CS162
Operating Systems and Systems Programming
Lecture 22

Remote Procedure Calls (RPC), Network Layering, TCP/IP

April 23rd, 2019

Prof. John Kubiatowicz

http://cs162.eecs.Berkeley.edu

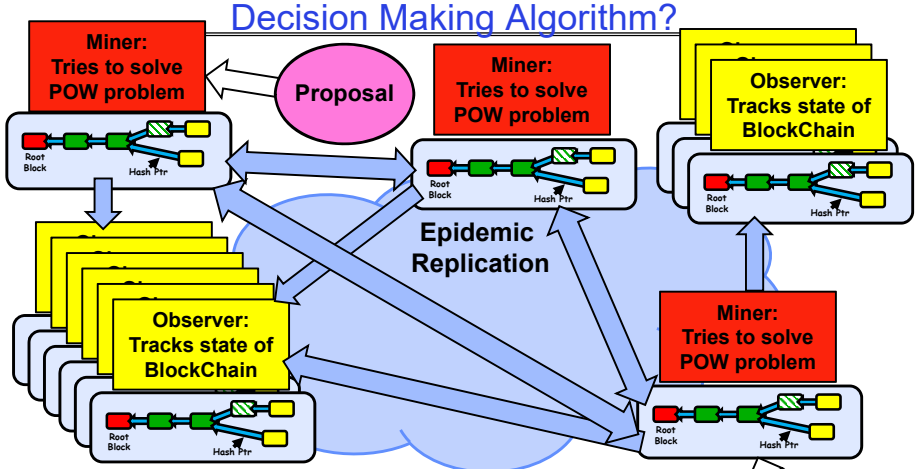
Recall: Distributed Consensus

- Consensus problem
  - All nodes propose a value
  - Some nodes might crash and stop responding
  - Eventually, all remaining nodes decide on the same value from set of proposed values
- Distributed Decision Making
  - Choose between "true" and "false"
  - Or Choose between "commit" and "abort"
- Equally important (but often forgotten!): make it durable!
  - How do we make sure that decisions cannot be forgotten?
    - » This is the "D" of "ACID" in a regular database
  - In a global-scale system?
    - » What about erasure coding or massive replication?
    - » Like BlockChain applications?

Recall: Two-Phase Commit

- Since we can't solve the General's Paradox (i.e. simultaneous action), let's solve a related problem
  - Distributed transaction: Two machines agree to do something, or not do it, atomically
- Two-Phase Commit protocol:
  - Persistent stable log on each machine: keep track of whether commit has happened
    - » If a machine crashes, when it wakes up it first checks its log to recover state of world at time of crash
  - Prepare Phase:
    - » The global coordinator requests that all participants will promise to commit or rollback the transaction
    - » Participants record promise in log, then acknowledge
    - » If anyone votes to abort, coordinator writes "Abort" in its log and tells everyone to abort; each records "Abort" in log
  - Commit Phase:
    - » After all participants respond that they are prepared, then the coordinator writes "Commit" to its log
    - » Then asks all nodes to commit; they respond with ack
    - » After receive acks, coordinator writes "Got Commit" to log
  - Log can be used to complete this process such that all machines either commit or don't commit

Recall: Is a Blockchain a Distributed Decision Making Algorithm?



- Yes! Decision means: Proposal is locked into BlockChain
  - Could be Commit/Abort decision
  - Could be Choice of Value, State Transition, ….
- NAK: Didn't make it into the block chain (must retry!)
- Anyone in world can verify the result of decision making!

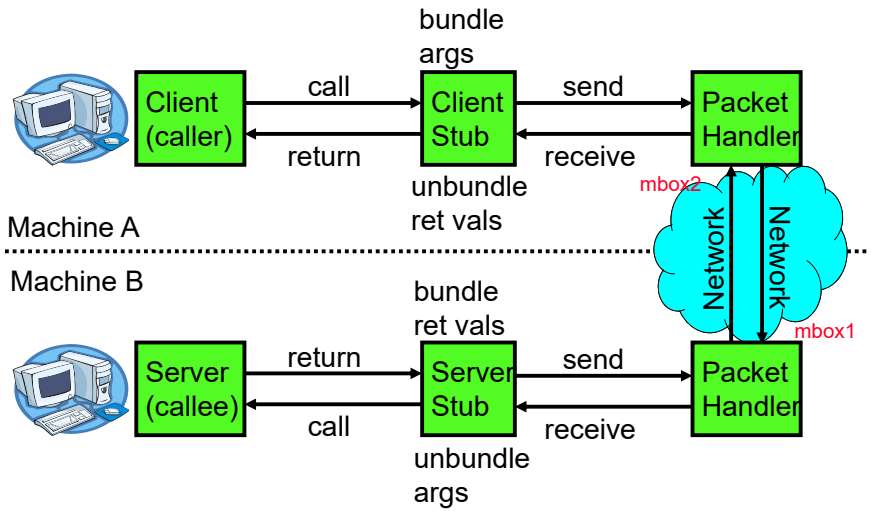
Remote Procedure Call (RPC)

- Raw messaging is a bit too low-level for programming
  - Must wrap up information into message at source
  - Must decide what to do with message at destination
  - May need to sit and wait for multiple messages to arrive
- Another option: Remote Procedure Call (RPC)
  - Calls a procedure on a remote machine
  - Client calls:
    ```
    remoteFileSystem→Read("rutabaga");
    ```
  - Translated automatically into call on server:
    ```
    fileSys→Read("rutabaga");
    ```


RPC Implementation

- Request-response message passing (under covers!)
- "Stub" provides glue on client/server
  - Client stub is responsible for "marshalling" arguments and "unmarshalling" the return values
  - Server-side stub is responsible for "unmarshalling" arguments and "marshalling" the return values.
- Marshalling involves (depending on system)
  - Converting values to a canonical form, serializing objects, copying arguments passed by reference, etc.

RPC Information Flow



RPC Details (1/3)

- Equivalence with regular procedure call
  - Parameters ⇔ Request Message
  - Result ⇔ Reply message
  - Name of Procedure: Passed in request message
  - Return Address: mbox2 (client return mail box)
- Stub generator: Compiler that generates stubs
  - Input: interface definitions in an "interface definition language (IDL)"
    - » Contains, among other things, types of arguments/return
  - Output: stub code in the appropriate source language
    - » Code for client to pack message, send it off, wait for result, unpack result and return to caller
    - » Code for server to unpack message, call procedure, pack results, send them off

RPC Details (2/3)

- Cross-platform issues:
  - What if client/server machines are different architectures/ languages?
    - » Convert everything to/from some canonical form
    - » Tag every item with an indication of how it is encoded (avoids unnecessary conversions)
- How does client know which mbox to send to?
  - Need to translate name of remote service into network endpoint (Remote machine, port, possibly other info)
  - Binding: the process of converting a user-visible name into a network endpoint
    - » This is another word for "naming" at network level
    - » Static: fixed at compile time
    - » Dynamic: performed at runtime

RPC Details (3/3)

- Dynamic Binding
  - Most RPC systems use dynamic binding via name service
    - » Name service provides dynamic translation of service → mbox
  - Why dynamic binding?
    - » Access control: check who is permitted to access service
    - » Fail-over: If server fails, use a different one
- What if there are multiple servers?
  - Could give flexibility at binding time
    - » Choose unloaded server for each new client
  - Could provide same mbox (router level redirect)
    - » Choose unloaded server for each new request
    - » Only works if no state carried from one call to next
- What if multiple clients?
  - Pass pointer to client-specific return mbox in request

Problems with RPC: Non-Atomic Failures

- Different failure modes in dist. system than on a single machine
- Consider many different types of failures
  - User-level bug causes address space to crash
  - Machine failure, kernel bug causes all processes on same machine to fail
  - Some machine is compromised by malicious party
- Before RPC: whole system would crash/die
- After RPC: One machine crashes/compromised while others keep working
- Can easily result in inconsistent view of the world
  - Did my cached data get written back or not?
  - Did server do what I requested or not?
- Answer? Distributed transactions/Byzantine Commit

Problems with RPC: Performance

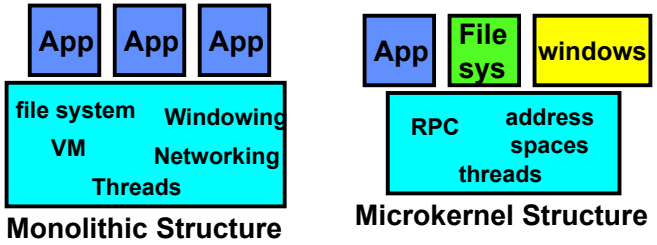
- Cost of Procedure call « same-machine RPC « network RPC
- Means programmers must be aware that RPC is not free
  - Caching can help, but may make failure handling complex

Cross-Domain Communication/ Location Transparency

- How do address spaces communicate with one another?
  - Shared Memory with Semaphores, monitors, etc…
  - File System
  - Pipes (1-way communication)
  - "Remote" procedure call (2-way communication)
- RPC's can be used to communicate between address spaces on different machines or the same machine
  - Services can be run wherever it's most appropriate
  - Access to local and remote services looks the same
- Examples of RPC systems:
  - CORBA (Common Object Request Broker Architecture)
  - DCOM (Distributed COM)
  - RMI (Java Remote Method Invocation)

Microkernel operating systems

- Example: split kernel into application-level servers.
  - File system looks remote, even though on same machine

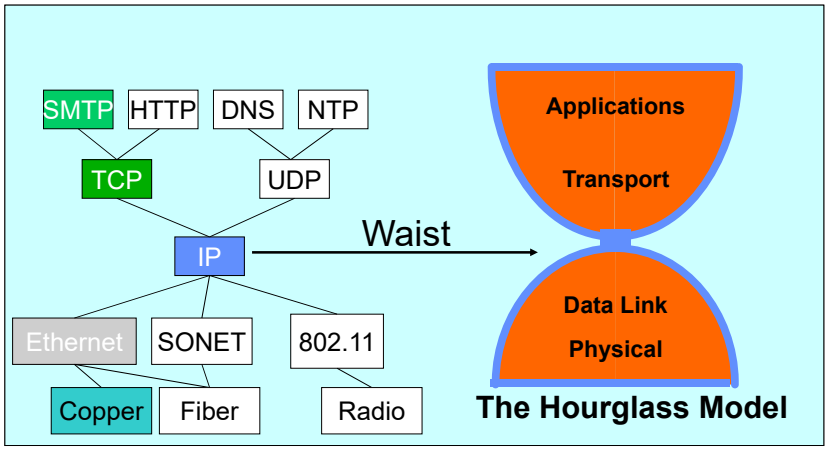


- Why split the OS into separate domains?
  - Fault isolation: bugs are more isolated (build a firewall)
  - Enforces modularity: allows incremental upgrades of pieces of software (client or server)
  - Location transparent: service can be local or remote
    - » For example in the X windowing system: Each X client can be on a separate machine from X server; Neither has to run on the machine with the frame buffer.

Administrivia

- Last Midterm: 5/2
  - Can have 3 handwritten sheets of notes – both sides
  - Focus on material from lecture 17-24, but all topics fair game!
- Please come to class on 4/30!
  - HKN evaluations!
- Don't forget to do your group evaluations!
  - Very important to help us understand your group dynamics
  - Important to do this for Project 3 as well!
    - » Even though it will be after Midterm 3!

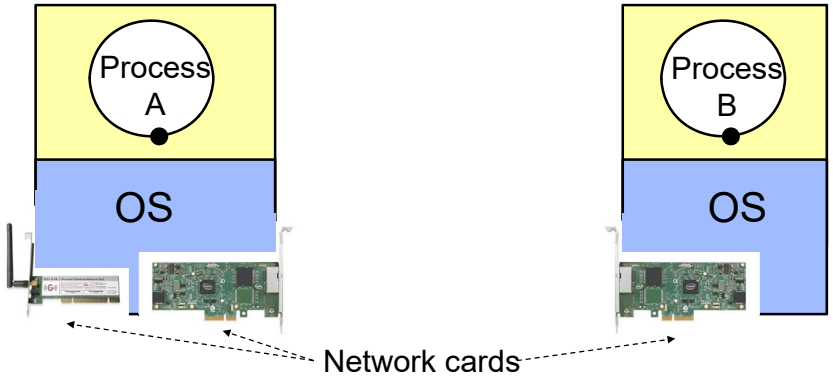
Recall: The Internet *Hourglass*



There is just one network-layer protocol, **IP**.
The "narrow waist" facilitates interoperability.

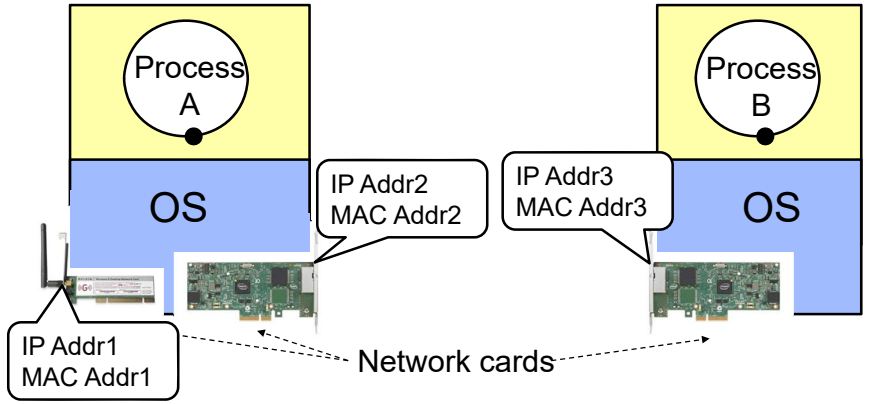
Network Concepts

- **Network (interface) card/controller**: hardware that physically connects a computer to the network
- A computer can have more than one networking card
  - E.g., one card for wired network, and one for wireless network



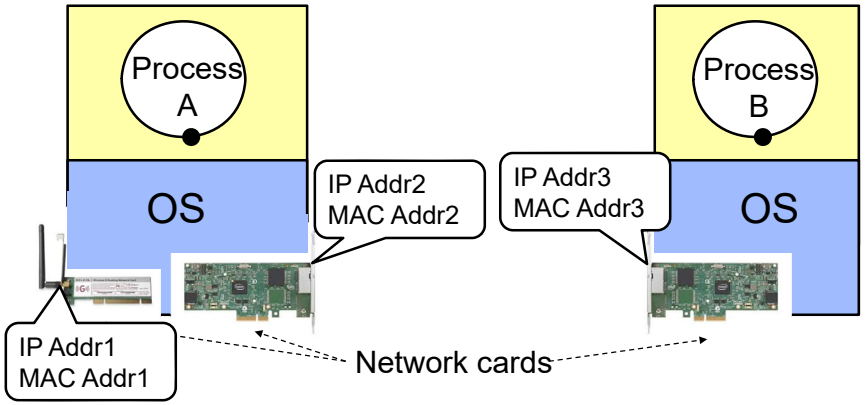
Network Concepts (cont'd)

- Typically, each network card is associated two addresses:
  - Media Access Control (MAC), or physical, address
  - IP, or network, address (can be shared by network cards on same host)



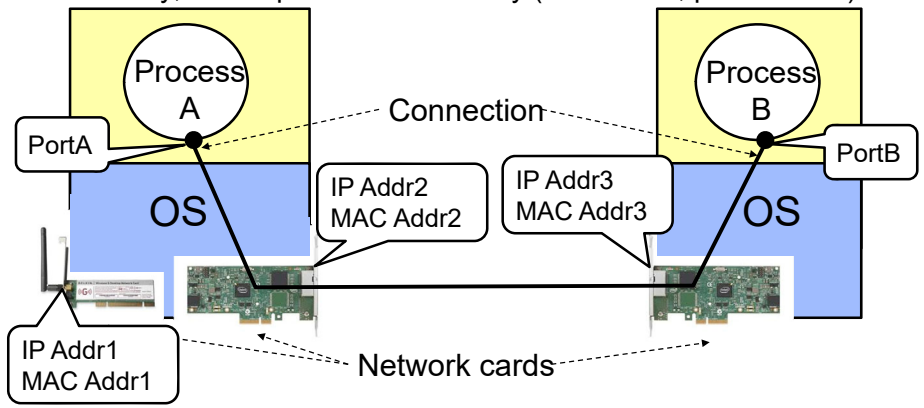
Network Concepts (cont'd)

- **MAC address**: 48-bit unique identifier assigned by card vendor
- **IP Address**: 32-bit (or 128-bit for IPv6) address assigned by network administrator or dynamically when computer connects to network



Network Concepts (cont'd)

- **Connection**: communication channel between two processes
- Each endpoint is identified by a **port number**
  - **Port number**: 16-bit identifier assigned by app or OS
  - Globally, an endpoint is identified by (IP address, port number)



Network Layering

- Layering: building complex services from simpler ones
  - Each layer provides services needed by higher layers by utilizing services provided by lower layers
- The physical/link layer is pretty limited
  - Packets are of limited size (called the "Maximum Transfer Unit or MTU: often 200-1500 bytes in size)
  - Routing is limited to within a physical link (wire) or perhaps through a switch
- Our goal in the following is to show how to construct a secure, ordered, message service routed to anywhere:

Physical Reality: Packets	Abstraction: Messages
Limited Size	Arbitrary Size
Unordered (sometimes)	Ordered
Unreliable	Reliable
Machine-to-machine	Process-to-process
Only on local area net	Routed anywhere
Asynchronous	Synchronous
Insecure	Secure

Main Network Functionalities

- **Delivery**: deliver packets between any two hosts in the Internet
  - E.g., how do you deliver a packet from a host in Berkeley to a host in Tokyo?
  - Datagram: an unreliable, unordered, packet sent from source to destination
- **Reliability**: tolerate packet losses
  - E.g., how do you ensure all bits of a file are delivered in the presence of packet loses?
- **Flow control**: avoid overflowing the receiver buffer
  - Recall our bounded buffer example: stop sender from overflowing receiver's buffer
    - » E.g., how do you ensure that a sever that can send at 10Gbps doesn't overwhelm a mobile phone?
- **Congestion control**: avoid overflowing the buffer of a router along the path
  - What happens if this happens?

Protocol Standardization

- Ensure communicating hosts speak the same protocol
  - Standardization to enable multiple implementations
  - Or, the same folks have to write all the software
- Standardization: Internet Engineering Task Force
  - Based on working groups that focus on specific issues
  - Produces "Request For Comments" (RFCs)
    - » Promoted to standards via rough consensus and running code
  - IETF Web site is ***http://www.ietf.org***
  - RFCs archived at ***http://www.rfc-editor.org***
- De facto standards: same folks writing the code
  - P2P file sharing, Skype, <your protocol here>…

Network System Modularity

Like software modularity, but:

- Implementation distributed across many machines (routers and hosts)
- Must decide:
  - How to break system into modules
    - » **Layering**
  - What functionality does each module implement
    - » **End-to-End Principle**
  - Where state is stored
    - » **Fate-sharing**
- We will address these choices in turn

Layering: A Modular Approach

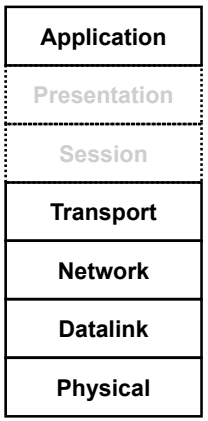
- Partition the system
  - Each layer solely relies on services from layer below
  - Each layer solely exports services to layer above
- Interface between layers defines interaction
  - Hides implementation details
  - Layers can change without disturbing other layers

Properties of Layers (OSI Model)

- **Service**: what a layer does
- **Service interface**: how to access the service
  - Interface for layer above
- **Protocol** (*peer interface*): how peers communicate to implement the service
  - Set of rules and formats that specify the communication between network elements
  - Does ***not*** specify the implementation on a single machine, but how the layer is implemented ***between*** machines

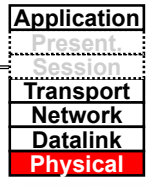
OSI Layering Model

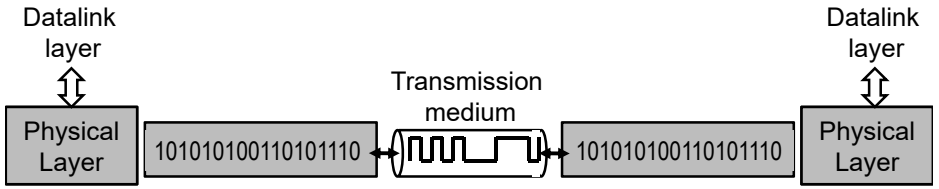
- Open Systems Interconnection (OSI) model
  - Developed by International Organization for Standardization (OSI) in 1984
  - **Seven** layers
- Internet Protocol (IP)
  - Only **five** layers
  - The functionalities of the missing layers (i.e., Presentation and Session) are provided by the Application layer



Physical Layer (1)

- **Service**: move information between two systems connected by a physical link
- **Interface**: specifies how to send and receive bits
- **Protocol**: coding scheme used to represent a bit, voltage levels, duration of a bit
- Examples: coaxial cable, optical fiber links; transmitters, receivers



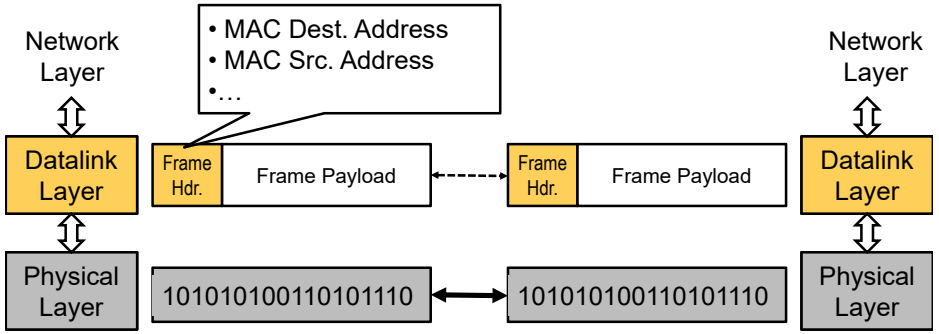


Datalink Layer (2)

- **Service**:
  - Enable end hosts to exchange frames (atomic messages) on the same physical line or wireless link
  - Possible other services:
    - » Arbitrate access to common physical media
    - » May provide reliable transmission, flow control
- **Interface**: send *frames* to other end *hosts*; receive *frames* addressed to end host
- **Protocols**: addressing, Media Access Control (MAC) (e.g., CSMA/CD - *Carrier Sense Multiple Access / Collision Detection*)

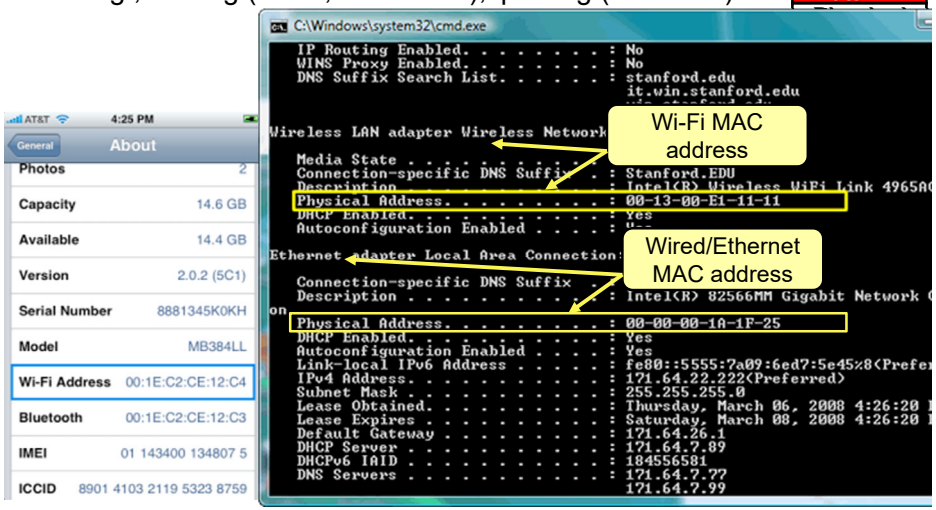
Datalink Layer (2)

- Each frame has a header which contains a source and a destination MAC address
- MAC (Media Access Control) address
  - Uniquely identifies a network interface
  - 48-bit, assigned by the device manufacturer



MAC Address Examples

- Can easily find MAC addr. on your machine/device:
  - E.g., ifconfig (Linux, Mac OS X), ipconfig (Windows)



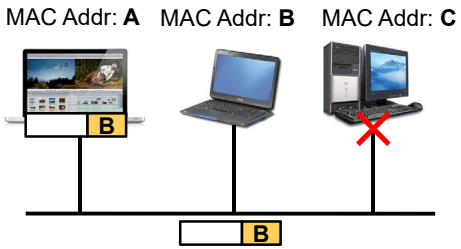
Local Area Networks (LANs)

- LAN: group of hosts/devices that
  - are in the same geographical proximity (e.g., same building, room)
  - use same physical communication technology
- Examples:
  - all laptops connected wirelessly at a Starbucks café
  - all devices and computers at home
  - all hosts connected to wired Ethernet in an office

Ethernet cable and port

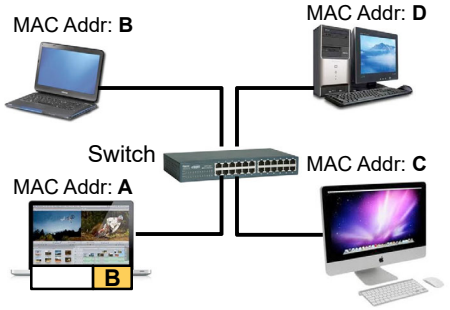
LANs

- All hosts in a LAN can share same physical communication media
  - Also called, broadcast channel
- Each frame is delivered to every host
- If a host is not the intended recipient, it drops the frame



Switches

- Hosts in same LAN can be also connected by switches
- A switch forwards frames only to intended recipients
  - Far more efficient than broadcast channel



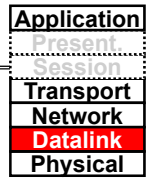
Media Access Control (MAC) Protocols

- Problem:
  - How do hosts access a broadcast media?
  - How do they avoid collisions?
- Three solutions:
  - Channel partition
  - "Taking turns"
  - Random access

MAC Protocols

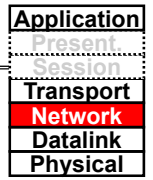
- Channel partitioning protocols:
  - Allocate 1/N bandwidth to every host
  - Share channel efficiently and fairly at high load
  - Inefficient at low load (where load = # senders):
    - » 1/N bandwidth allocated even if only 1 active node!
  - E.g., Frequency Division Multiple Access (FDMA); optical networks
- "Taking turns" protocols:
  - Pass a token around active hosts
  - A host can only send data if it has the token
  - More efficient at low loads: single node can use >> 1/N banwidth
  - Overhead in acquiring the token
  - Vulnerable to failures (e.g., failed node or lost token)
  - E.g., Token ring

MAC Protocols



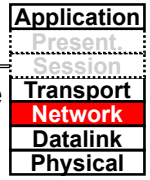
- Random Access
  - Efficient at low load: single node can fully utilize channel
  - High load: collision overhead
- Key ideas of random access:
  - Carrier sense (CS)
    - » Listen before speaking, and don't interrupt
    - » Checking if someone else is already sending data
    - » … and waiting till the other node is done
  - Multiple Access (MA) with Collision detection (CD)
    - » If someone else starts talking at the same time, stop
    - » Realizing when two nodes are transmitting at once
    - » …by detecting that the data on the wire is garbled
  - Adaptive Randomized Waiting Strategy
    - » Don't start talking again right away
    - » Waiting for a random time before trying again
    - » Exponential backoff: increase average wait time with each failure
- Examples: CSMA/CD, Ethernet, best known implementation

(Inter) Network Layer (3)

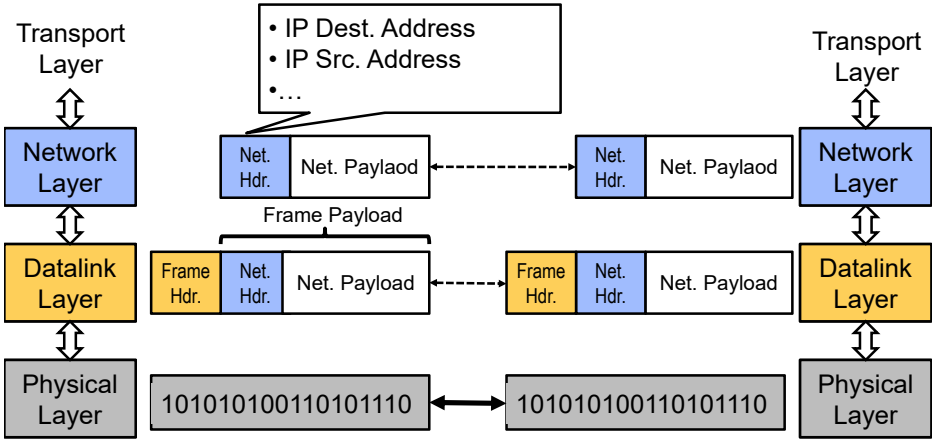


- **Service**:
  - Deliver packets to specified **network addresses** across multiple datalink layer networks
  - Possible other services:
    - » Packet *scheduling/priority*
    - » Buffer management
- **Interface**: send *packets* to specified network address destination; receive packets destined for end host
- **Protocols**: define network addresses (globally unique); construct forwarding tables; packet forwarding

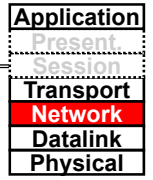
(Inter) Network Layer (3)



- **IP address**: unique addr. assigned to network device
- Assigned by network administrator or dynamically when host connects to network

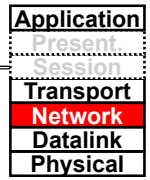


The Internet Protocol (IP)



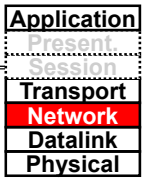
- Internet Protocol: Internet's network layer
- Service it provides: "Best-Effort" Packet Delivery
  - Tries it's "best" to deliver packet to its destination
  - Packets may be lost
  - Packets may be corrupted
  - Packets may be delivered out of order
- IP Is a Datagram service!

IPv4 Address Space



- IP Address: a 32-bit integer used as the destination of an IP packet
  - Often written as four dot-separated integers, with each integer from 0—255 (thus representing 8x4=32 bits)
  - Example CS file server is: 169.229.60.83 ≡ 0xA9E53C53
- Internet Host: a computer connected to the Internet
  - Host has one or more IP addresses used for routing
    - » Some of these may be private and unavailable for routing
  - Not every computer has a unique IP address
    - » Groups of machines may share a single IP address
    - » In this case, machines have private addresses behind a "Network Address Translation" (NAT) gateway
- Subnet: A network connecting a set of hosts with related destination addresses
  - A subnet is identified by 32-bit value, with the bits which differ set to zero, followed by a slash and a mask
    - » Example: 128.32.131.0/24 designates a subnet in which all the addresses look like 128.32.131.XX
    - » Same subnet: 128.32.131.0/255.255.255.0
  - Mask: The number of matching prefix bits
    - » Expressed as a single value (e.g., 24) or a set of ones in a 32-bit value (e.g., 255.255.255.0)

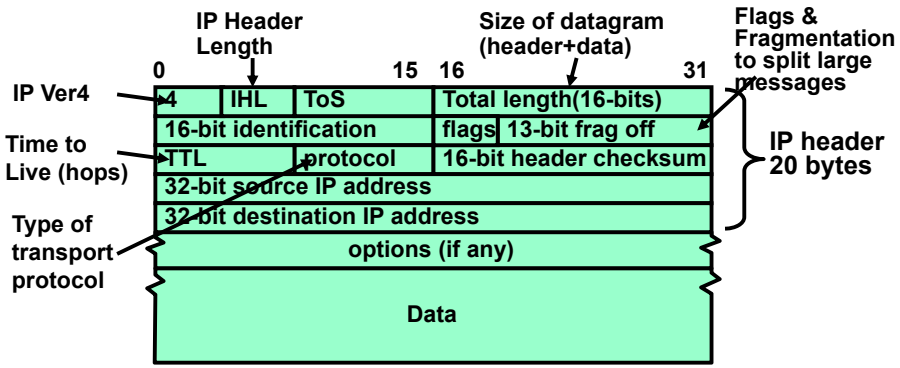
Address Ranges in IPv4



- IP address space divided into prefix-delimited ranges:
  - Class A: NN.0.0.0/8
    - » NN is 1–126 (126 of these networks)
    - » 16,777,214 IP addresses per network
    - » 10.xx.yy.zz is private
    - » 127.xx.yy.zz is loopback
  - Class B: NN.MM.0.0/16
    - » NN is 128–191, MM is 0-255 (16,384 of these networks)
    - » 65,534 IP addresses per network
    - » 172.[16-31].xx.yy are private
  - Class C: NN.MM.LL.0/24
    - » NN is 192–223, MM and LL 0-255 (2,097,151 of these networks)
    - » 254 IP addresses per networks
    - » 192.168.xx.yy are private
- Address ranges are often owned by organizations
  - Can be further divided into subnets

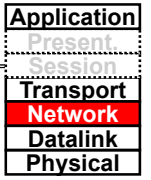
IPv4 Packet Format

- IP Packet Format:

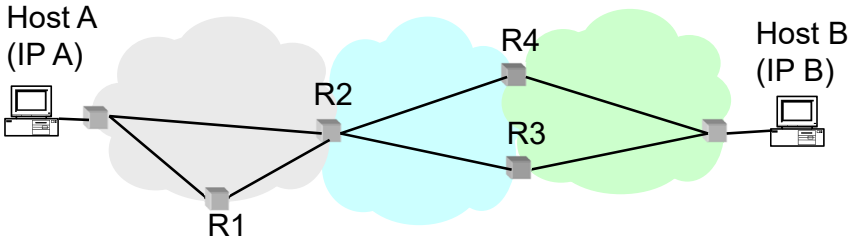


- IP Datagram: an unreliable, unordered, packet sent from source to destination
  - Function of network – deliver datagrams!

Wide Area Network

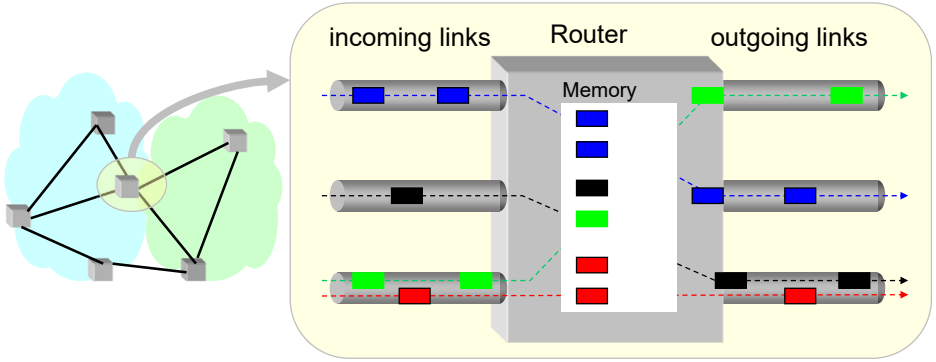


- **Wide Area Network** (WAN): network that covers a broad area (e.g., city, state, country, entire world)
  - E.g., Internet is a WAN
- WAN connects multiple datalink layer networks (LANs)
- Datalink layer networks are connected by **routers**
  - Different LANs can use different communication technology (e.g., wireless, cellular, optics, wired)



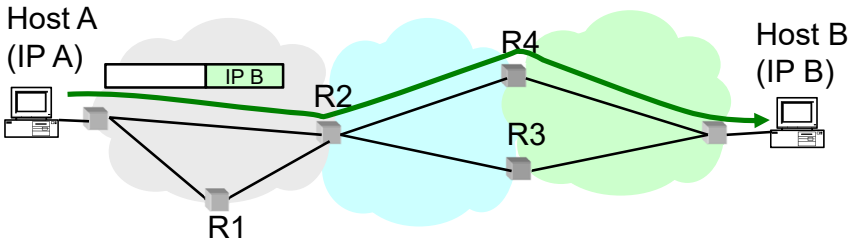
Routers

- **Forward** each packet received on an **incoming link** to an **outgoing link** based on packet's destination IP address (towards its destination)
- **Store & forward**: packets are buffered before being forwarded
- **Forwarding table**: mapping between IP address and the output link



Packet Forwarding

- Upon receiving a packet, a router
  - read the IP destination address of the packet
  - consults its forwarding table → output port
  - forwards packet to corresponding output port
- Default route (for subnets without explicit entries)
  - Forward to more authoritative router



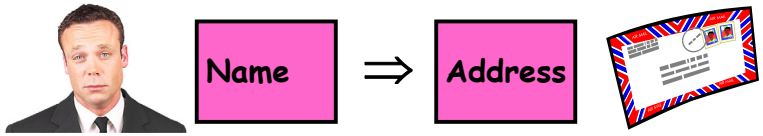
IP Addresses vs. MAC Addresses

- Why not use MAC addresses for routing?
  - Doesn't scale
- Analogy
  - MAC address → SSN
  - IP address → (unreadable) home address
- MAC address: uniquely associated to the device for the entire lifetime of the device
- IP address: changes as the device location changes
  - Your notebook IP address at school is different from home

IP Addresses vs. MAC Addresses

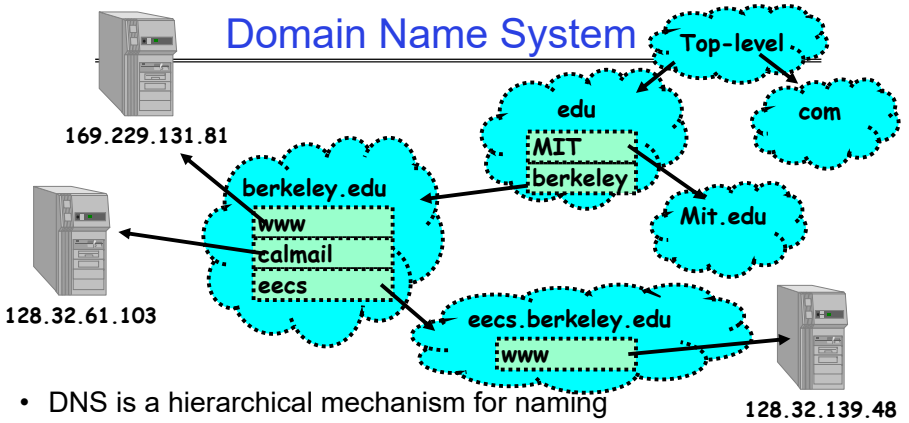
- Why does packet forwarding using IP addr. scale?
- Because IP addresses can be aggregated
  - E.g., all IP addresses at UC Berkeley start with **0xA9E5**, i.e., any address of form 0xA9E5**** belongs to Berkeley
  - Thus, a router in NY needs to keep a **single** entry for **all** hosts at Berkeley
  - If we were using MAC addresses the NY router would need to maintain **an entry for every** Berkeley host!!
- Analogy:
  - Give this letter to person with SSN: 123-45-6789 vs.
  - Give this letter to "John Smith, 123 First Street, LA, US"

Naming in the Internet



- How to map human-readable names to IP addresses?
  - E.g. www.berkeley.edu ⇒ 128.32.139.48
  - E.g. www.google.com ⇒ different addresses depending on location, and load
- Why is this necessary?
  - IP addresses are hard to remember
  - IP addresses change:
    - » Say, Server 1 crashes gets replaced by Server 2
    - » Or – google.com handled by different servers
- Mechanism: Domain Naming System (DNS)

Domain Name System

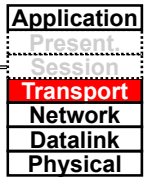


- DNS is a hierarchical mechanism for naming
  - Name divided in domains, right to left: www.eecs.berkeley.edu
- Each domain owned by a particular organization
  - Top level handled by ICANN (Internet Corporation for Assigned Numbers and Names)
  - Subsequent levels owned by organizations
- Resolution: series of queries to successive servers
- Caching: queries take time, so results cached for period of time

How Important is Correct Resolution?

- If attacker manages to give incorrect mapping:
  - Can get someone to route to server, thinking that they are routing to a different server
    - » Get them to log into "bank" – give up username and password
- Is DNS Secure?
  - Definitely a weak link
    - » What if "response" returned from different server than original query?
    - » Get person to use incorrect IP address!
  - Attempt to avoid substitution attacks:
    - » Query includes random number which must be returned
- In July 2008, hole in DNS security located!
  - Dan Kaminsky (security researcher) discovered an attack that broke DNS globally
    - » One person in an ISP convinced to load particular web page, then all users of that ISP end up pointing at wrong address
  - High profile, highly advertised need for patching DNS
    - » Big press release, lots of mystery
    - » Security researchers told no speculation until patches applied

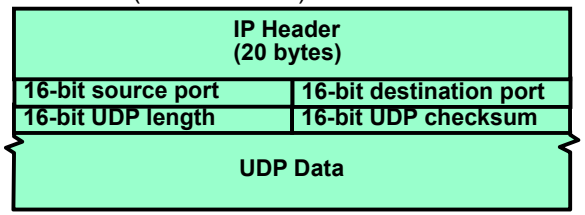
Transport Layer (4)



- **Service**:
  - Provide end-to-end communication between processes
  - Demultiplexing of communication between hosts
  - Possible other services:
    - » Reliability in the presence of errors
    - » Timing properties
    - » Rate adaptation (flow-control, congestion control)
- **Interface**: send message to specific process at given destination; local process receives messages sent to it
- **Protocol**: port numbers, perhaps implement reliability, flow control, packetization of large messages, framing
- Examples: TCP and UDP

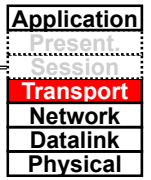
Building a messaging service on IP

- Process to process communication
  - Basic routing gets packets from machine→machine
  - What we really want is routing from process→process
    - » Add "ports", which are 16-bit identifiers
    - » A communication channel (connection) defined by 5 items: [source addr, source port, dest addr, dest port, protocol]
- For example: The Unreliable Datagram Protocol (UDP)
  - Layered on top of basic IP (IP Protocol 17)
    - » Datagram: an unreliable, unordered, packet sent from source user → dest user (Call it UDP/IP)

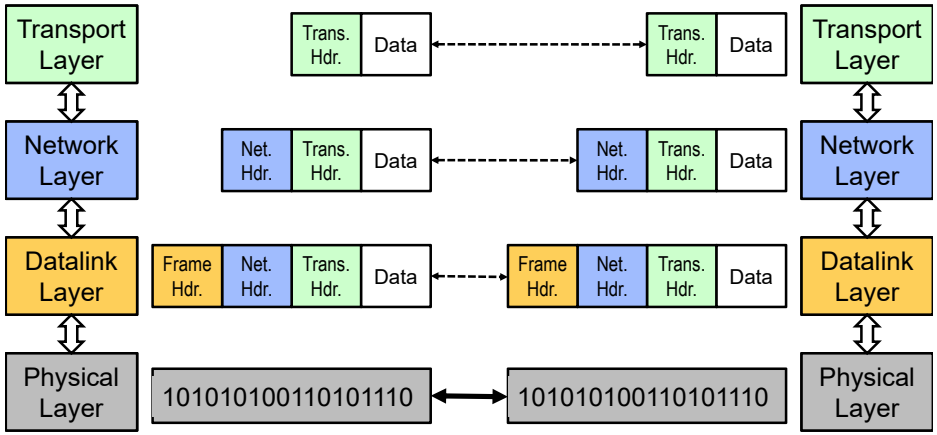


  - Important aspect: low overhead!
    - » Often used for high-bandwidth video streams
    - » Many uses of UDP considered "anti-social" – none of the "well-behaved" aspects of (say) TCP/IP

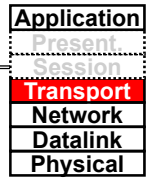
Process-to-Process Delivery (4)



- Five-tuple unique connection from process at one side to process at other side:
  - [Src Addr, Src Port, Dest Addr, Dest Port, Protocol]

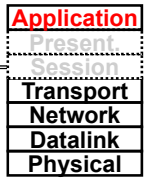


Internet Transport Protocols



- Datagram service (**UDP**)
  - No-frills extension of "best-effort" IP
  - Multiplexing/Demultiplexing among processes
- Reliable, in-order delivery (**TCP**)
  - Connection set-up & tear-down
  - Discarding corrupted packets (segments)
  - Retransmission of lost packets (segments)
  - Flow control
  - Congestion control
- Services not available
  - Delay and/or bandwidth guarantees
  - Sessions that survive change-of-IP-address
  - Security/denial of service resilience/…

Application Layer (7 - not 5!)

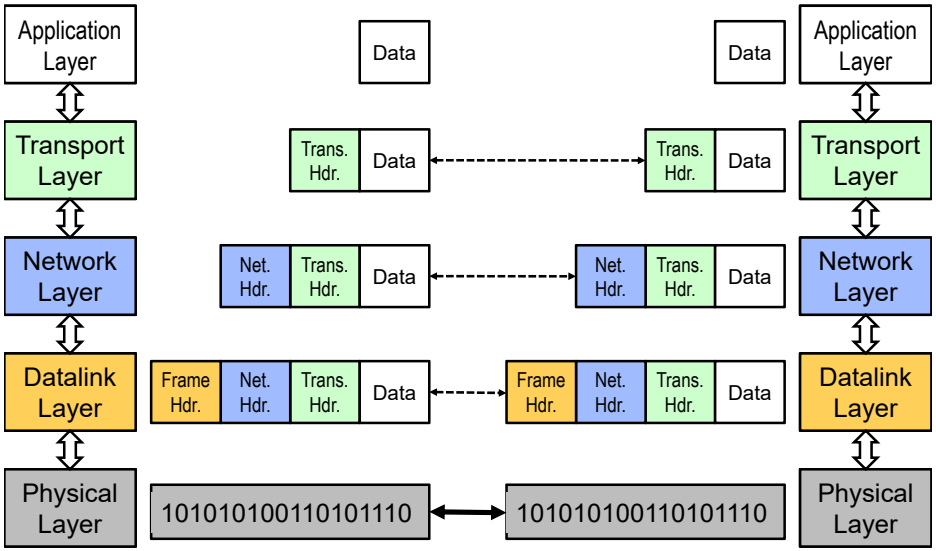


- **Service**: any service provided to the end user
- **Interface**: depends on the application
- **Protocol**: depends on the application

- Examples: Skype, SMTP (email), HTTP (Web), Halo, BitTorrent …

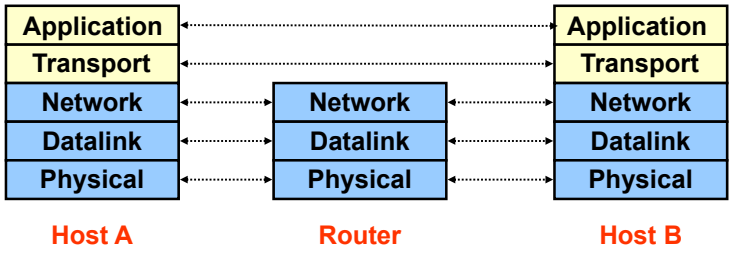
- What happened to layers 5 & 6?
  - "Session" and "Presentation" layers
  - Part of ***OSI*** architecture, but not Internet architecture
  - Their functionality is provided by application layer
    - » E.g. RPC is thought of a "session" protocol
    - » Encoding to handling mismatches in representation is a "Presentation" mechanism. MIME, SSL, XDR

Putting it all together



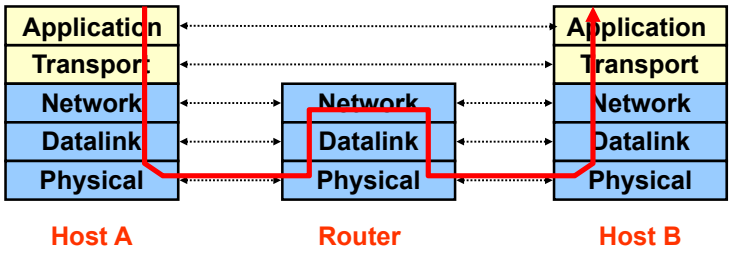
Five Layers Summary

- Lower three layers implemented everywhere
- Top two layers implemented only at hosts
- Logically, layers interacts with peer's corresponding layer



Physical Communication

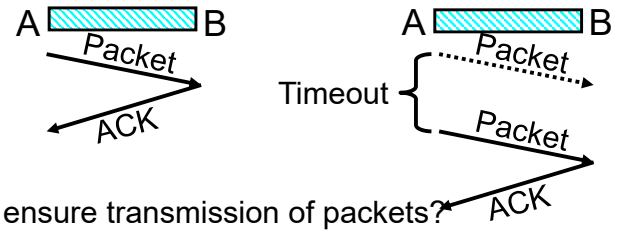
- Communication goes down to physical network
- Then from network peer to peer
- Then up to relevant layer



Reliable Message Delivery: the Problem

- All physical networks can garble and/or drop packets
  - Physical media: packet not transmitted/received
    - » If transmit close to maximum rate, get more throughput – even if some packets get lost
    - » If transmit at lowest voltage such that error correction just starts correcting errors, get best power/bit
  - Congestion: no place to put incoming packet
    - » Point-to-point network: insufficient queue at switch/router
    - » Broadcast link: two host try to use same link
    - » In any network: insufficient buffer space at destination
    - » Rate mismatch: what if sender send faster than receiver can process?
- Reliable Message Delivery on top of Unreliable Packets
  - Need some way to make sure that packets actually make it to receiver
    - » Every packet received at least once
    - » Every packet received at most once
  - Can combine with ordering: every packet received by process at destination exactly once and in order

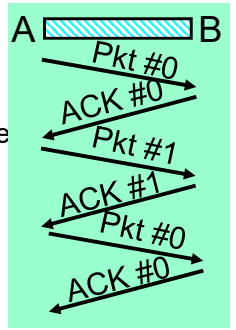
Using Acknowledgements



- How to ensure transmission of packets?
  - Detect garbling at receiver via checksum, discard if bad
  - Receiver acknowledges (by sending "ACK") when packet received properly at destination
  - Timeout at sender: if no ACK, retransmit
- Some questions:
  - If the sender doesn't get an ACK, does that mean the receiver didn't get the original message?
    - » No
  - What if ACK gets dropped? Or if message gets delayed?
    - » Sender doesn't get ACK, retransmits, Receiver gets message twice, ACK each

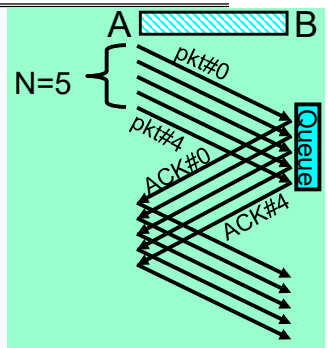
How to Deal with Message Duplication?

- Solution: put sequence number in message to identify re-transmitted packets
  - Receiver checks for duplicate number's; Discard if detected
- Requirements:
  - Sender keeps copy of unACK'd messages
    - » Easy: only need to buffer messages
  - Receiver tracks possible duplicate messages
    - » Hard: when ok to forget about received message?
- Alternating-bit protocol:
  - Send one message at a time; don't send next message until ACK received
  - Sender keeps last message; receiver tracks sequence number of last message receive
- Pros: simple, small overhead
- Con: Poor performance
  - Wire can hold multiple messages; want to fill up at (wire latency × throughput)
- Con: doesn't work if network can delay or duplicate messages arbitrarily

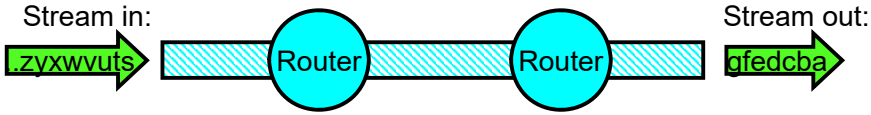


Better Messaging: Window-based Acknowledgements

- Windowing protocol (not quite TCP):
  - Send up to N packets without ack
    - » Allows pipelining of packets
    - » Window size (N) < queue at destination
  - Each packet has sequence number
    - » Receiver acknowledges each packet
    - » ACK says "received all packets up to sequence number X"/send more
- ACKs serve dual purpose:
  - Reliability: Confirming packet received
  - Ordering: Packets can be reordered at destination
- What if packet gets garbled/dropped?
  - Sender will timeout waiting for ACK packet
    - » Resend missing packets ⇒ Receiver gets packets out of order!
  - Should receiver discard packets that arrive out of order?
    - » Simple, but poor performance
  - Alternative: Keep copy until sender fills in missing pieces?
    - » Reduces # of retransmits, but more complex
- What if ACK gets garbled/dropped?
  - Timeout and resend just the un-acknowledged packets

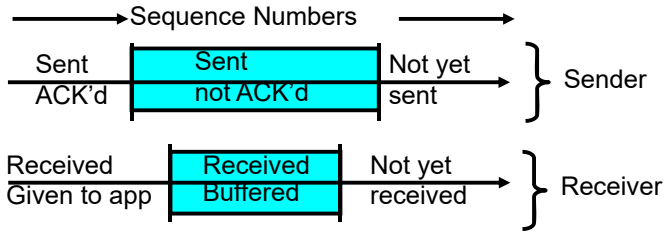


Transmission Control Protocol (TCP)



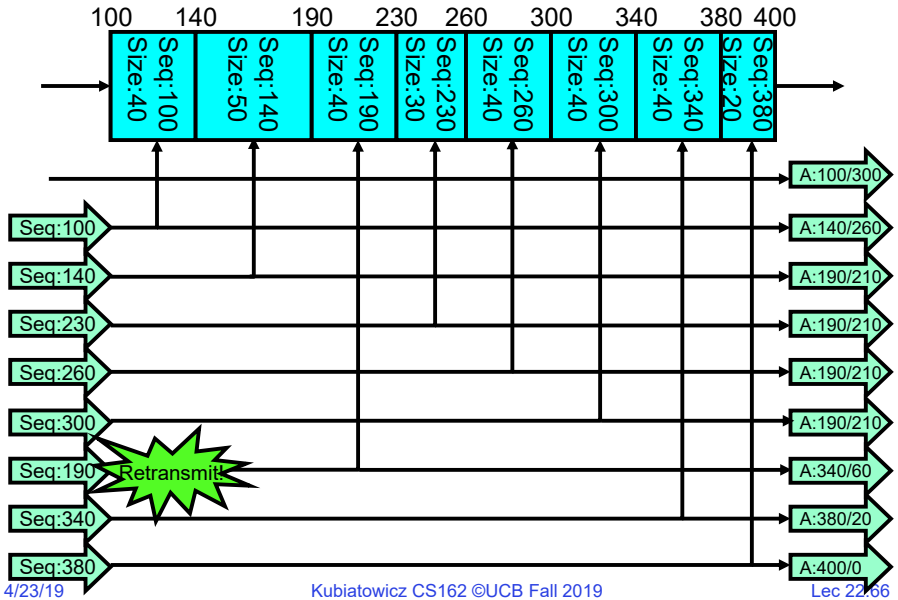
- Transmission Control Protocol (TCP)
  - TCP (IP Protocol 6) layered on top of IP
  - Reliable byte stream between two processes on different machines over Internet (read, write, flush)
- TCP Details
  - Fragments byte stream into packets, hands packets to IP
    - » IP may also fragment by itself
  - Uses window-based acknowledgement protocol (to minimize state at sender and receiver)
    - » "Window" reflects storage at receiver – sender shouldn't overrun receiver's buffer space
    - » Also, window should reflect speed/capacity of network – sender shouldn't overload network
  - Automatically retransmits lost packets
  - Adjusts rate of transmission to avoid congestion
    - » A "good citizen"

TCP Windows and Sequence Numbers



- Sender has three regions:
  - Sequence regions
    - » sent and ACK'd
    - » sent and not ACK'd
    - » not yet sent
  - Window (colored region) adjusted by sender
- Receiver has three regions:
  - Sequence regions
    - » received and ACK'd (given to application)
    - » received and buffered
    - » not yet received (or discarded because out of order)

Window-Based Acknowledgements (TCP)



Congestion Avoidance

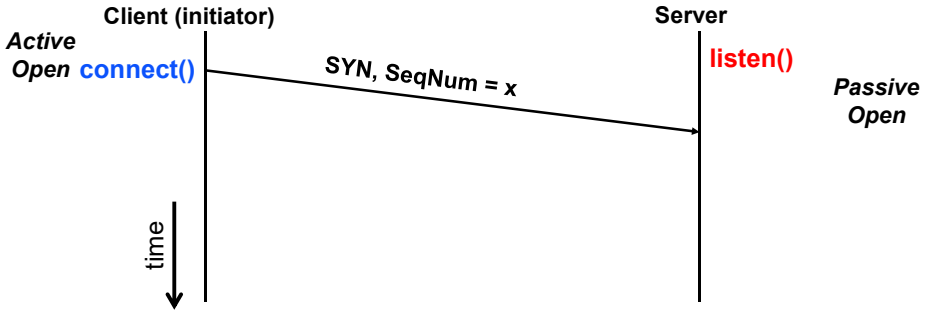
- Congestion
  - How long should timeout be for re-sending messages?
    - » Too long → wastes time if message lost
    - » Too short → retransmit even though ACK will arrive shortly
  - Stability problem: more congestion ⇒ ACK is delayed ⇒ unnecessary timeout ⇒ more traffic ⇒ more congestion
    - » Closely related to window size at sender: too big means putting too much data into network
- How does the sender's window size get chosen?
  - Must be less than receiver's advertised buffer size
  - Try to match the rate of sending packets with the rate that the slowest link can accommodate
  - Sender uses an adaptive algorithm to decide size of N
    - » Goal: fill network between sender and receiver
    - » Basic technique: slowly increase size of window until acknowledgements start being delayed/lost
- TCP solution: "slow start" (start sending slowly)
  - If no timeout, slowly increase window size (throughput) by 1 for each ACK received
  - Timeout ⇒ congestion, so cut window size in half
  - "*Additive Increase, Multiplicative Decrease*"

Open Connection: 3-Way Handshaking

- Goal: agree on a set of parameters, i.e., the start sequence number for each side
  - Starting sequence number (first byte in stream)
  - Must be unique!
    - » If it is possible to predict sequence numbers, might be possible for attacker to hijack TCP connection
- Some ways of choosing an initial sequence number:
  - Time to live: each packet has a deadline.
    - » If not delivered in X seconds, then is dropped
    - » Thus, can re-use sequence numbers if wait for all packets in flight to be delivered or to expire
  - Epoch #: uniquely identifies *which* set of sequence numbers are currently being used
    - » Epoch # stored on disk, Put in every message
    - » Epoch # incremented on crash and/or when run out of sequence #
  - Pseudo-random increment to previous sequence number
    - » Used by several protocol implementations

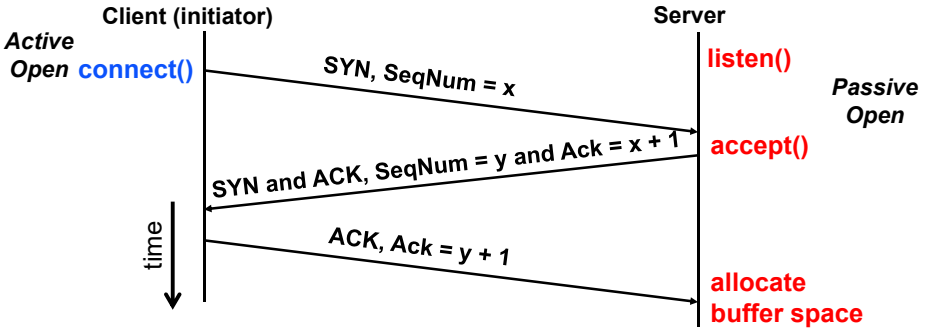
Open Connection: 3-Way Handshaking

- Server waits for new connection calling listen()
- Sender call connect() passing socket which contains server's IP address and port number
  - OS sends a special packet (SYN) containing a proposal for first sequence number, x



Open Connection: 3-Way Handshaking

- If it has enough resources, server calls accept() to accept connection, and sends back a SYN ACK packet containing
  - Client's sequence number incremented by one, (x + 1)
    - » Why is this needed?
  - A sequence number proposal, y, for first byte server will send

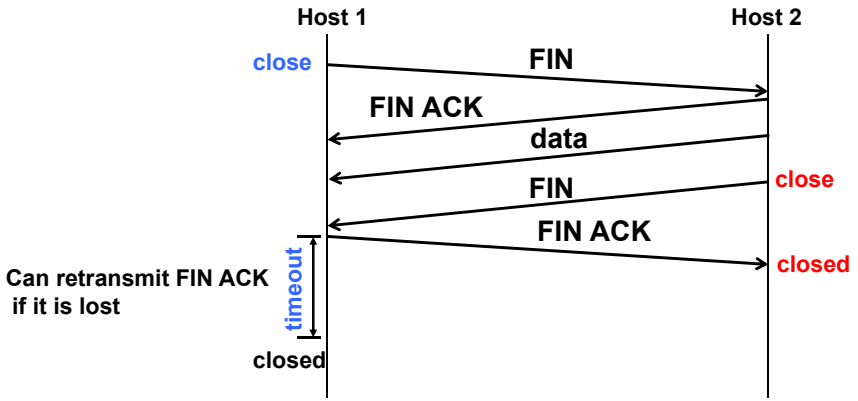


3-Way Handshaking (cont'd)

- Three-way handshake adds 1 RTT delay
- Why do it this way?
  - Congestion control: SYN (40 byte) acts as cheap probe
  - Protects against delayed packets from other connection (would confuse receiver)

Close Connection

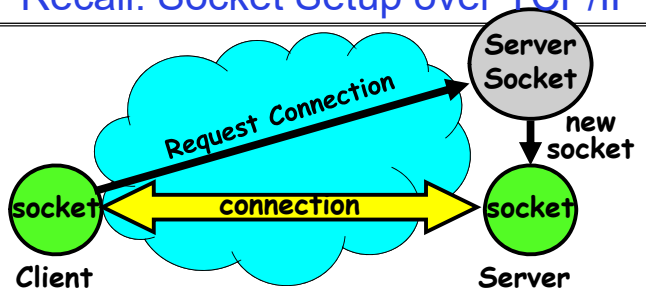
- Goal: both sides agree to close the connection
- 4-way connection tear down



Use of TCP: Sockets

- Socket: an abstraction of a network I/O queue
  - Embodies one side of a communication channel
    - » Same interface regardless of location of other end
    - » Could be local machine (called "UNIX socket") or remote machine (called "network socket")
  - First introduced in 4.2 BSD UNIX: big innovation at time
- Using Sockets for Client-Server (C/C++ interface):
  - On server: set up "server-socket"
    - » Create socket, Bind to protocol (TCP), local address, port
    - » Call listen(): tells server socket to accept incoming requests
    - » Multiple accept() calls on socket to accept incoming connection requests
    - » Each successful accept() returns a new socket for a new connection
  - On client:
    - » Create socket, Bind to protocol (TCP), remote address, port
    - » Perform connect() on socket to make connection
    - » If connect() successful, have socket connected to server
- Network Address Translation (NAT):
  - Local subnet (non-routable IP addresses) ⇒external IP
  - Client-side firewall replaces local IP address/port combination with external IP address/new port
  - Firewall handles translation between different address domains using table of current connections

Recall: Socket Setup over TCP/IP



- Things to remember:
  - Connection involves 5 values:
    [Client Addr, Client Port, Server Addr, Server Port, Protocol]
  - Often, Client Port "randomly" assigned
  - Server Port often "well known"
    - » 80 (web), 443 (secure web), 25 (sendmail), etc
    - » Well-known ports from 0—1023
- Network Address Translation (NAT) allows many internal connections (and/or hosts) with a single external IP address

Summary (1/2)

- Remote Procedure Call (RPC): Call procedure on remote machine
  - Provides same interface as procedure
  - Automatic packing and unpacking of arguments without user programming (in stub)
- Internet: 5 layers
  - Physical: send bits
  - Datalink: Connect two hosts on same physical media
  - Network: Connect two hosts in a wide area network
  - Transport: Connect two processes on (remote) hosts
  - Applications: Enable applications running on remote hosts to interact
- Unified Internet layering (Application/Transport/ Internetwork/Link/Physical) decouples apps from networking technologies

Summary (2/2)

- Internet Protocol (IP): Datagram packet delivery
  - Used to route messages through routes across globe
  - 32-bit addresses, 16-bit ports
- DNS: System for mapping from names⇒IP addresses
  - Hierarchical mapping from authoritative domains
  - Recent flaws discovered
- Ordered messages:
  - Use sequence numbers and reorder at destination
- Reliable messages:
  - Use Acknowledgements
- TCP: Reliable byte stream between two processes on different machines over Internet (read, write, flush)
  - Uses window-based acknowledgement protocol
  - Congestion-avoidance dynamically adapts sender window to account for congestion in network