
WELCOME
TO
15-213
14-513
18-213
18-613
15-513

Network Programming: Part I

18-213/18-613:
Introduction to Computer Systems
20^{th} Lecture,

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

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.

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

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.

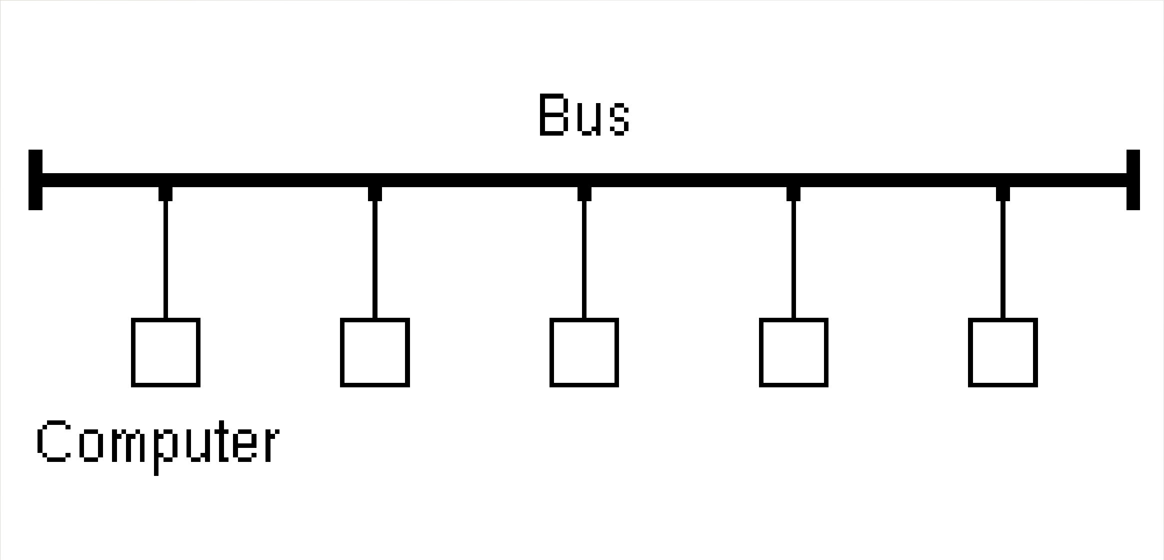
LINK LAYER:
QUICK EVOLUTION OF LANS: SIMPLE LANS

- 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

LINK LAYER: 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.

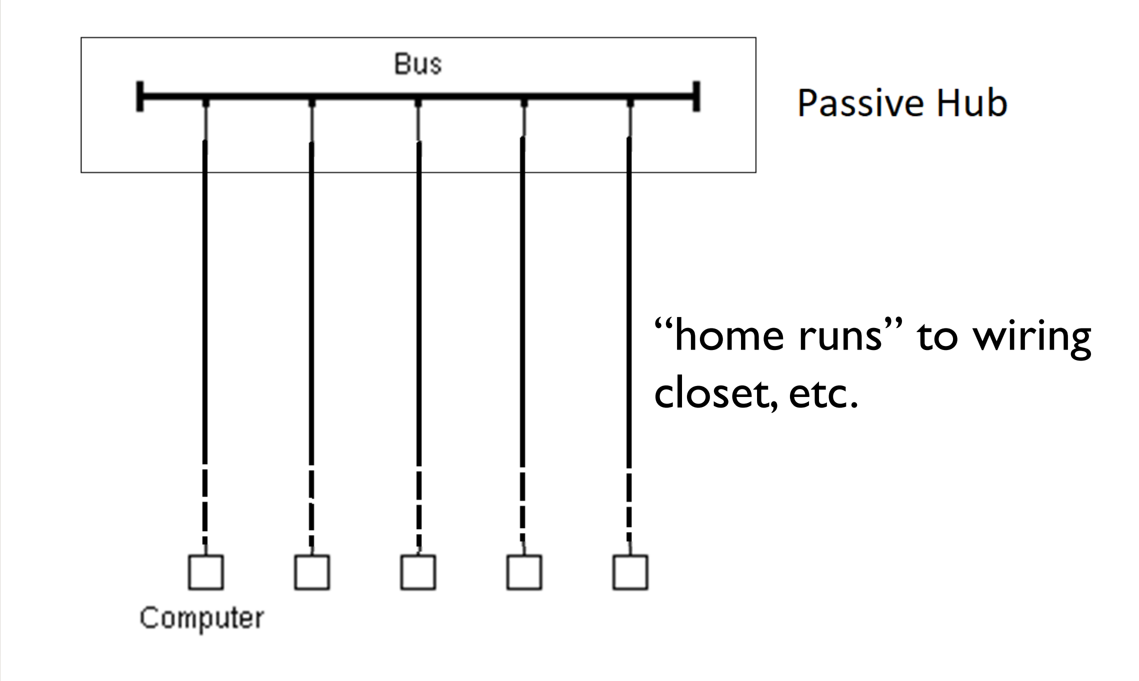
LINK LAYER:
QUICK EVOLUTION OF LANS: BUS TOPOLOGY



Imagine having to snake one wire around the building!

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

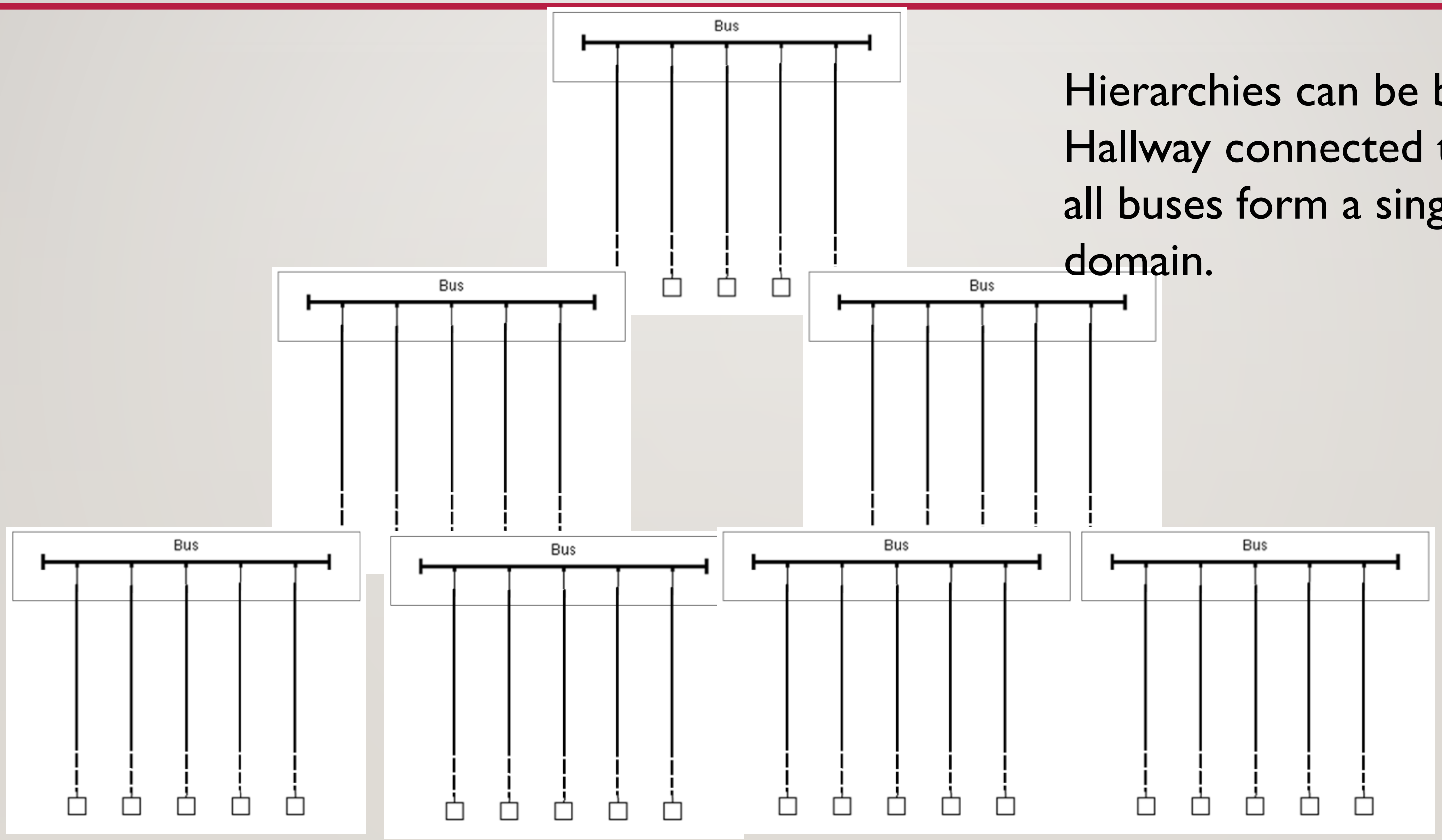
LINK LAYER:
QUICK EVOLUTION OF LANS: HUB TOPOLOGY



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

LINK LAYER:
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.

LINK LAYER:
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

LINK LAYER:
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.

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

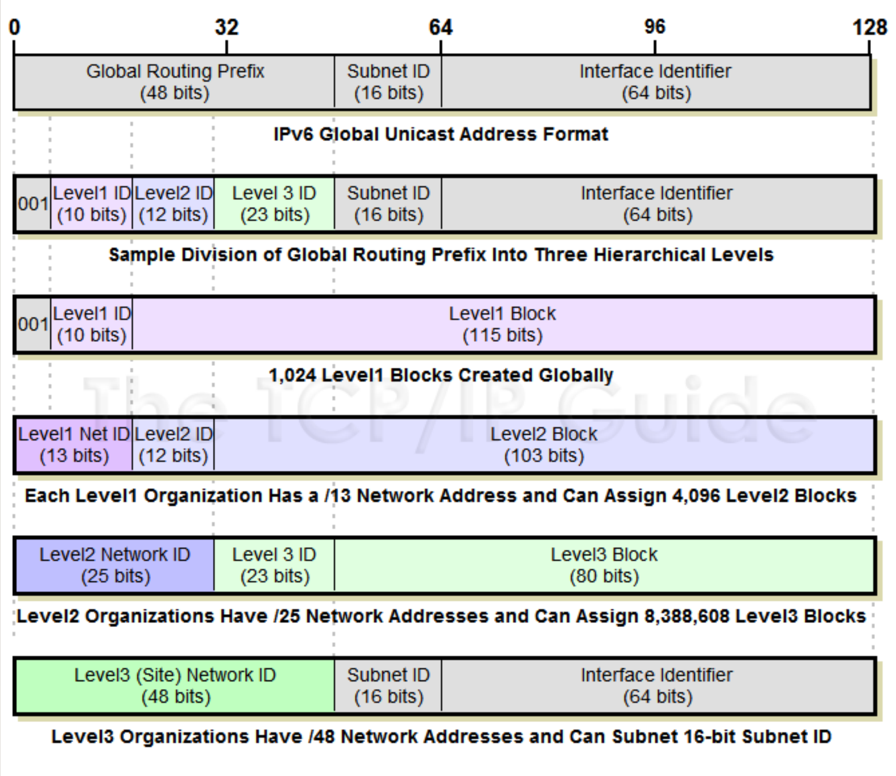
NETWORK LAYER: SCALING UP

- 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.

NETWORK LAYER: ADDRESSING

- 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

NETWORK LAYER: ADDRESSING



http://www.tcpipguide.com/free/t_IPv6GlobalUnicastAddressFormat-4.htm

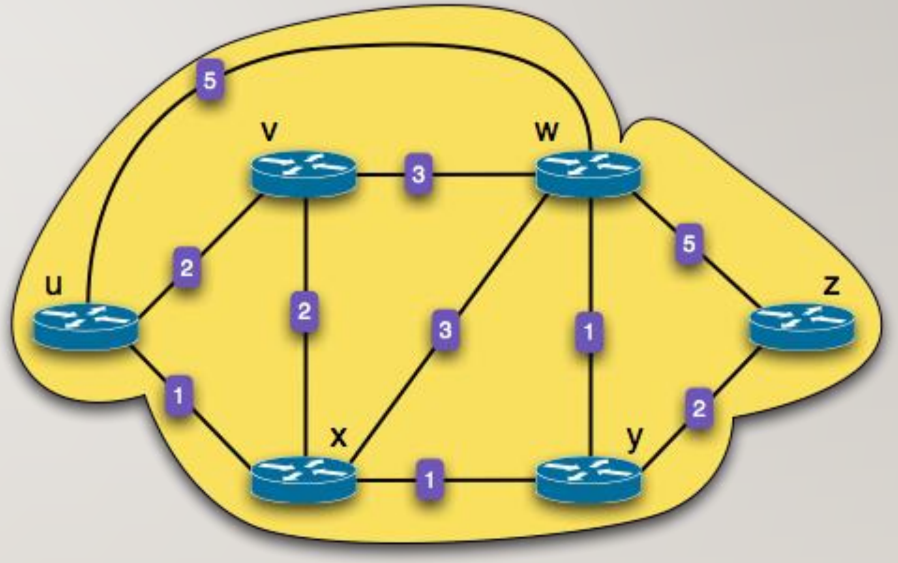
https://medium.com/hd-pro/cidr-addressing-and-subnet-masking-on-ip-networks-part-i-3054e3fbb0a1

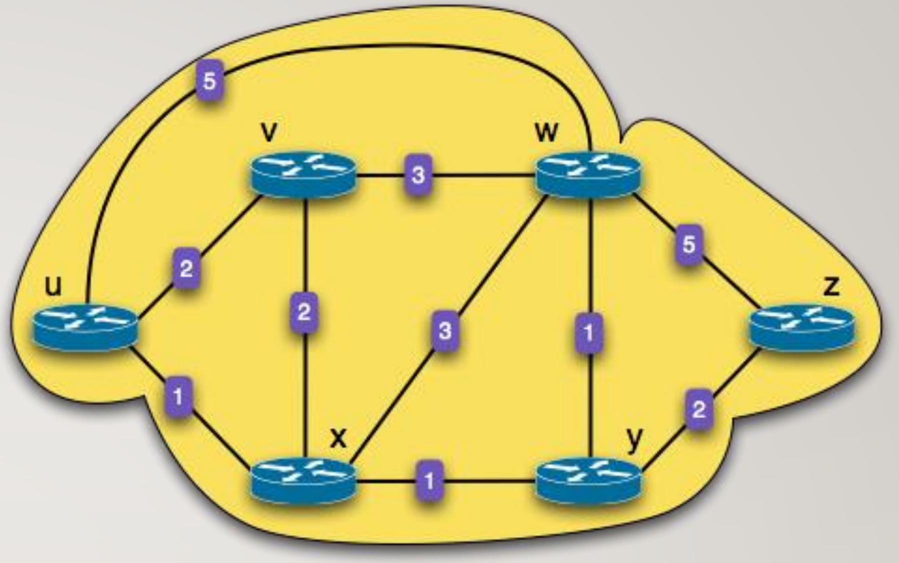
THE NETWORK LAYER:
IP ADDRESS ASSIGNMENT

- 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

THE NETWORK LAYER: ROUTING

- What is the least-cost path between u and z?
  - There are 17 different paths
- Routing Algorithm: find the least-cost path between any pairs of nodes
- When u forwards a packet bound for z :
  - Choose exit link with least-cost path





- What is the least-cost path between u and z?
  - There are 17 different paths
- Routing Algorithm: find the least-cost path between any pairs of nodes
- When u forwards a packet bound for z :
  - Choose exit link with least-cost path

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

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

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

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

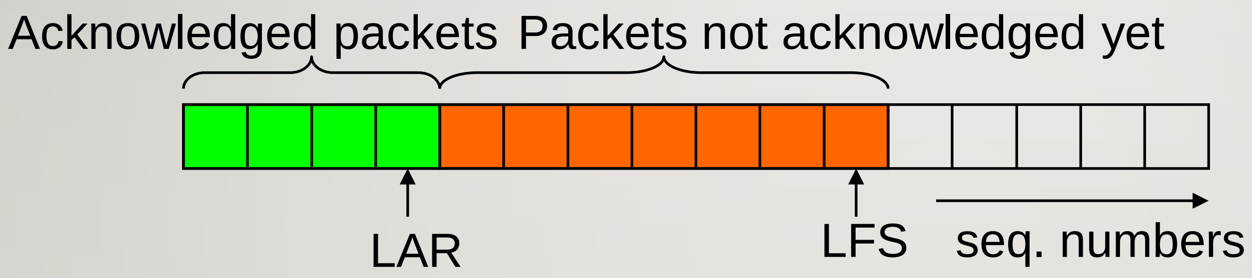
TRANSPORT LAYER: STOP-AND-WAIT PROTOCOLS

- 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

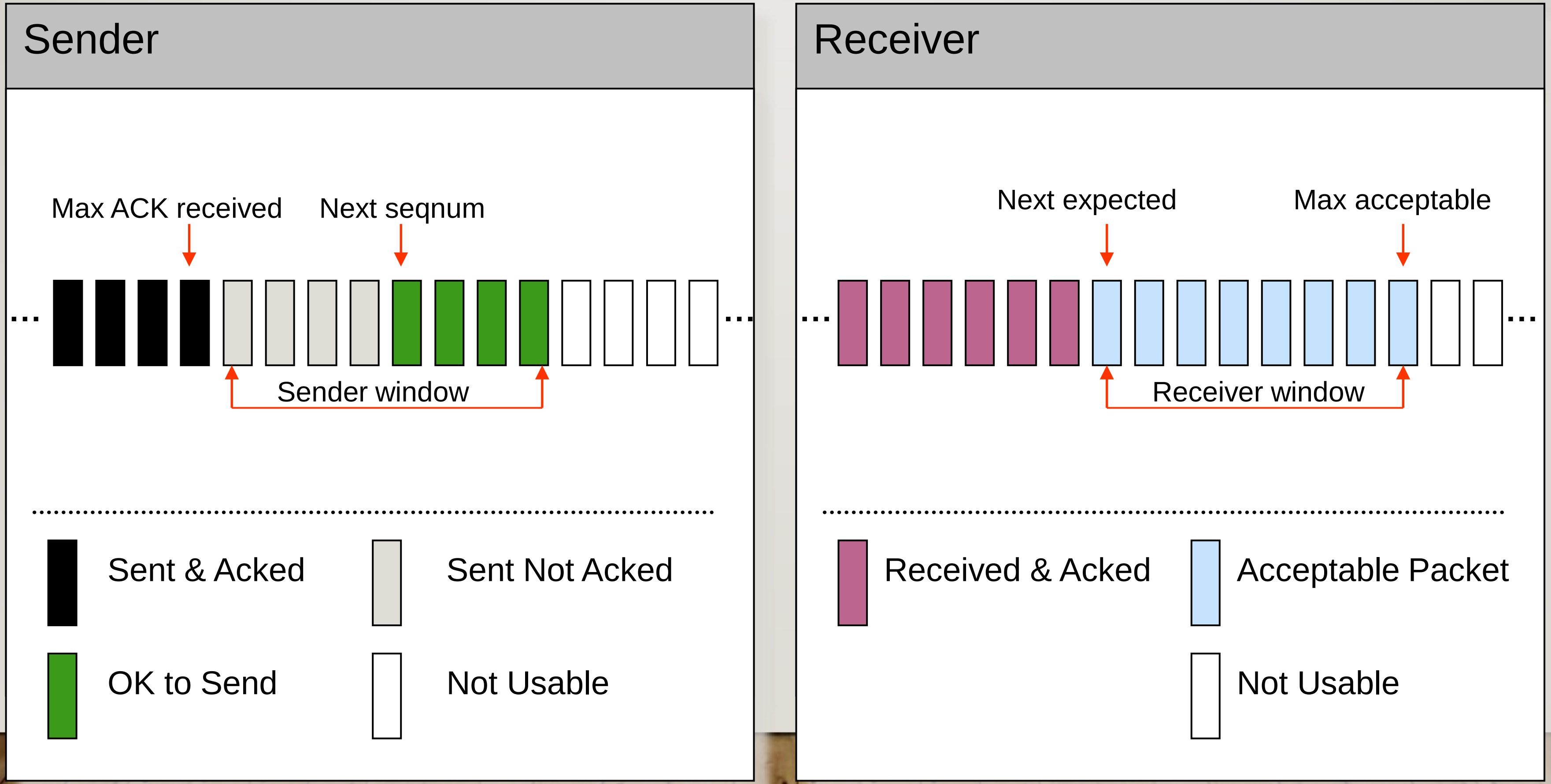
- Buffer enough data on sender to keep sending for the entire RTT.
- Treat sending buffer as circular: As ACKs come back, "slide window" to buffer new data, releasing old data and keep sending.
- If ACK doesn't come back in time, resend data.
  - Head-of-line blocking is possible
- Keep buffer an receiver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

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

APPLICATION LAYER
DOMAIN NAME SYSTEM (DNS)

- 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.
  - We'll talk about it later in detail
- Protocol specifies message format for queries and replies, etc.

PROPERTIES OF DNS MAPPINGS

- Can explore properties of DNS mappings using `nslookup`
  - (In our examples, the output is edited for brevity)

- Each host has a locally defined domain name `localhost` which always maps to the *loopback address* `127.0.0.1`

```
linux> nslookup localhost
Address: 127.0.0.1
```

- Use `hostname` to determine real domain name of local host:

```
linux> hostname
whaleshark.ics.cs.cmu.edu
```

PROPERTIES OF DNS MAPPINGS (CONT)

- Simple case: one-to-one mapping between domain name and IP address:

```
linux> nslookup whaleshark.ics.cs.cmu.edu
Address: 128.2.210.175
```

- Multiple domain names mapped to the same IP address:

```
linux> nslookup cs.mit.edu
Address: 18.62.1.6
linux> nslookup eecs.mit.edu
Address: 18.62.1.6
```

PROPERTIES OF DNS MAPPINGS (CONT)

- Multiple domain names mapped to multiple IP addresses:

```
linux> nslookup www.twitter.com
Address: 104.244.42.65
Address: 104.244.42.129
Address: 104.244.42.193
Address: 104.244.42.1

linux> nslookup twitter.com
Address: 104.244.42.129
Address: 104.244.42.65
Address: 104.244.42.193
Address: 104.244.42.1
```

- Some valid domain names don't map to any IP address:

```
linux> nslookup ics.cs.cmu.edu
(No Address given)
```