Packet loss

- Queue preceding link in buffer has finite capacity
- Packet arriving to full queue dropped (aka lost)
- Lost packet may be retransmitted by previous node, by source end system, or not at all.

"Real" Internet delays and routes

- 'traceroute' performs delay measurement from source to router along end-to-end Internet path toward destination. For all i:
  - → sends 3 packets that will reach router i on path toward destination
  - → router i will return packets to sender
  - → sender times interval between transmission and reply.

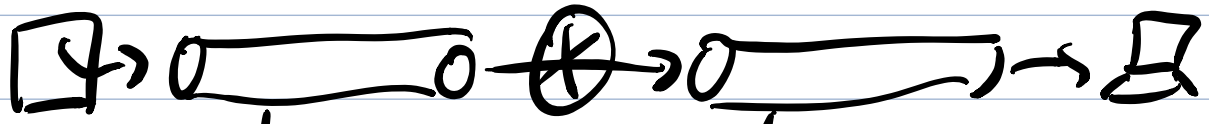
gaia.cs.umass.edu → eurecom.fr

22ms → 106ms → trans-oceanic link

Throughput

- rate at which bits transferred between sender and receiver
  - → instantaneous: rate at given point in time

→ average: rate over longer period of time



server sends bits (fluid) into pipe

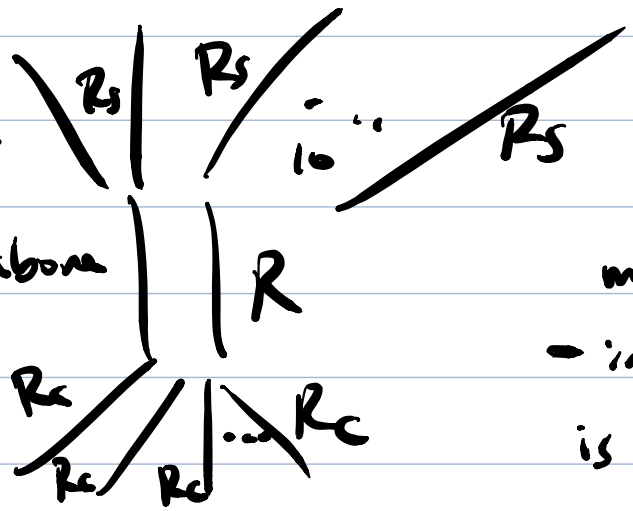
pipe that can carry fluid at rate R_s bps

pipe that can carry fluid at rate R_c bps

$R_s < R_c$ → what is the avg end-to-end throughput

$R_s > R_c$ → "

<u>bottleneck link</u>: link on e2e path that constrains e2e throughput



10 cars fairly share backbone bottleneck link R bps

- per-connection e2e throughput: $\min\{R_c, R_s, \frac{R}{10}\}$
- in practice, R_c or R_s is often bottleneck

<u>Protocol layers</u>

(eg. optical)

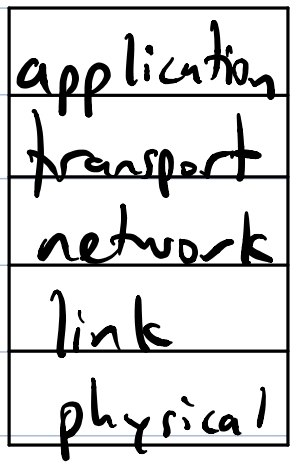
- Networks have many pieces: hosts, routers, links of various media, applications, protocols, hardware, software
- How do we organize structure of our discussion of networks?

- layers: each layer implements a service
  - → via its own internal-layer actions
  - → relying on services provided by layer below

- why layering? dealing with complex systems:
  - → explicit structure allows identification, relationship of complex system's pieces
    - → layered reference model for discussion
  - → modularization eases maintenance, updating of system
    - → change of implementation of a layer's service transparent to rest of system
    - → e.g. change in gate (airport example) procedure does not affect rest of system
  - → layering considered harmful? (duplicate functionality, e.g. error checking)

Internet protocol stack:

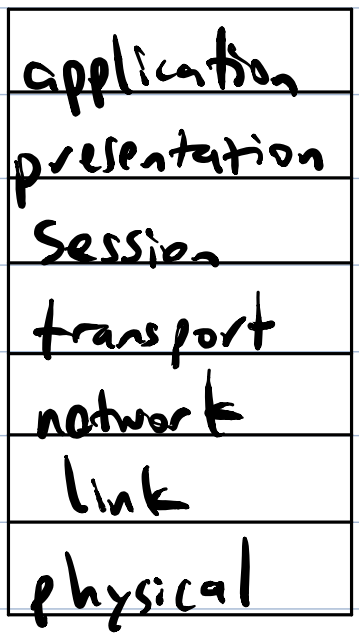
- application: supporting network applications
  - → e.g. FTP, SMTP, HTTP (sw)
- transport: process-process data transfer (sw)
  - → e.g. TCP, UDP
- network: routing of datagrams from src to dst (sw + hw)
  - → e.g. IP, routing protocols (host-to-host)



- link: data transfer between neighboring network elements (hw) (hop-to-hop)
  → e.g. Ethernet, 802.11 (WiFi), PPP
- physical: bits "on the wire" (hw)

ISO/OSI reference model

- presentation: allow applications to interpret meaning of data, e.g. encryption, compression, machine-specific conventions
- session: synchronization, checkpointing, recovery of data exchange
- Internet stack "missing" these layers!
  → if needed, must be implemented in Application
  → needed?



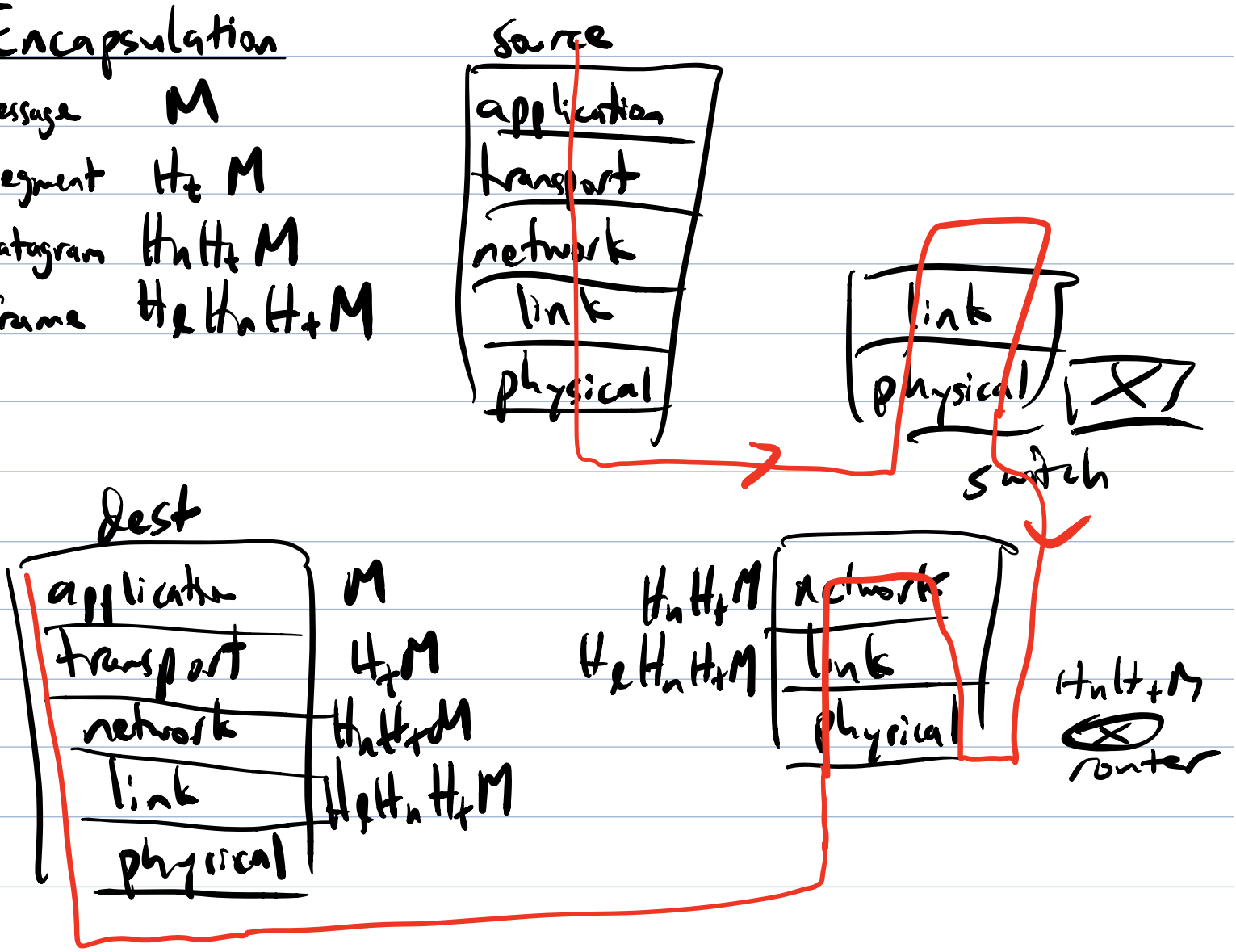
Encapsulation

message M

segment $H_t M$

datagram $H_n H_t M$

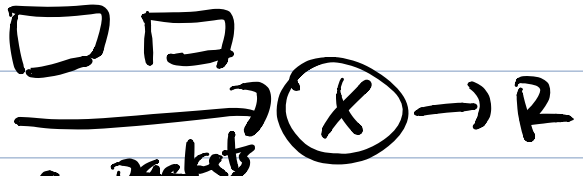
Frame $H_2 H_n H_t M$



Summary

- Covered:
  - → Internet overview
  - → What's a protocol?
  - → Network edge, core, access network
    - → packet switching vs. circuit switching
    - → Internet structure
  - → performance: loss, delay, throughput
- Now have:
  - → context, overview, "feel" of networking
  - → more depth, detail to follow

→ layering, service models



traffic utilization = traffic intensity = offered intensity

$= \frac{\text{incoming data rate}}{\text{outgoing data rate}} = \frac{a}{R}$

*=no response (probe lost/router not replying)
↳ not shown for security reasons

application	assigned a port	message
transport	process-to-process	segment
network	host-to-host	datagram
link	hop-to-hop	frame
physical	bits on wire	
layers ↑	connections ↑	packets ↑