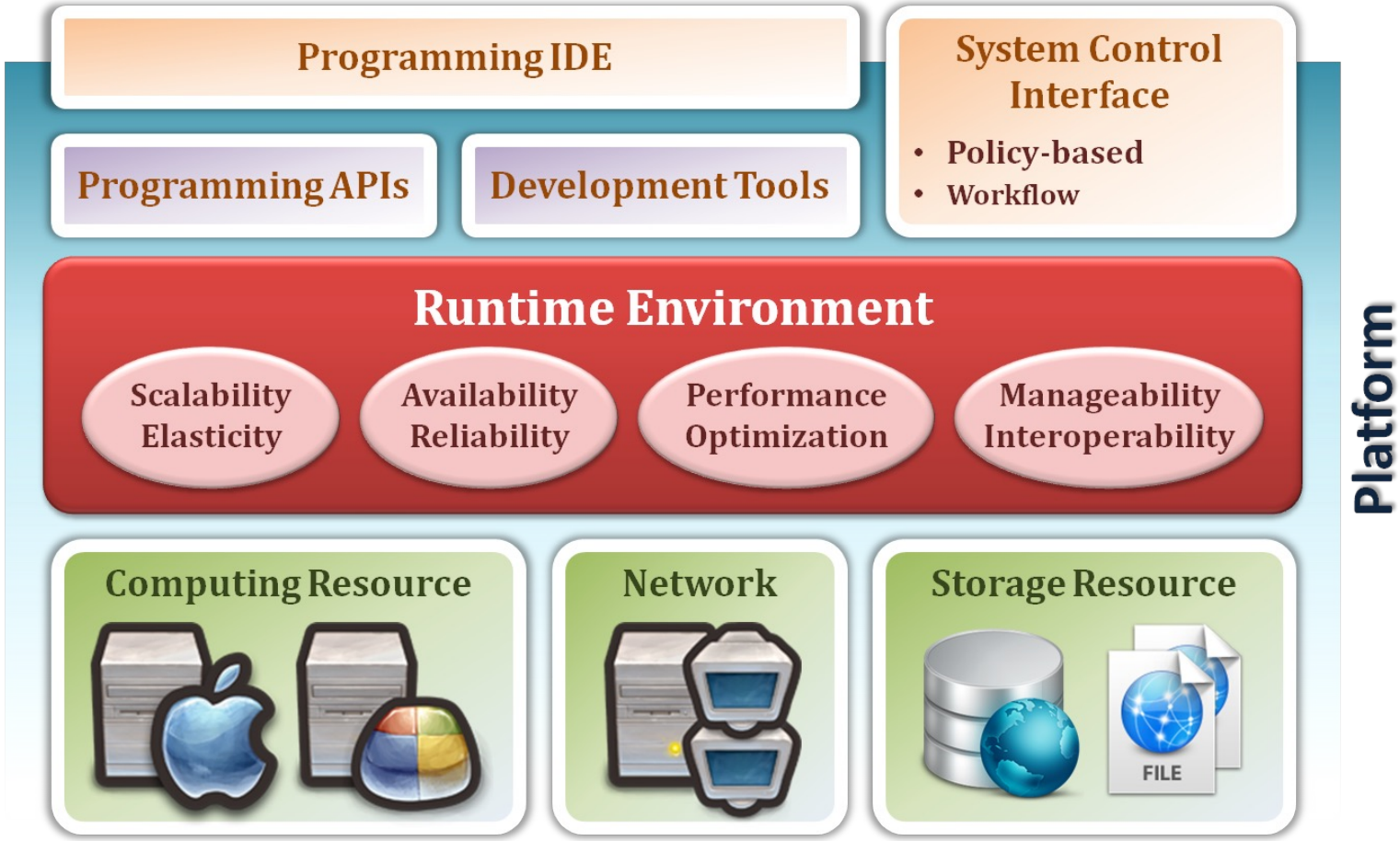
- Examples :
  - Microsoft Windows Azure
  - Google App Engine
  - Hadoop
  - … etc

PaaS Services

서비스 구분	서비스 내용	활용사례
확장 플랫폼 (Software Platform)	· 시스템 소프트웨어를 완성한 형태 (pre-built)로 필요한 기관에 제공 · 재정 여력이 부족한 중소기업의 경우, 시스템 개발에 필요한 표준 환경을 저렴한 비용으로 단기간 내에 제공 받는 것이 가능	· Amazon : EC2 · WuXi의 클라우드 · IBM : TAP
구축 플랫폼 (Development Platform)	· 개발자가 손쉽게 프로그램 개발 및 테스트할 수 있는 개발 프레임워크를 제공 · 어플리케이션 소프트웨어의 개발을 위한 실행환경 (Java, NET 등) 및 프레임워크를 함께 제공	· Google App Engine · Amazon : EC2 · Hadoop
운영 플랫폼 (Delivery Platform)	· IaaS 서비스 제공을 위한 운영환경 제공의 기반 (운영플랫폼이 없다면 서버 위에 운영체제, 실행환경, 관리환경, 네트워크 구성 등의 작업을 수작업으로 진행하여나 하나 이를 바로사용할 수 있는 형태로 제공	· Google App Engine · 세일즈포스 : CRM S/W 변경 및 확장 API 제공

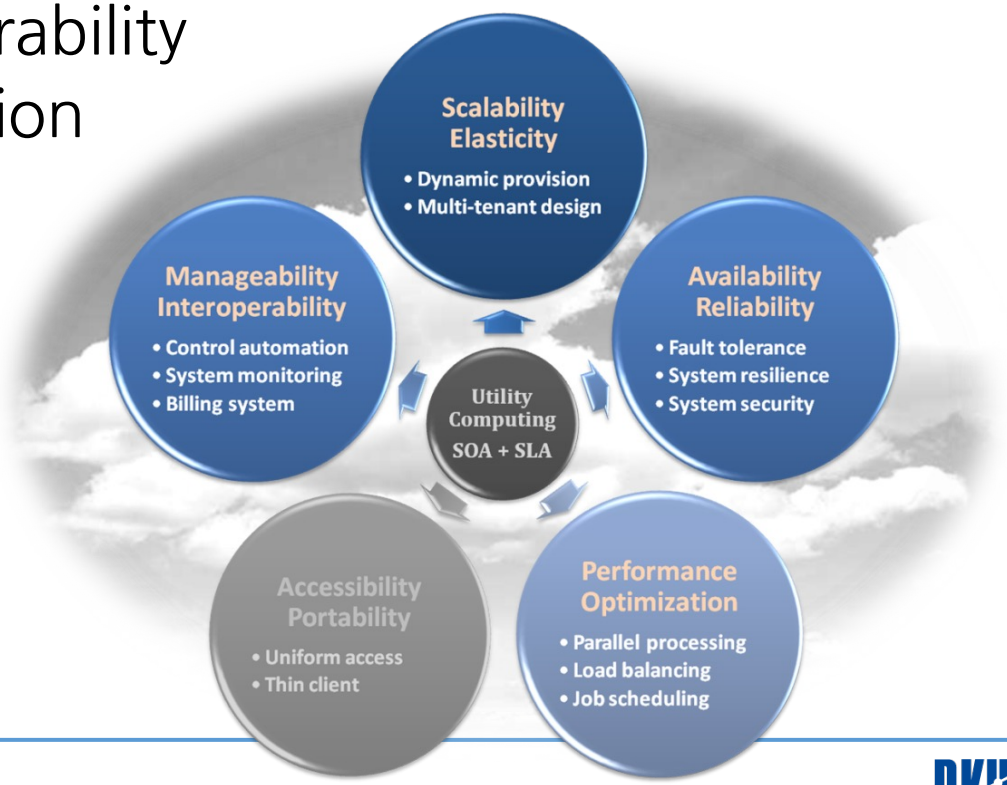
Platform as a Service

- System architecture :



Platform as a Service

- Enabling technique - **Runtime Environment Design**
  - Runtime environment refers to collection of software services available. Us ually implemented by a collection of program libraries.
- Common properties in Runtime Environment :
  - Manageability and Interoperability
  - Performance and Optimization
  - Availability and Reliability
  - Scalability and Elasticity



Platform as a Service

- Provide service - **Programming IDE**
  - Users make use of programming IDE to develop their service among PaaS.
    - This IDE should integrate the full functionalities which supported from the underling r untime environment.
    - This IDE should also provide some development tools, such as profiler, debugger and t esting environment.
  - The programming APIs supported from runtime environment may be vario us between different cloud providers, but there are still some common ope rating functions.
    - Computation, storage and communication resource operation

Platform as a Service

- Provide service - **System Control Interface**
  - Policy-Based Control
    - Typically described as a principle or rule to guide decisions and achieve rational outco me(s)
    - Make the decision according to some requirements
  - Workflow Control
    - Describe the flow of installation and configuration of resources
    - Workflow processing daemon delivers speedy and efficient construction and manage ment of cloud resources

PaaS - Summary

- PaaS is the development platform that abstract the infrastructure, OS, and middleware to drive developer productivity.

- PaaS enabling technique
  - Runtime Environment

- PaaS provide services
  - Programming IDE
    - Programming APIs
    - Development tools
  - System Control Interface
    - Policy based approach
    - Workflow based approach

Software as a Service

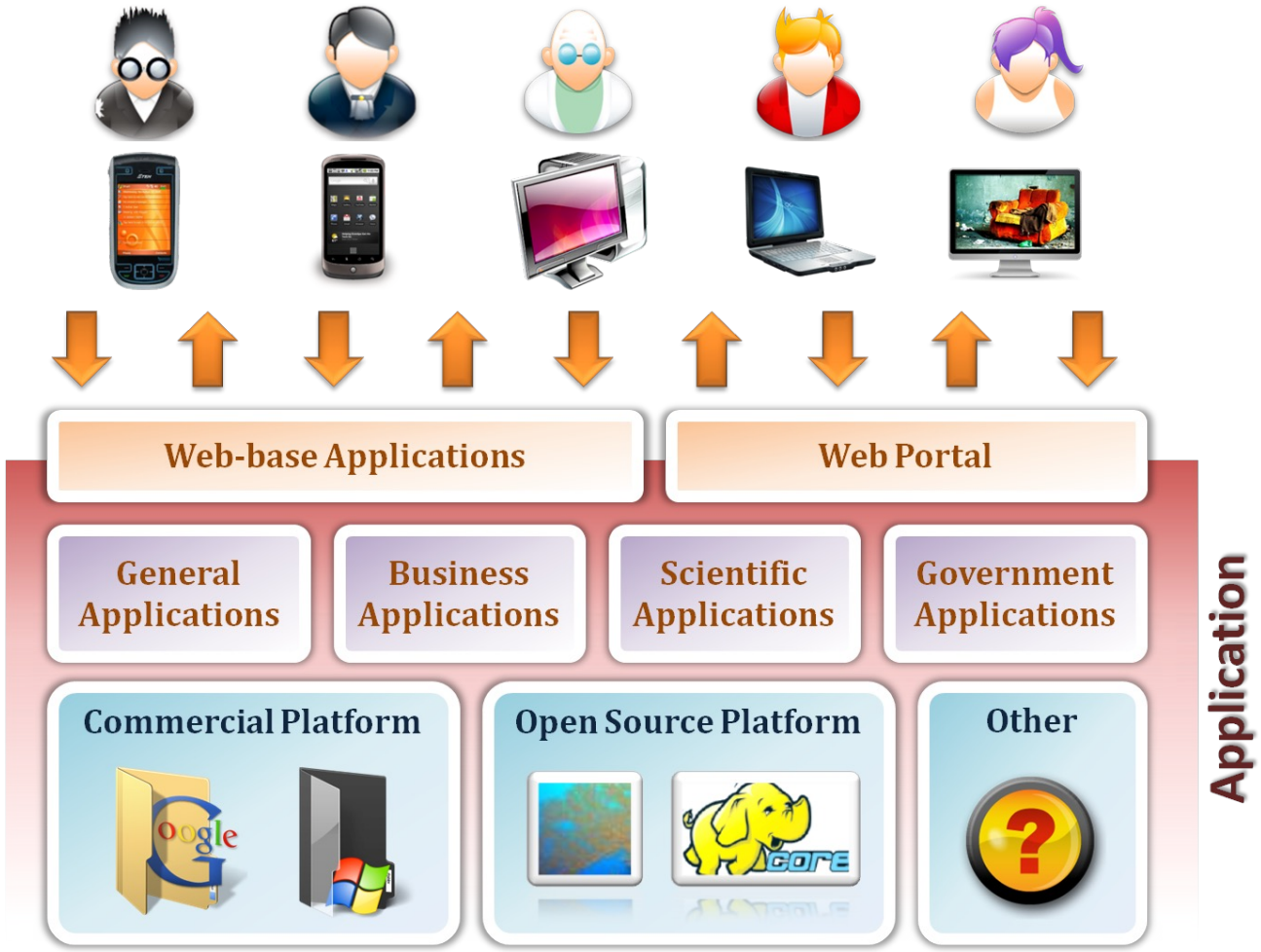
- The capability provided to the consumer is to use the provider's application s running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email).
- The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

- Examples :
  - Google Apps (e.g., Gmail, Google Docs, Google sites, …etc)
  - SalesForce.com
  - EyeOS
  - … etc

SaaS Services

구 분	내 용	예 시	사업자
단순 사무 자동화기능	데이터 계산, 워드 프로세싱 등 단순사무를 위한 소프트웨어	· 사무자동화 · 자료 관리	· 구글(구글 독스) · 싱크프리 웹 오피스
기업 단일기능	회계, 급여, 재고 관리와 같은 단일 기능을 처리하기 위한 소프트웨어	· 회계 패키지 · 고객관리 · 재고관리 · 생산관리 · 영업관리	· 세일즈포스닷컴 영업 자동화 · 오라클 시벨 고객관계 관리 · 영업 자동화 고객관계 관리 세일즈
기업내 통 합	회계, 급여, 고객 관리 등의 기능을 연계 처리할 수 있는 통합 솔루션	· 그룹웨어 · 전사적 자원 관리	· 마이크로소프트 라이브 미팅 · 넷스위트 전사적 자원 관리
기업간 통 합	공급사슬 관리, 연구개발 등 기업 간 협업 및 공동 거래를 처리할 수 있는 솔루션	· 공급사슬관리 · 자동 주문 및 납품	

Software as a Service



Software as a Service

- Enabling Technique - **Web Service**
  - Web 2.0 is the trend of using the full potential of the web
    - Viewing the Internet as a computing platform
    - Running interactive applications through a web browser
    - Leveraging interconnectivity and mobility of devices
    - Enhanced effectiveness with greater human participation
- Properties provided by Internet :
  - Accessibility and Portability

Software as a Service

- Provide service - **Web-based Applications**
  - Conventional applications should translate their access interface onto web-based platform.
  - Applications in different domains
    - ***General Applications*** - Applications which are designed for general propose, such as office suit, *multimedia* and *instant message*, …etc.
    - ***Business Applications*** - Application which are designed for business propose, such as *ERP*, *CRM* and *market trading system*, …etc.
    - ***Scientific Applications*** - Application which are designed for scientific propose, such as *aerospace simulation* and *biochemistry simulation*, …etc.
    - ***Government Applications*** - Applications which are designed for government propose, such as *national medical system* and *public transportation system service*, …etc.

Software as a Service

- Provide service - **Web Portal**
  - Apart from the standard search engine feature, web portals offer other ser vices such as e-mail, news, stock prices, information, databases and enterta inment.
  - Portals provide a way for enterprises to provide a consistent look and feel with access control and procedures for multiple applications and databases, which otherwise would have been different entities altogether.
  - Some examples :
    - iGoogle
    - MSNBC
    - Netvibes
    - Yahoo!

SaaS - Summary

- SaaS is the finished applications that you rent and customize.

- SaaS enabling technique
  - Web Service

- SaaS provide services
  - Web-based Applications
    - General applications
    - Business applications
    - Scientific applications
    - Government applications
  - Web Portal

System Administration

server chores

- make users / separate security levels
- review logs → logging
- tedious and repetitive job handling → automatize jobs
- configure settings → keep consistency
- install pkgs
- monitor utilizations → trouble shooting
- backup & periodic check-ups → dealing with failures and serious tr oubles

making a separate user

- usually for limited access privileges
  - not to ruin others work
- visudo
  - add sudoer
  - know what you're doing

Checking the log

- string parsing (using python, perl, js, etc.)
  - periodic check ups for security attacks
  - auditing files back up
  - production system log
  - development system log will give you more information
  - logging takes serious overhead
- periodic check for
  - system log
  - auth log
  - firewall log

automatize tedious jobs

- learn from your repetition
  - if you do some jobs multiple times, consider shell programming
- learn some shell script
  - that does much things on behalf of your hand-dirty works
- Bash, csh, tcsh, etc.
  - crond helps you conduct periodic operations

configuration check up

- Library dependency is a HUGE problem!
  - someone needs 1.0.1, another needs 1.2.3
  - they run different applications on different platforms - its okay
  - they run different applications on the same platforms - persuasive
  - they develop different applications targeting on the same platform - persua sive
  - they develop the same application targeting on the same platform - no wa y but conflict
- git code shape management
  - somebody has to merge the code base
  - merge window mgmt.
  - use version management software
  - make some documents

install pkgs

- rpm (redhat package manager)
- deb (debian package file)
- apk (android package file)
- OS has a package repositories
  - RHEL - redhat enterprise Linux: subscription-based pkg mgmt
  - Ubuntu - free repository, and you can add personal or 3rd-party repository

monitoring tools

- kernel
  - perf, ftrace, dmesg, proc. fs
- cpu usage
  - top, ps
- memory usage
  - vmstat, sar, free
- disk usage
  - df, du
- I/O devices, files
  - iotop, lspci, sysfs, lsof, iostat
- network
  - tcpdump, netstat,
- System services
  - systemctl, service

crontab for periodic jobs

- min / hour / day / month / weekday / running_job
  - example) Run a batch job (run_batch.sh) in the 8pm. everyday
  - * 20 * * * home/script/run_batch.sh
- man crontab

tar for packing files

- preparing for more serious failures such as sudo rm -rf /.
- To create)
  - tar cvf backup.tar /backup_directory
- periodic backup and send it to a separate storage
- you may want to compress with zip or bzip
- man tar

man pages and --help

- Please read the man page
  - manual of the system
- Please check the help before getting into Internet
  - --help gives much information, in many cases

WordPress with MySQL

WordPress: a personal webserver s/w

- helps you to host a personal web server
  - If you have a server, try that
- Install apache2 server
- install php engine with it
- install mysql, configure mysql
- download wordpress from Internet
- configure apache2 server for wordpress
  - wordpress location, secure access
- configure mysql for wordpress
  - create db user, grant access to DB for the user
- configure wordpress
  - wordpress configuration file
- restart db server and apache2 server
- check it from web browser

In the lab., we will do

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/hosting-wordpress.html

Mini-lab.