14-848: CLOUD INFRASTRUCTURE

"WHAT IS A CLOUD? * NETWORKING IN A DAY

LECTURE 2 * FALL 2018 (KESDEN)

REMEMBER SOCRATIVE

- https://api.socrative.com/rc/Nfu6Lp

What's "A Cloud" ?

B.C. BEFORE CLOUD

- Let's remiss about the "Good Ole' Days"
  - Software ran on computers
  - Computers used storage and other peripherals
  - A network (often) connected the computers, storage, and other peripherals
  - One bought computers, network gear, storage, other peripherals, and software
  - One hired people to maintain computers, storage, network gear, other peripherals, and software

CHALLENGES OF LIFE B.C.

- It cost money to buy all of the things and all of the people
- Scaling was hard because it involved buying new things, getting them delivered, getting them installed, and managing them.
  - This might involve getting more people.
- Partitioning of resources across business units meant overprovisioning.
  - Unifying them meant losing the ability to tailor offerings.
- The same factors made robustness hard
- Bursty loads were necessarily inefficient.
  - Overprovision? Or underprovision?
- Technology changes were hard.
  - How to evaluate? Transition?

UTILITY, *DEF.*

- A service or commodity for which cost is metered by use
  - Electricity, natural gas, water, some types of data service, etc.
  - Often times publicly, commonly or ubiquitously available

- Wouldn't it be nice if computing infrastructure were a utility?
  - Buy as much as needed, whenever needed
  - None wasted
  - Economy of scale averages risk, promotes availability, and drives consumption
    - Making provisioning for robustness and scalability more cost-effective

CLOUD COMPUTING vs UTILITY COMPUTING

- *Utility computing* is the idea of providing metered services
  - Often in place of what would previously been discrete purchases
  - Often possible through shared infrastructure
- *Cloud computing* is the idea of providing *utility computing* through a shared infrastructure
  - May be a *public cloud* shared by external users
  - May be a *private cloud* shared by internal users
- Fundamentally, clouds provide services

WHAT CAN BE A SERVICE?

- Anything that used to be a product
  - Software
  - Computing
  - Storage
  - Other peripherals
  - Platforms, e.g. APIs and the systems providing them

(SOME) PEOPLE LIKE ACRONYMS

- Software as a Service (Saas)
  - Application software provided as utility
    - Think about Google Apps, Microsoft 360, etc.
- Platforms as a Service (Paas)
  - Ecosystems provided as a utility
    - Salesforce.com is my favorite domain-specific example
    - AWS, Microsoft Azure, Google AppEngine,
- Infrastructure as a Service (IaaS)
  - What would otherwise be hardware provided as a utility
    - Storage, Computing, Networking, etc.
    - Think about S3, EC^2, Amazon *VPC, etc.*

KEY ENABLING TECHNOLOGY: VIRTUALIZATION

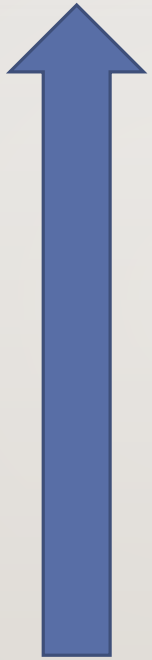
- Virtual, *def*: Not really existing, but appear to exist
  - *Alt. def:* Having the good qualities of something, without having its physical form
- Virtualization, *def:* The act of creating a virtual, rather than physical, instance of something
- Virtualization enables the sharing of resources
  - This decouples the services from the hardware and people that provide them, enabling everything else

KEY CHALLENGES OF CLOUD COMPUTING

- Isolating while sharing
  - Need to be able to protect privacy
    - Data, methods, organization, etc.
  - Need to be able to isolate performance impact
  - Need to do it in a dynamic, but trustworthy way
- Elastic Provisioning
  - Enables utility computing
  - Enables energy savings, etc
  - Requires virtualization of all aspects, e.g. computing, storage, network, etc.
- Managing scale
  - For example, communication and other interaction complexity often naturally grows super-linearly

BUILDING THE CLOUD

- Providing Services, e.g. Frameworks
  - MRv2, Spark, etc
- Elasticity and Provisioning
  - YARN, Spark, etc
- Computing
- Caching
- Storage
  - Files
  - Keys and Values
  - SQL and NoSQL Databases
- The Network



We'll work from the bottom up

Networking:

A Quick Primer

(And then on to building data centers)

NETWORK REFERENCE MODEL

- Application – Establish an idiom for communicating with a particular application
- Transport – Establish endpoints useful to a programmer
- Network – Given multiple inter-connected LANs, achieve cross-connectivity,
- Link – Manage the channel to enable actual communication, i.e. establish a LAN
- Physical – Establish a channel with connectivity and signaling

PHYSICAL LAYER: ESTABLISHES THE CHANNEL

- Medium? Light? Radio frequency? Electrical signals?
  - What color(s) of light? How bright?
  - What RF frequencies? How powerful?
  - What signals represent what values?
  - What shape are the connectors?
  - How far can cables run?
  - Etc.
- We have a functioning physical layer once we have the ability to send and receive signals

PHYSICAL LAYER: BANDWIDTH VS LATENCY

- Bandwidth = bits/second.
  - Improved with parallelism or faster clock rate
- Latency = Function of signal propagation speed
  - Limited by speed of light
  - Major paradigm shift would be needed to make traffic to India or China less latent
- Latency tends to be limiting at a global scale
  - Speed of light over long distances
- Bandwidth tends to be limited at local scale, e.g data center
  - How to divide up and recombine messages to utilize parallelism?
  - How to clock faster without losing signal to noise.

LINK LAYER: MANAGES THE CHANNEL

- When do we start transmitting? When do we stop?
- When do we start receiving? When do we stop?
- Who is sending? Who is receiving?
- How do we know if it is correct?
- What happens if there is contention for, or collision in, a shared channel?
- Key contributions: Framing, among others
- We have a functioning link layer once we can build a functioning LAN of at least two stations.

NETWORK LAYER: SCALING UP

- Passing messages among multiple networks
  - For scale
  - Of different types (wired, wireless, fiber, infrared, etc)
  - Managed by different domains, etc.
- Globally meaningful addressing
- Ability to choose paths among multiple options
- We have a functioning network layer once we can connect multiple networks, identify hosts among them, and messages can find their way across networks from source to destination.

TRANSPORT LAYER: MEANINGFUL ENDPOINTS

- Hosts don't communication – various aspects of software systems do
  - Consider how many different sessions your Web browser has with servers. Now add for your IM sessions, upgrades-in-progress, music streaming, etc.
- Endpoints enable the establishment of sessions
  - Classic model is <<IP:port>:<IP:port>>
  - Client: Ephemeral port
  - Host: Well-known port

TRANSPORT LAYER:MEANINGFUL ENDPOINTS, *CONT.*

- Character of communication
  - Reliable/session-oriented, e.g. TCP
  - Unreliable/datagram, e.g. UDP
  - Etc.
- The transport layer exists once we have the ability to establish communication from end-point to end-point with well-understood properties

APPLICATION LAYER: PURPOSEFUL COMMUNICATION

- Defined by the messaging we, as programs, bake into our applications, shaped by our applications,
  - e.g. client-server interactions, peer-to-peer interactions, etc.
- E.g. HTTP: PUT, GET, POST, etc
- E.g DNS: queries, responses, updates, etc.
- MIME, VOIP protocols, etc.
- Application protocols exist when applications can communicate

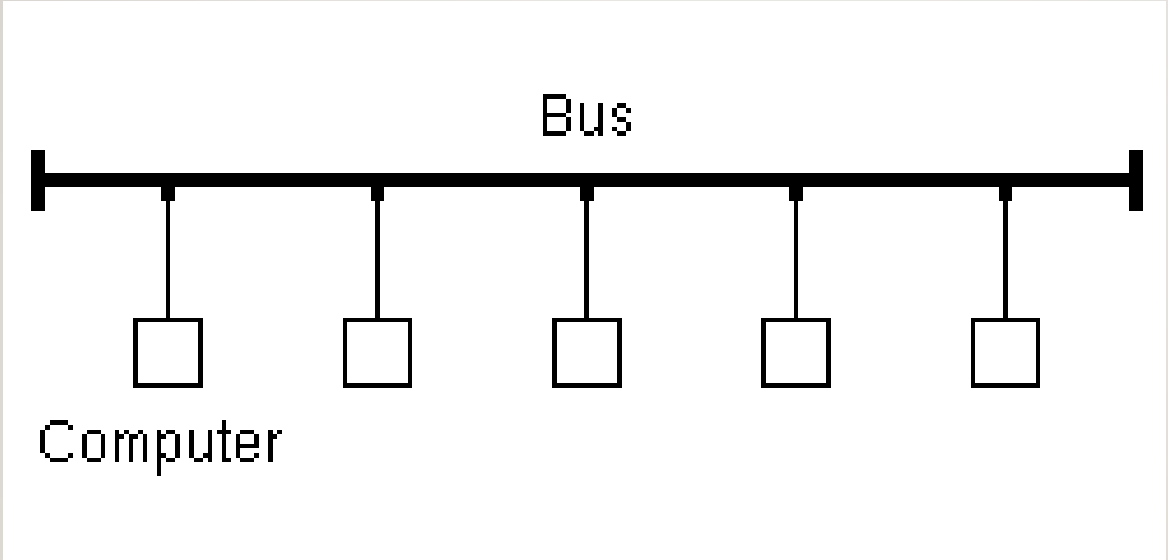
QUICK EVOLUTION OF LANS: SIMPLE LANS

- Two or more hosts share a common physical medium, e.g. ethernet
- Physical protocols define hardware
- Link-layer protocols manage it
- Point-to-point, e.g. fiberoptic
- Broadcast, e.g. ethernet

QUICK EVOLUTION OF LANS: WIRED LIMITS

- Consider ethernet: Multiple stations share a common wire
- What happens if more than one transmits at the same time?
  - Collision, e.g. corruption
  - Link layer manages collision, e.g. exponential back-off, jamming signals, etc.
- Wire can only get so long
  - Attenuation, power to drive it, noise, etc.
  - Mess of actually getting the wire through the building, etc.
- Can only have a certain number of stations
  - Too little network time per each, otherwise
- Aside: Wireless has similar limits, too.

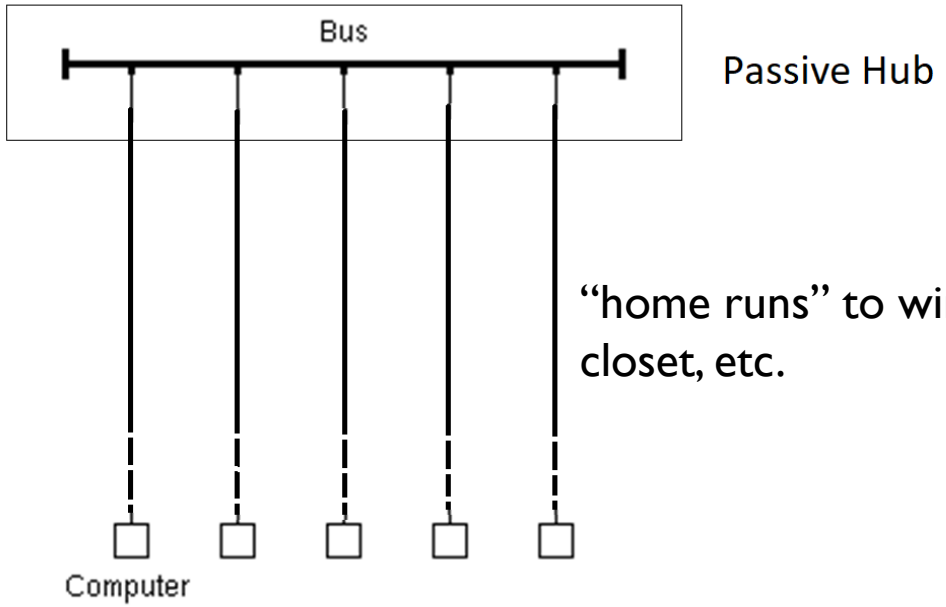
QUICK EVOLUTION OF LANS: BUS TOPOLOGY



Imagine having to snake one wire around the building!

https://upload.wikimedia.org/wikipedia/commons/9/9e/Bustopologie.png

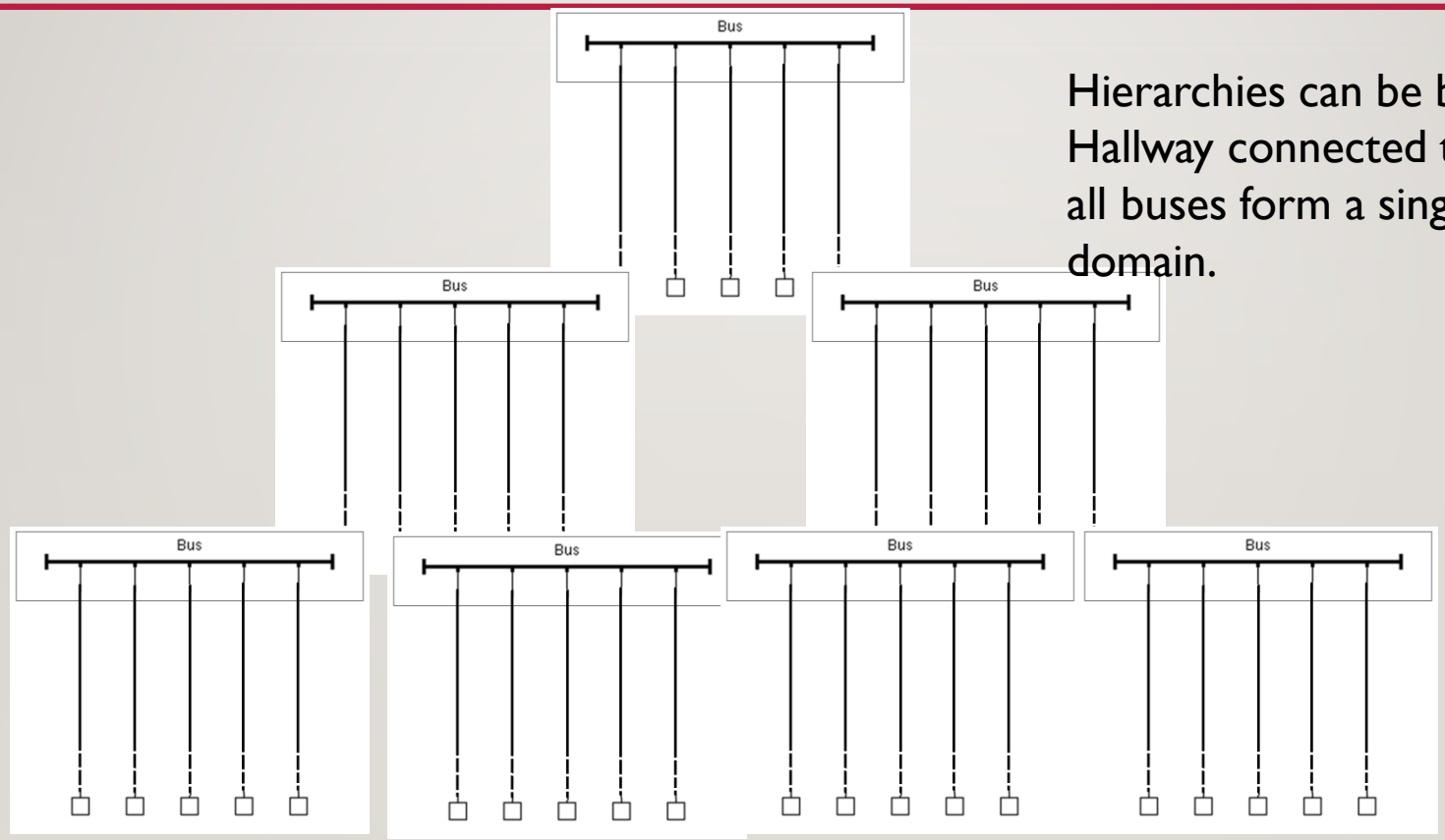
QUICK EVOLUTION OF LANS: HUB TOPOLOGY



The bus remains an equivalent collision/contention domain, but the wiring gets easier, physically.

Adapted from: https://upload.wikimedia.org/wikipedia/commons/9/9e/Bustopologie.png

QUICK EVOLUTION OF LANS: HUB HIERARCHY



Hierarchies can be built, e.g. one hub per Hallway connected to one hub for the floor. But, all buses form a single collision/contention domain.

QUICK EVOLUTION OF LANS: NETWORK SWITCHES

- Enable connection of input and output port pairs without sharing a single common channel
  - Crossbar switch: Mess of switched connections
  - Input and output buffering with shared memory and control
  - Etc
- Learning
  - Pay attention when host sends to learn which port it is on, then direct messages to that host only to that port
  - Flood all ports only when destination unknown.
  - Enables larger networks
- Terminology note: A *bridge* is a simple *switch* with only two ports.

QUICK EVOLUTION OF LANS: LIMITS

- Even with switching, there is a limit to the size of a LAN
- In the worst case, a host which is not known, the entire LAN is still a single contention/collision domain
  - If a host hasn't yet sent, or hasn't sent recently enough to be cached, flooding will be needed
  - The flooding can, in the worst case, flood every port on every switch
- There obviously is no way to know the location of every host on the Internet
- And, of course, networks use different technologies, are managed by different domains, etc.

LANS, TODAY

- Switches provide a fabric
  - This fabric is virtualized into "VLANs" or *Virtual LANs*
    - Logical LANs projected onto the same hardware
  - This is one technique for sharing a physical LAN

THE NETWORK LAYER

- Hierarchical addressing
  - Traditional IPv4 addresses: 32-bits: Network# + Host Number
    - IPv6 addresses: 128-bits and more structured
  - Host number translated to LAN station ID, e.g by ARP between IPv4 and 802.11
- Routing selects path to take from one network to another, often on a hop-by-hop basis
  - Does this packet belong on one of my LANs? If so ARP and deliver
    - If not, send upstream (or to a peer, or…)
- This provides for an order of magnitude more hosts

THE NETWORK LAYER: IP ADDRESS ASSIGNMENT

- Old school: Go to system administrator and trade MAC address for IP address
- Today: DHCP server automates this. Broadcast of request with MAC is answered with assigned IP
  - Assigned IP is leased and needs to be renewed
  - Assigned IP can be from dynamic pool
  - Assigned IP can also be according to a pre-configured rule, such as to give a server a well-known address
  - DHCP can also communicate other configuration information

DOMAIN NAME SYSTEM

- Old school
  - Let "The Keeper of All Things" know about a hostname:IP assignment in your organization
  - The Keeper updates a "hosts" text file with the information
  - Periodically download this file to keep your system up to date
  - Obvious scalability problems, but /etc/hosts still exists vestigially and for special cases
- Today
  - Domain Name System (DNS) is a distributed data base that delegates assignments for information to the responsible domains and can direct queries to the servers associated with those domains.
  - Uses caching for efficiency.

TRANSPORT LAYER: USER DATAGRAM PROTOCOL (UDP)

- Reminder: Port numbers in addition to IP addresses
- Best effort = Unreliable
  - Messages can be lost or reordered
  - Message corruption is assumed to be detected at the link layer
- Message oriented. Max message size
- Simple
- Used for timely updates, e.g. send audio or video for teleconferencing

TRANSPORT LAYER: RELIABLE PROTOCOLS

- Keeps trying to send data until it succeeds or times out
- Used acknowledgements to determine that it does not need to resent
- Buffers to ensure in-order delivery, which allows head-of-line blocking
- Messages are assumed to be correct or undelivered via checksums at link layer
  - "Byzantine Failures" are possible, occur commonly at Internet scale, but infrequent enough to be (mostly) ignored. Maybe.
- Can't guarantee delivery.
  - At best can trade timeliness for delivery

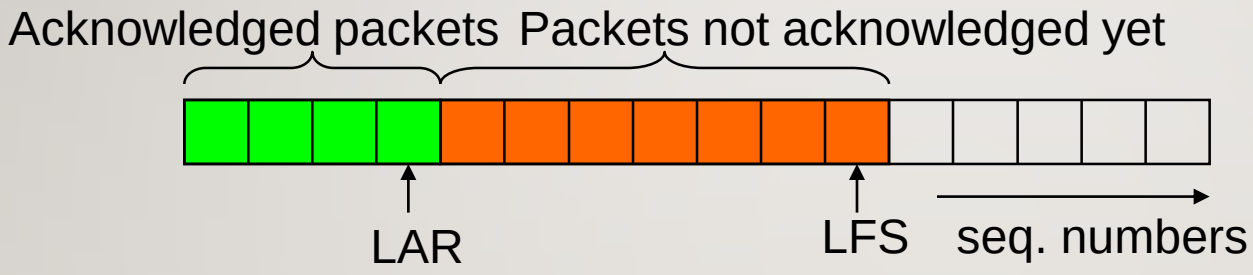
TRANSPORT LAYER: STOP-AND-WAIT PROTOCOLS

- Send a message. Wait one fully network latency for it to get to the recipient. Wait for the recipient to process it. Wait another full network latency to send back the acknowledgement
  - In one round-trip time (RTT), only one message is sent.
  - RTT sec * bits/sec = total bits we can send in that time.
  - Size of message is what we actually sent. Rest of time is wasted waiting.

TRANSPORT LAYER: SLIDING WINDOW

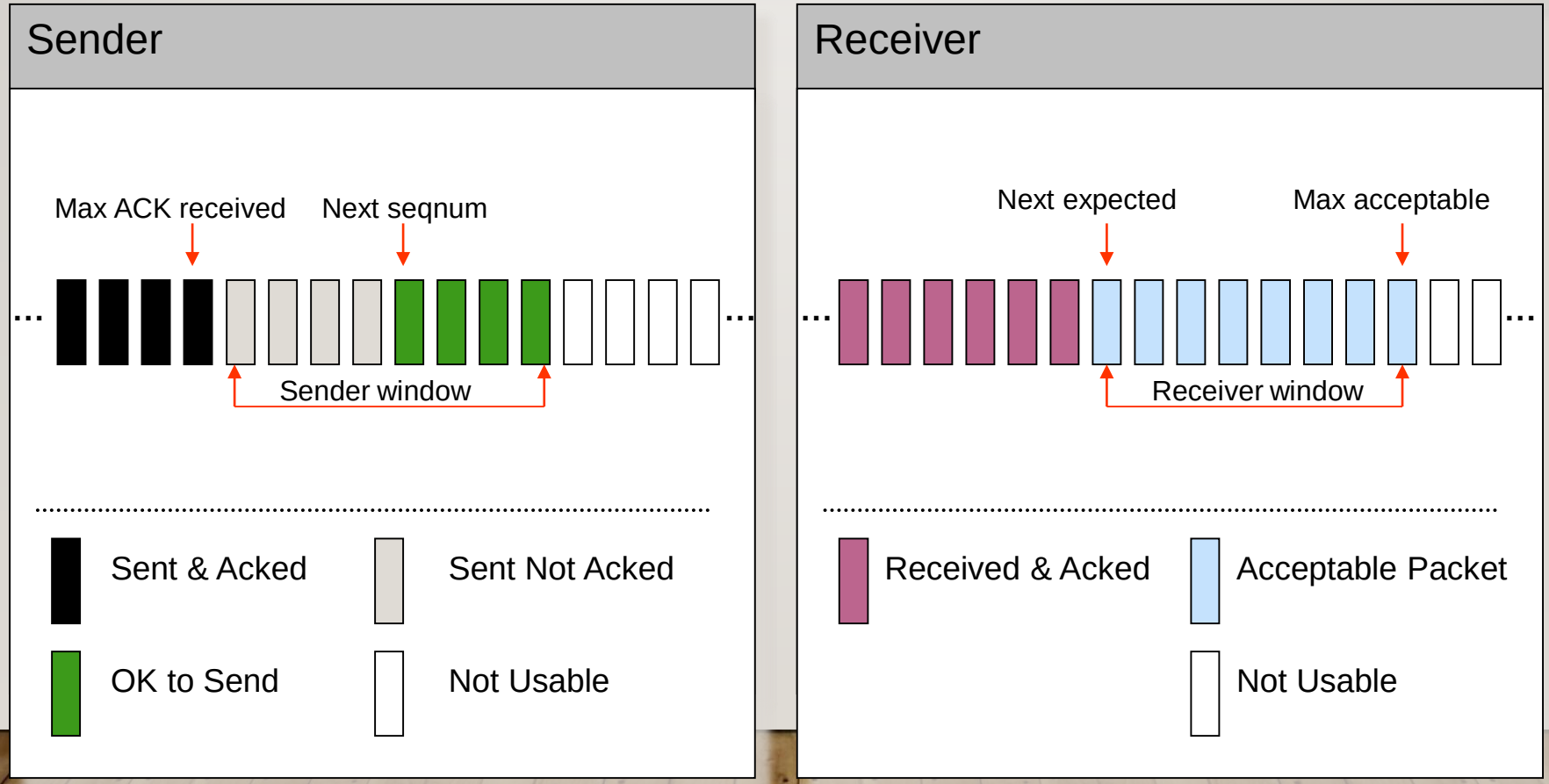
- Buffer enough data on sender to keep sending for the entire RTT.
- Treat sending buffer as circular: As ACKs come back, "slide window" to buffer new data, releasing old data and keep sending.
- If ACK doesn't come back in time, resend data.
  - Head-of-line blocking is possible
- Keep buffer an receiver in sync with sender to buffer, releasing segments up the stack in order.
- Requires segments, segment numbers

TRANSPORT LAYER: SLIDING WINDOW



LAR (last ACK received)
LFS (last frame sent)

TRANSPORT LAYER: SLIDING WINDOW, *CONT.*



Credit:
Hui Zhang

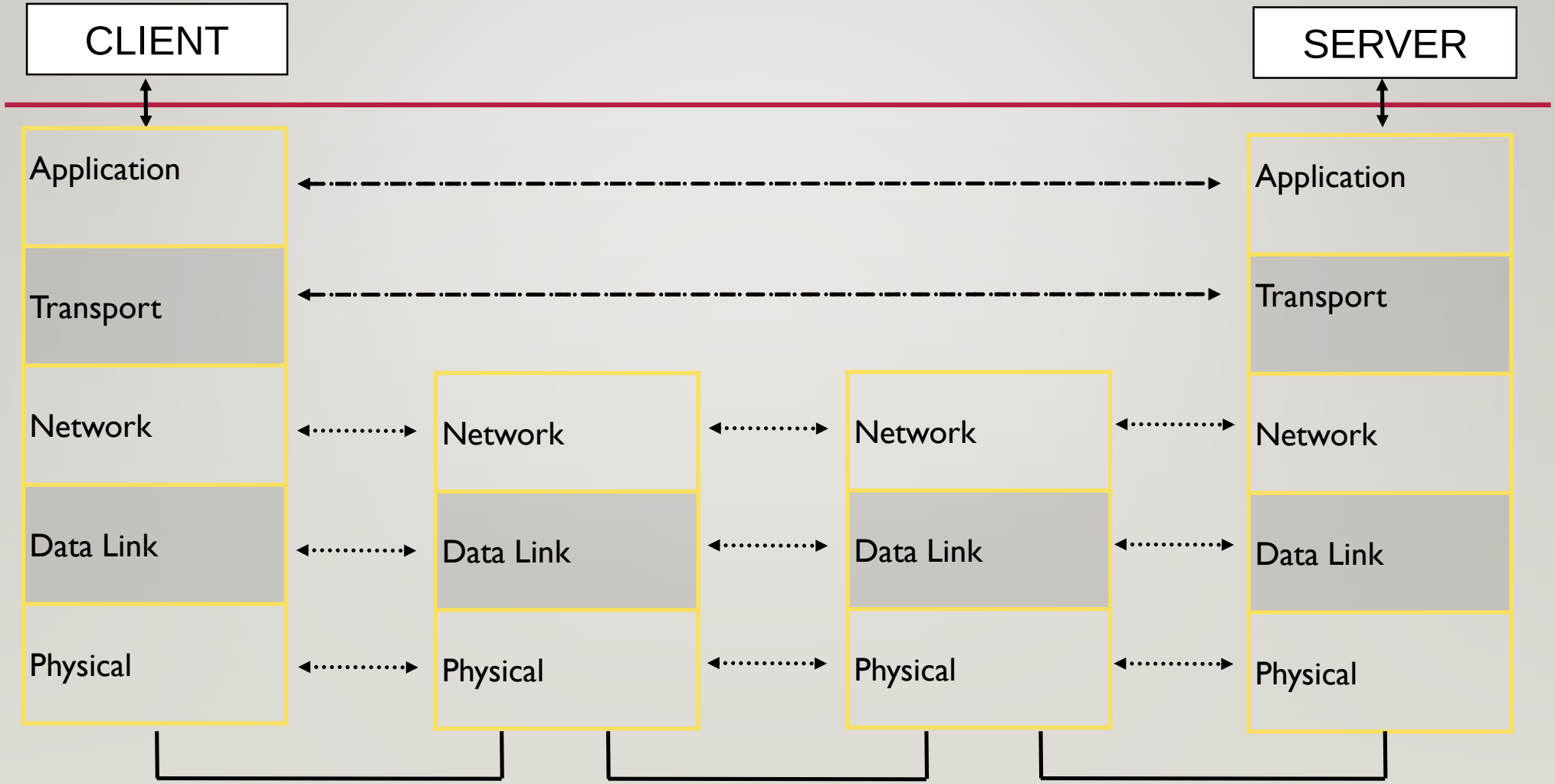
TRANSPORT LAYER: TRANSMISSION CONTROL PROTOCOL (TCP)

- Reminder: Port numbers in addition to IP addresses
- Used for transmissions that need to be correct, but not timely
  - Streaming audio or video, e.g. recorded movies
  - Bulk data transfer, e.g. uploads or downloads
- Requires overhead of establishing a session to maintain shared state between sender and receiver to coordinate.
- Different schemes for ACKs
  - Delayed ACKs, Cumulative ACKs, Selective ACKs

TRANSPORT LAYER: TRANSMISSION CONTROL PROTOCOL (TCP)

- Congestion Control
  - Packet loss can be due to many types of failure, including congestion
  - If congestion, desire is to slow down.
  - Slowing down can be achieved by shrinking window, which leaves network time unused
  - TCP has different strategies it can use to determine when to slow down and how to speed back up.
- 3-Way Handshake: SYN, SYN-ACK, ACK-SYN
  - Establishes session, negotiates window sizes and other options, e.g. SACK, window sizes, etc.

END-TO-END PROTOCOLS



END-TO-END ARGUMENT

- “Functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level.”
- Examples include, “Bit-error recovery, security using encryption, duplicate message suppression, recovery from system crashes, and delivery acknowledgment.“
- “Low-level mechanisms to support these functions are justified only as performance enhancements.“