Communication in Electronic Systems

Lecture 8: Networking and Transport Layers

Lecturer: Petar Popovski

TA: Junya Shiraishi, João H. Inacio de Souza

email: petarp@.es.aau.dk

AALBORG UNIVERSITY
DENMARK

Connectivity

Course Overview: Part 2. Communication and Networking

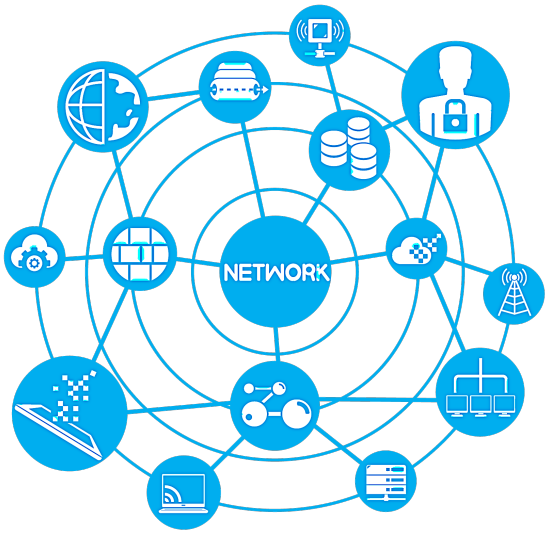
- MM5: Introduction to Communication Systems
- MM6: Simple Multiuser Systems and Layered System Design
- MM7: Network Topology and Architecture
- **MM8: Networking and Transport Layers**
- MM9: Introduction to Security
- Guest lecture
- MM10: Packets and Digital Modulation
- MM11: Communication Waveforms
- MM12: Workshop on Modulation and Link Operation

outline

- network layer
- routing
- transport layer
- TCP connection
- congestion control
- fairness

recap: lecture 3

- **lecture 3:**
  protocols and network infrastructure

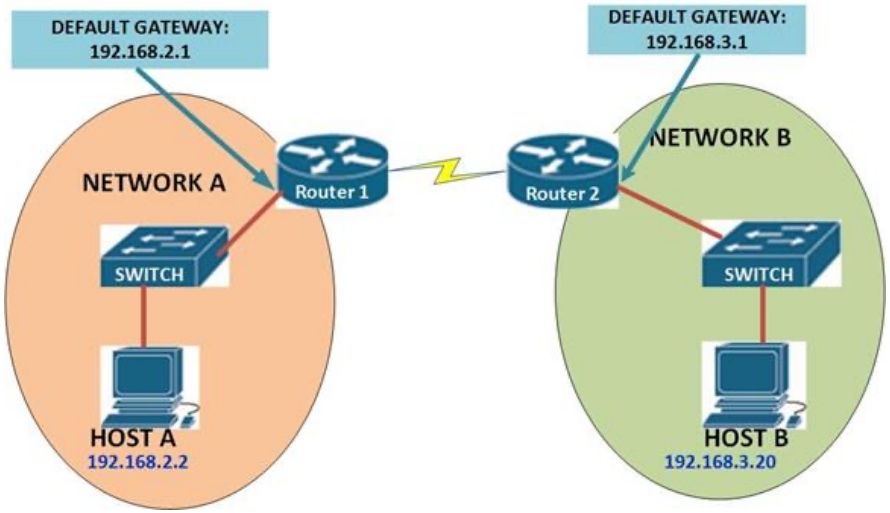


- why is layering a good architectural solution for protocols?
- how to ease the implementation of the network?
  - idea of communication modules and layering
- why do we need networking?
- network systems and topologies have been discussed

network layer

- transport segment from sending to receiving host
  - on sending side: encapsulates segments into datagrams
  - on receiving side: delivers segments to transport layer
  - principle of **store-and-forward**

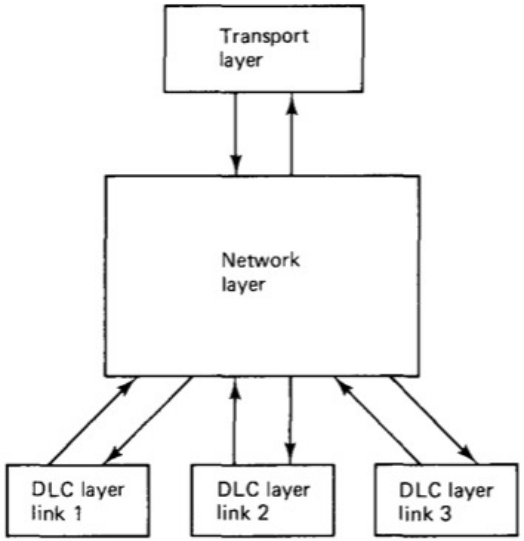
- network layer protocols in *every* host, router
- the lowest layer that deals with end-to-end transmission



source: https://www.ccnablog.com/network-layer/

functionality of the network layer (1)

- interfacing to the transport layer and the data link layer (OSI) / the host-to-network layer (TCP/IP).
- goals:
  - services independent of router technology
  - transport layer shielded from
    the number, type, and topology of routers
  - network addresses with uniform numbering/identification plan
- the most important functionality is routing:
  - make sure that the packets can find their way through the networks.
  - QoS (Quality of Service) can also be handled here,
    and so can some elements of congestion control.

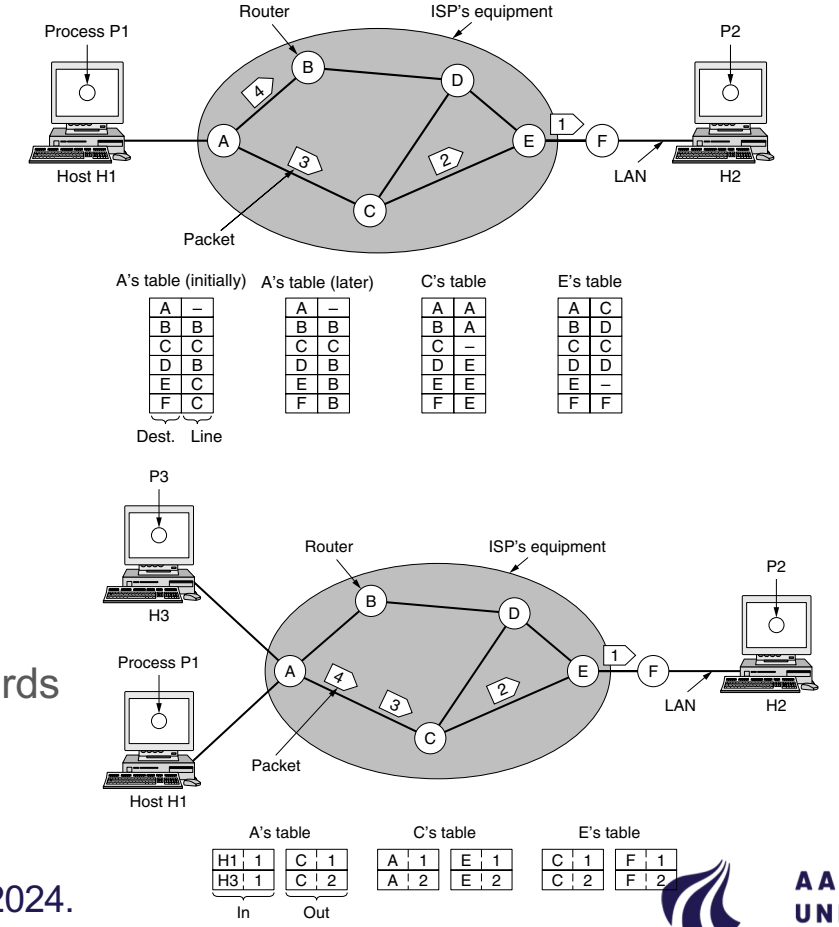


- analogy: the (snail) mail system.
  - a mail is sent by putting it in the mail box, in the required format
  - the mail system takes care that it will arrive at the receiver

connectionless vs. connection-oriented services

- connectionless
  - modelled by the postal service
  - no prior setup
  - no readiness to receive, e.g., random access
  - suitable for multicast

- connection-oriented
  - modelled by the phone service
  - initial handshake to establish
  - low control information/metadata afterwards
  - suitable for longer, frequent exchanges



functionality of the network layer (2)

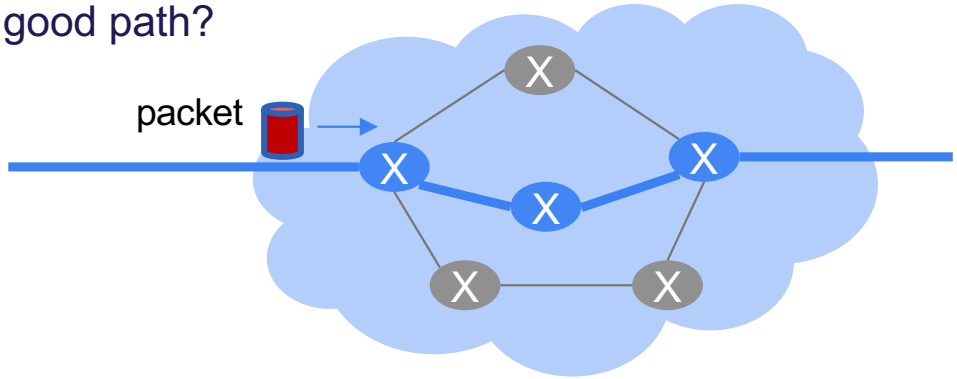
- the network layer can offer either connectionless or connection-oriented services to the transport layer:
  - ATM (**Asynchronous Transfer Mode**): Connection-oriented with virtual circuits
  - IP: Connectionless, with datagrams (store and forward)

- even if the network layer is connectionless,
  the transport layer can still be connection-oriented (e.g. TCP).

- the internet camp thinks network layer should do only routing,
  no flow control or packet ordering
  - end-to-end argument has been driving the internet development

routing (1)

- the purpose of routing protocols: to ensure that packets are sent the best possible way through a (sub)net. but what is a good path?



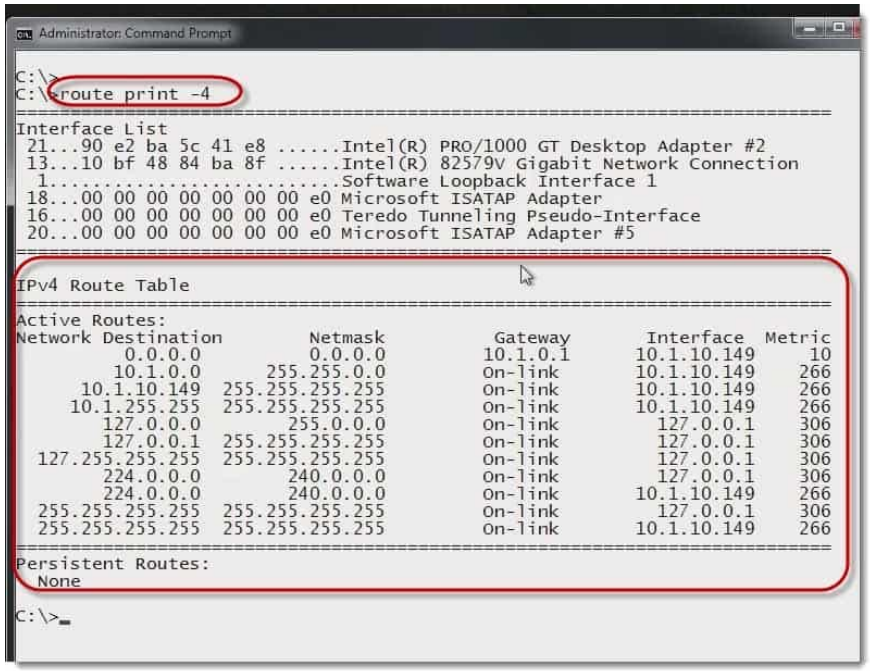
- several metrics:
  Hop count, Bandwidth, Delay, Jitter, Packet loss, Cost… Or combinations of the above…

- sometimes an advantage to use different paths through the network for the same session.
  - this can be part of a congestion control algorithm.

- in the Internet: packets are usually routed from router to router (hop by hop), where each router looks up the destination of the next.
  - different packets can travel different paths…

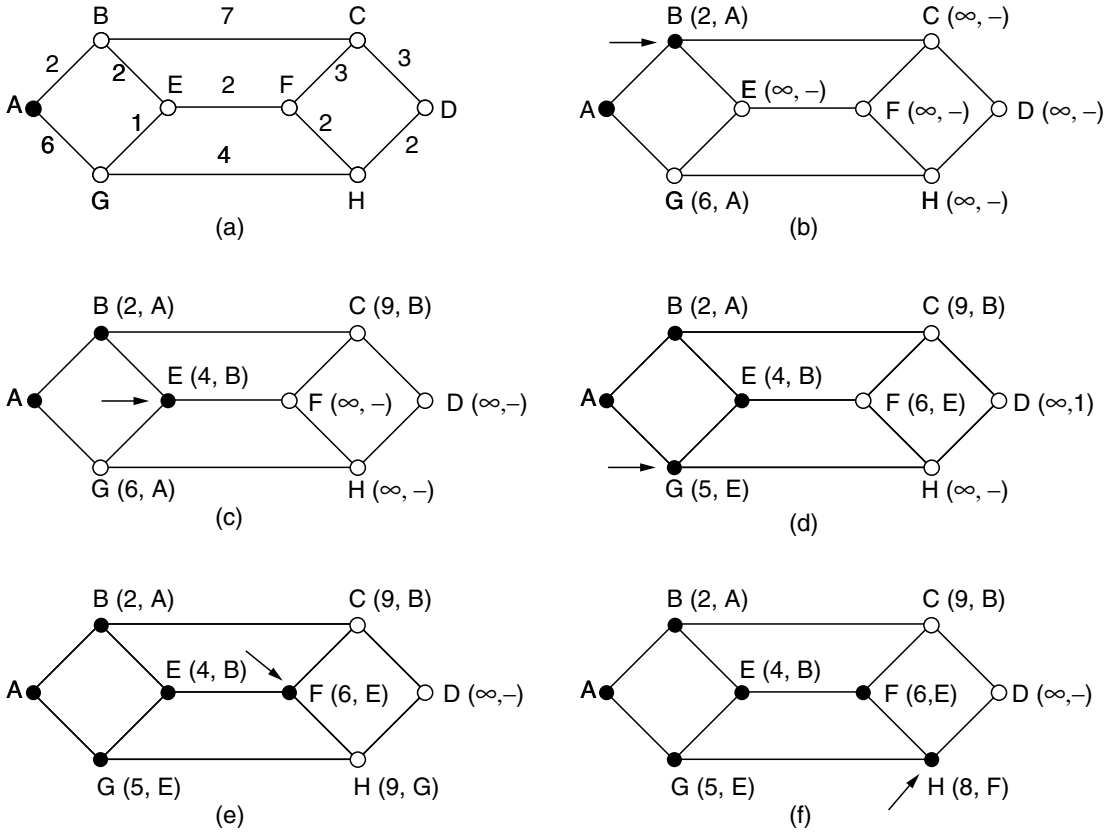
routing (2)

- handling routing tables and tables updated
- forwarding traffic according to routing tables



shortest path routing in static network

- popular Dijkstra algorithm



routing engine

- how can a router maintain an overview of a large network that can change dynamically?

- different methods and routing algorithms exist, such as:
  - flooding
  - Distance Vector Routing
  - Link State Routing

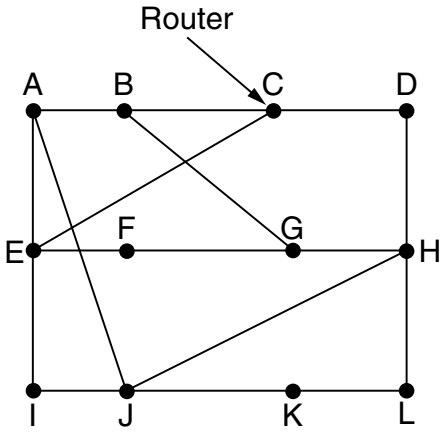
- flooding
  - replicate the incoming packet on all other links
  - many duplicates
  - introduce decreasing hop counter, ignoring of previously seen packets
  - only option when topology is unknown

distance vector routing

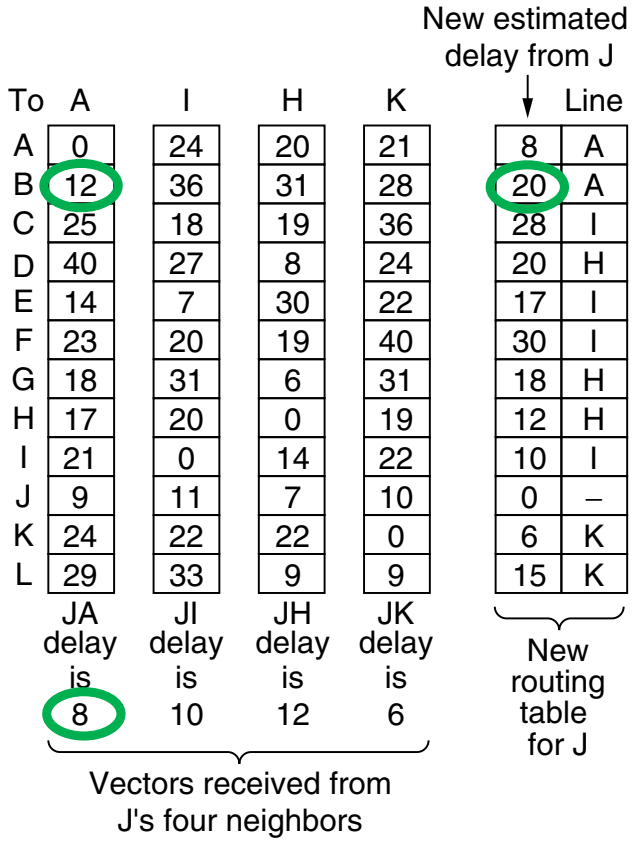
- each node maintains routing table to the other routers in the network.
- each node stores the following information:
  - distance (measured in hops)
  - direction (what is the next hop)

- this information is spread to the neighbors of each node
  triggered by events or in fixed intervals
- upon receiving this information, it can re-calculate its routing tables.

distance vector routing: example



(a)



(b)

distance vector algorithm (1)

- based on Bellman-Ford algorithm

$d_x(y)$:= cost of least-cost path from x to y

then

$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

$d_v(y)$: cost from neighbor v to destination y

$c(x,v)$: cost to neighbor v

$\min_v$: *min* taken over all neighbors v of x

distance vector algorithm (2)

- key idea
  - from time-to-time, each node sends its own distance vector estimate to neighbours
  - when a node receives an update from any neighbor,
    it updates its own DV using B-F equation
  - convergence to the actual least cost to every destination in the network

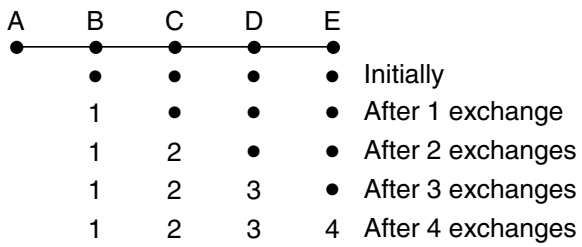
- each node:
  → wait (change in local link cost or message update)
  → recompute
  → if gotten a new DV, notify neighbours
  → repeat the process

- completely distributed process

how long it takes to propagate through all network?

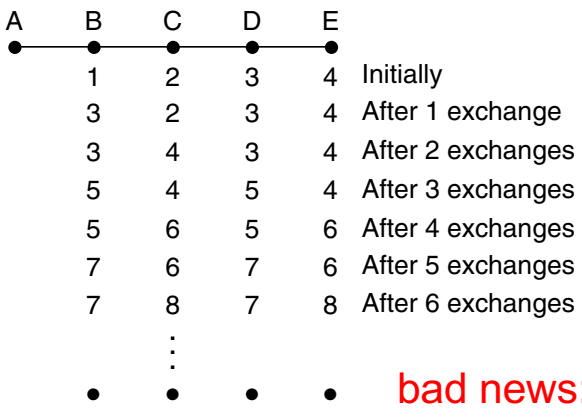
- with every algorithmic step, information is propagated one hop further
  → time needed for convergence is linearly proportional to n

- ”good news travel fast; bad news travel slow”
  - count-to-infinity problem



(a)

good news:
A comes up after a failure



(b)

bad news:
A fails after a proper operation

link cost changes

- convergence time varies (because it is iterative process)
- routing loops can occur (eventually, they will be resolved)
- security threaths: a router can advertise incorrect path cost → leading to black-holing
- reliability:

  unintentional error can propagate thru network while routing tables are updated

conclusion

- smart: very little overhead, since only communication with neighbours.
- not so smart: slow convergence, count-to-infinity problem.

link state routing

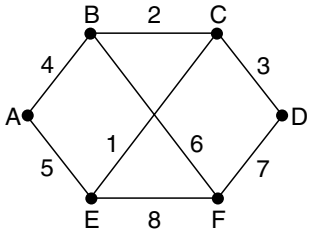
- each node maintains a map of the network, from which distances can be calculated using standard **shortest-path** algorithms (e.g. Dijkstra).

- the maps are maintained by each node **flooding** information to all other nodes about itself and the distance to its neighbours.

- variants of link state routing that are widely used on the internet
  - IS-IS (Intermediate System to Intermediate System)
  - OSPF (Open Shortest Path First)

link state routing: steps

- discover its neighbors and learn their network addresses
  - HELLO messages
- set the distance or cost metric to each of its neighbors
  - inversely proportional to bandwidth of a link
- construct a packet telling all it has just learned.
  - age: countdown for robustness
- send this packet to and receive packets from all other routers: flood
- compute the shortest path to every other router.



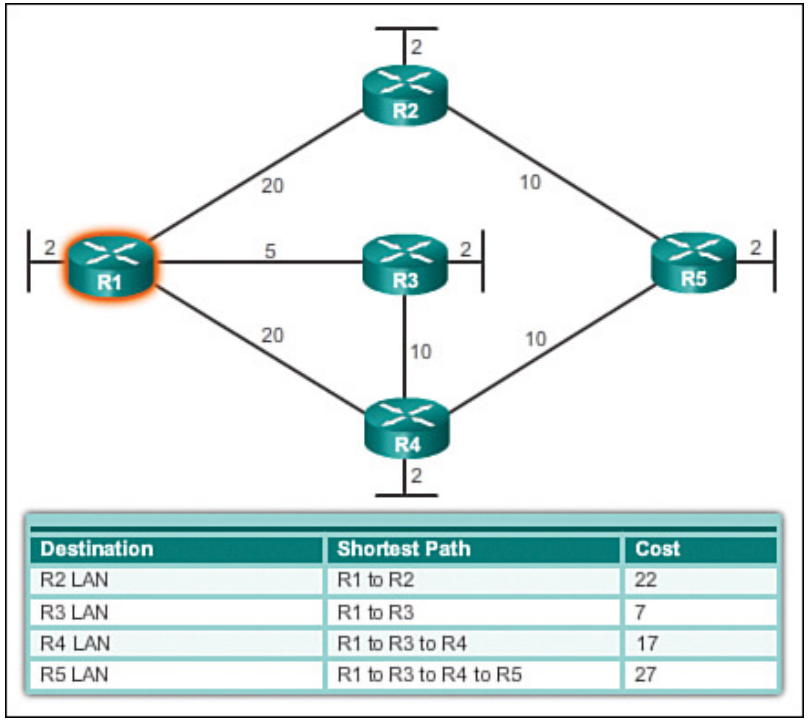
Link State Packets

A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

link state routing: features

- centralized:
  network topology and link costs are known to ALL nodes
  - Link State broadcast messages
  - all nodes will get the same info eventually

- Each node computes least cost paths from one node, itself, (source) to all other nodes → forwarding tables



Destination	Shortest Path	Cost
R2 LAN	R1 to R2	22
R3 LAN	R1 to R3	7
R4 LAN	R1 to R3 to R4	17
R5 LAN	R1 to R3 to R4 to R5	27

Source: https://www.ciscopress.com/articles/article.asp?p=2180210&seqNum=11

complexity

- assume n nodes

- computation complexity
  - n iterations; for each iteration there is a comparison with nodes that are not selected yet → *O(n^2)* complexity
  - there exists a more efficient implementation that gives complexity of *O(n logn)*

- communication complexity
  - each node broadcasts its link state information to other n nodes
  - message complexity is *O(n^2)*

challenges and a verdict

- oscilations of link costs
  - especially, if costs related to available capacity or links congestion levels are used
- scalability
  - how to design the network so that it will still scale horizontally
    in 10 or more years but without becoming unmaintainable

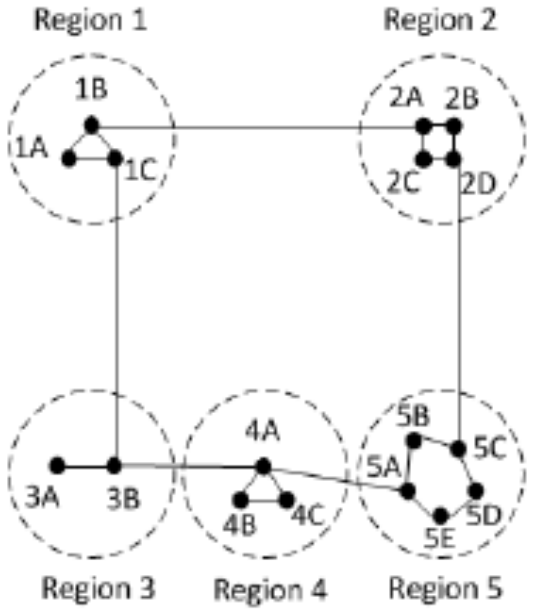
conclusions

- smart: no count-to-infinity problem
- not so smart: scalability

large networks: hierarchical routing

- large networks can be a challenge with respect to routing: table updates
  - table storage
  - look-up times

- solution: hierarchical routing



other routing notions

- unicast (described so far)
- broadcast: send to all
- multicast: send to a group
- anycast: multiple nodes can have the response, so any response from them is fine

transport layer

AALBORG UNIVERSITY
DENMARK

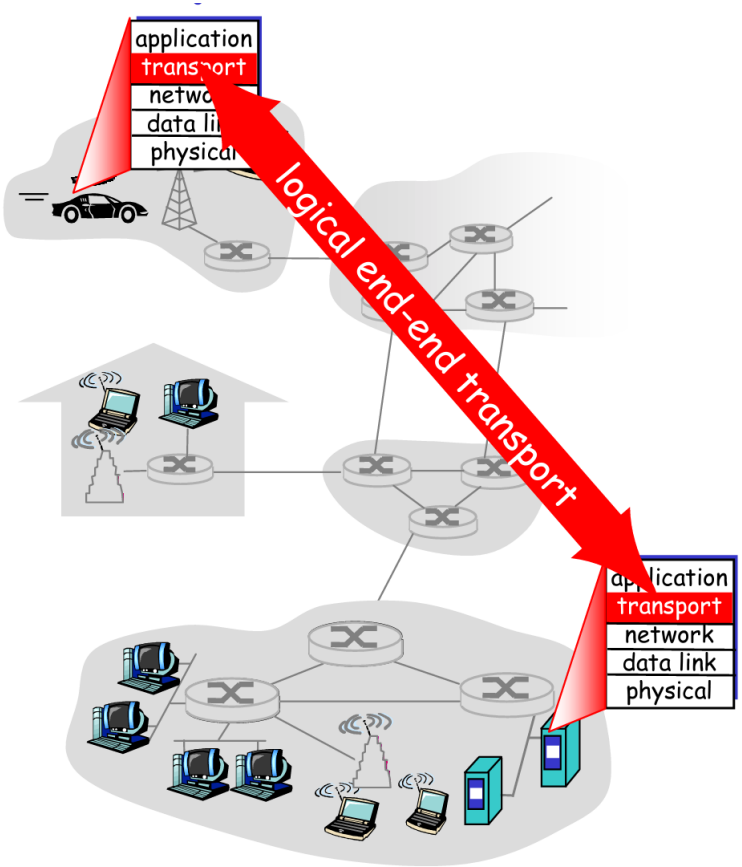
Connectivity

transport layer - functionality

- the transport layer is end-to-end addressing based on port numbers.
- it makes it possible to offer reliable and/or connection-oriented services over an unreliable and/or connectionless network.
  - connection setup, maintenance and tear-down
  - retransmitting lost packets
  - handling out-of-order packet arrivals

- most used protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

- warning: simplifications may apply, a TCP is a huge subject, with many tweaks and optimization possibilities.



source: http://www.sfu.ca/~ljilja/ENSC835/Spring09/News/Kurose_Ross/PowerPoint_Slides/Chapter3_4th_ed_June_8_2007.pdf

some transport layer features and challenges

- goal: provide reliable transport service over a network of unreliable links
- code runs in all connected devices/machines; network-layer code in routers
- there can possibly be delays from the network layer
- retransmission of packets imply that there can exist
  multiple copies of the same data.
- packet loss is difficult to detect: when is a packet lost, and when it is delayed?
- it is often impossible to detect if a packet loss is due to congestion,
  or if it is real packet loss (on e.g. a wireless connection).

TCP

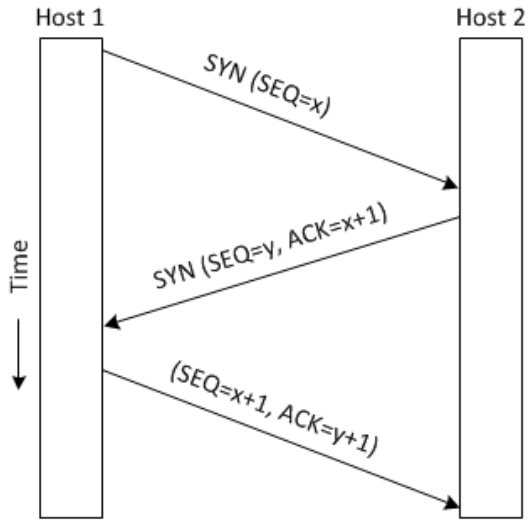
- it is defined in a number of RFCc: 793, 1122, 2018, 5681, 7323
- many versions
- byte stream, not a message stream
- TCP segment strcuture: consists of a *segment header* and a *data section*

TCP segment header

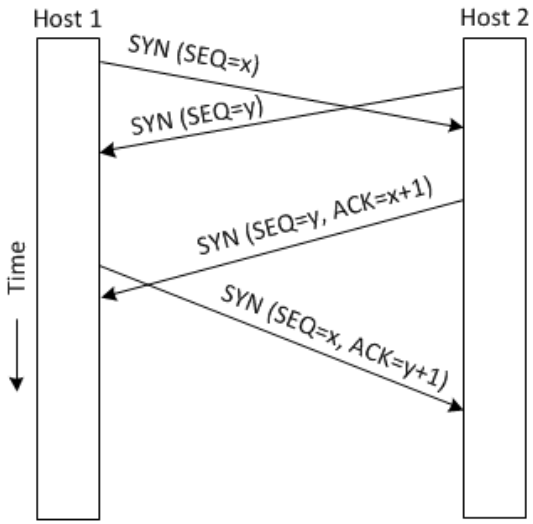
Offsets	Octet	0								1								2								3							
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port																
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																	Urgent pointer (if URG set)														
20	160	Options (if *data offset* > 5. Padded at the end with "0" bits if necessary.)																															
⋮	⋮																																
60	480																																

TCP connection set-up

- sequence numbers in one direction corresponds to acknowledgement numbers in the other.
  - a) one connection.
  - b) one connection in each direction



a)



b)

data transmission

how to achieve a reliable data transfer over an unreliable network?

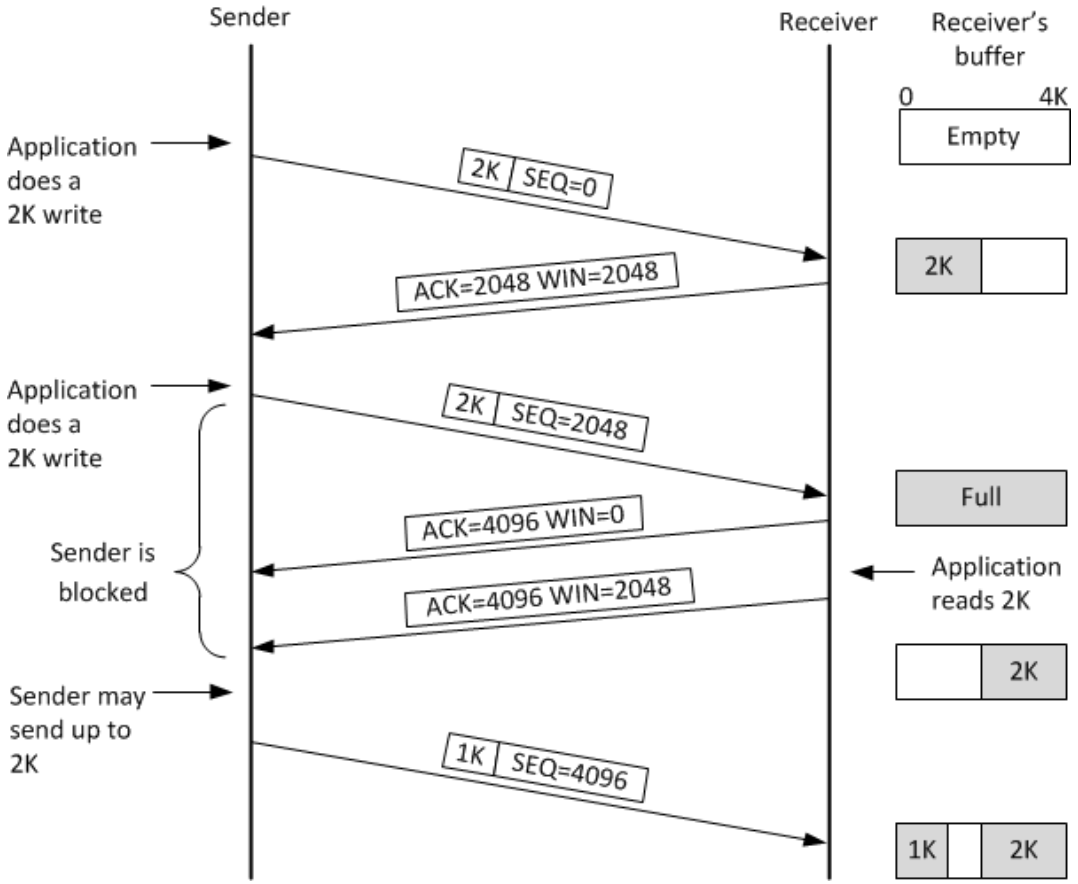
- whenever data is sent, it is also acknowledged.
- in principle we could acknowledge every packet,

  but in TCP delayed acknowledgements can be used to reduce overhead
  - we wait a little before sending the acknowledgement,

    and if another packet arrives we can acknowledge them both in the same packet
  - if a packet does not arrive, an acknowledgement is not sent, and it will be retransmitted.

TCP acknowledgements

- sliding windows
- ACK number tells
  which SEQ number to transmit next.
- if packets 1,2,3 are sent, but e.g. 2 is lost,
  the acknowledgement sent when packet 3
  is received will be a duplicate ACK (and
  just the data from packet 1 will be
  acknowledged).
- once packet 2 is received,
  data can be acknowledged incl. packet 3.

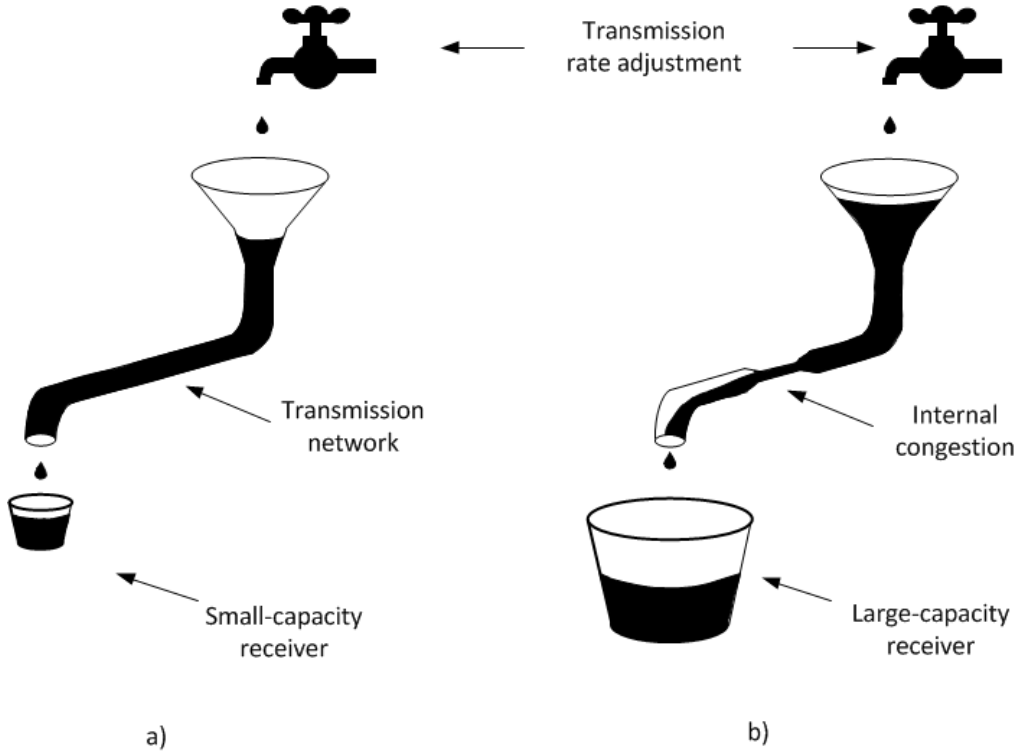


flow control

- there are two potential bottlenecks:
  - the network (or part of the network)
  - the receiver

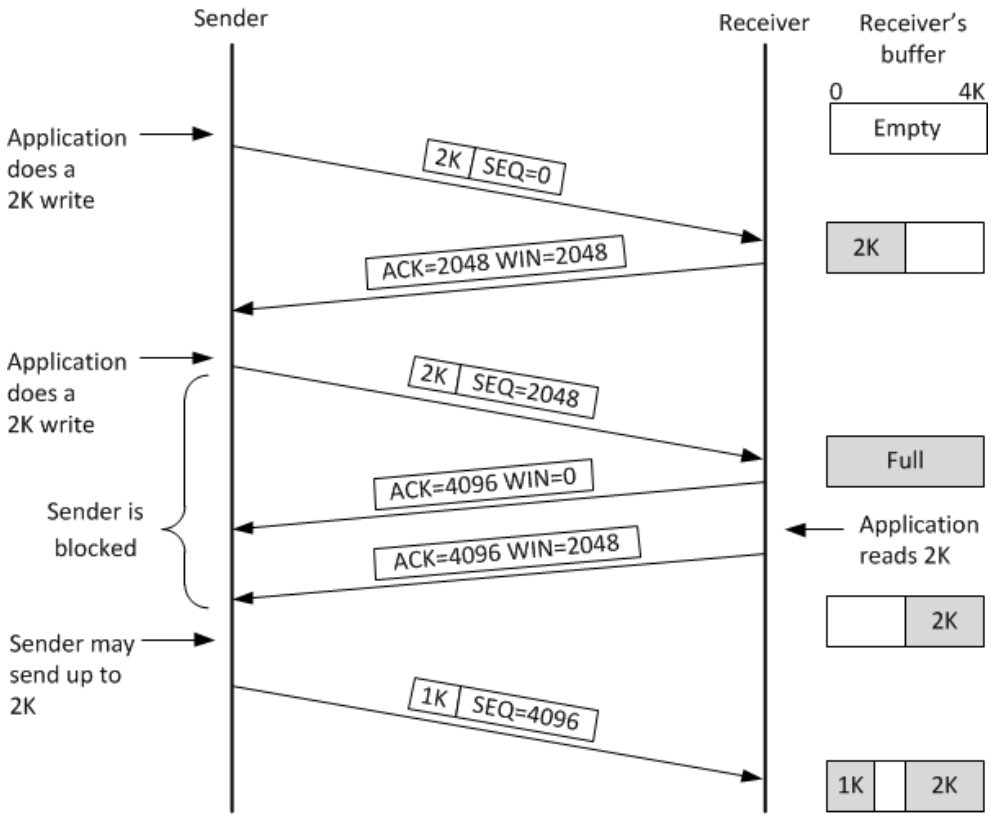
flow control:
preventing from overloading the receiver
congestion control:
preventing from overloading the network

Flow control
sender should not overflow receiver's buffer by transmitting too much, too fast



sliding windows: adjust to the receivers capacity

- the sender constantly maintains a picture of the receiver's window.
- it only sends data if the window is "open". ss soon as it is filled, the sender is blocked until an update is received.



congestion control and fairness

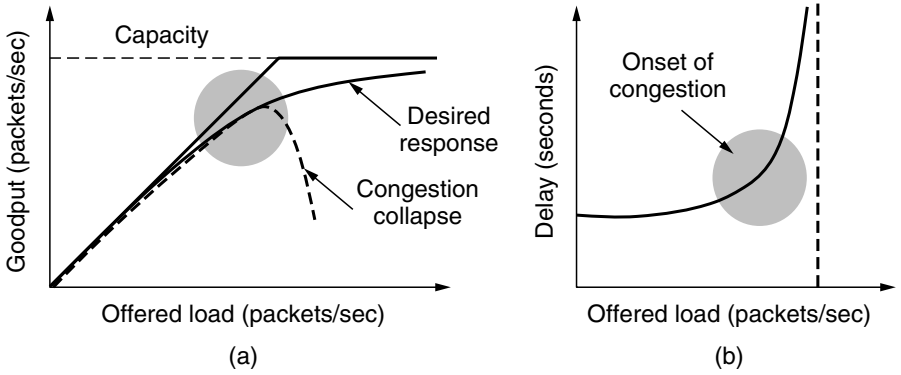
AALBORG UNIVERSITY
DENMARK

Connectivity

congestion control

- what causes congestion?
  - informally: “**too many sources** sending **too much** data **too fast** for network to handle”

- insights into congestion:
  - throughput can never exceed capacity
  - delay increases as capacity approached
  - loss/ retransmissions decreases the effective goodput
  - duplicates further decrease goodput



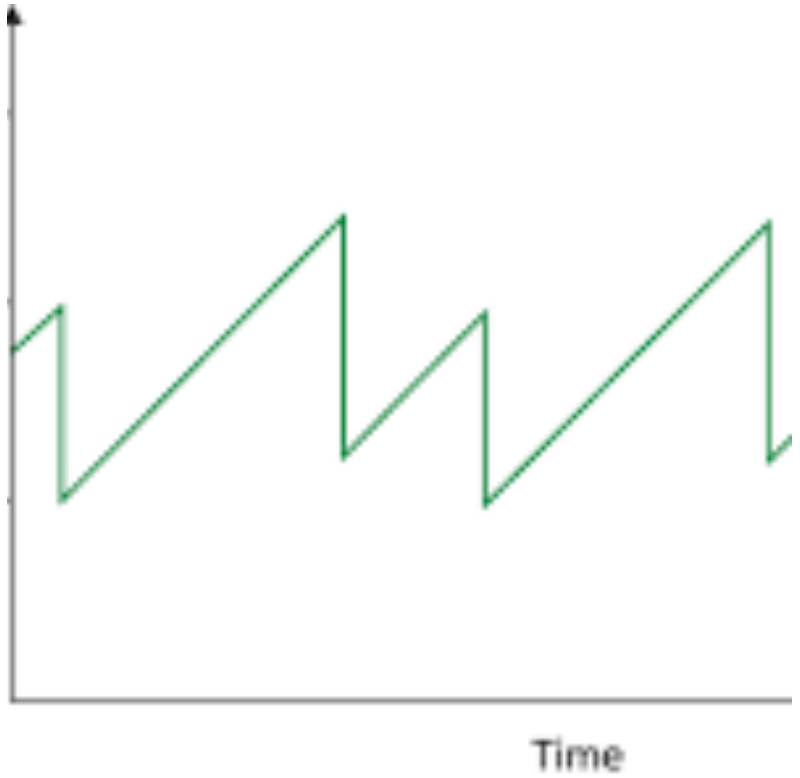
approaches towards congestion control

- end-to-end congestion control
  - **congestion window:** the number of bytes the sender may have in the network at any time
  - no explicit feedback from network (congestion “happens” on layer 3)
  - congestion is inferred from observed loss (and/or delay)
  - approach taken by TCP

- alternative approach = network assisted congestion control:
  - routers provide direct feedback
  - examples: TCP ECN (Explicit Congestion Notification), ATM

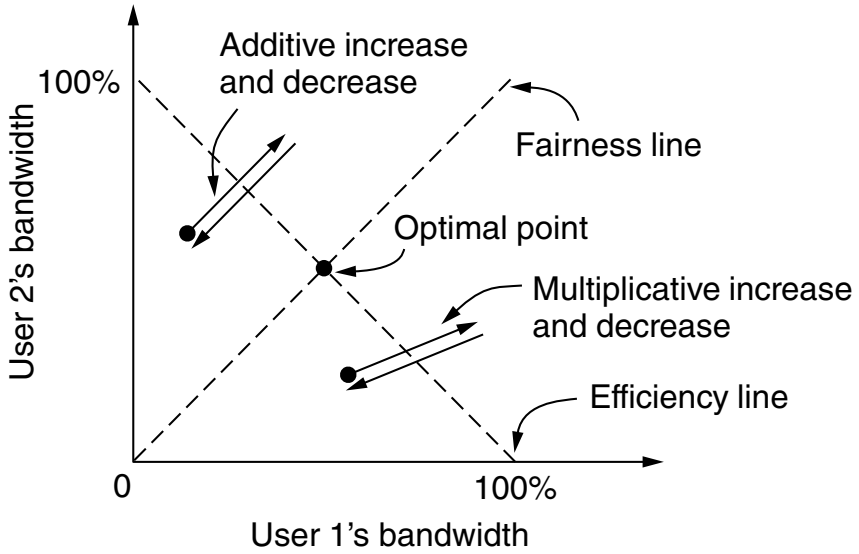
TCP congestion control: AIMD

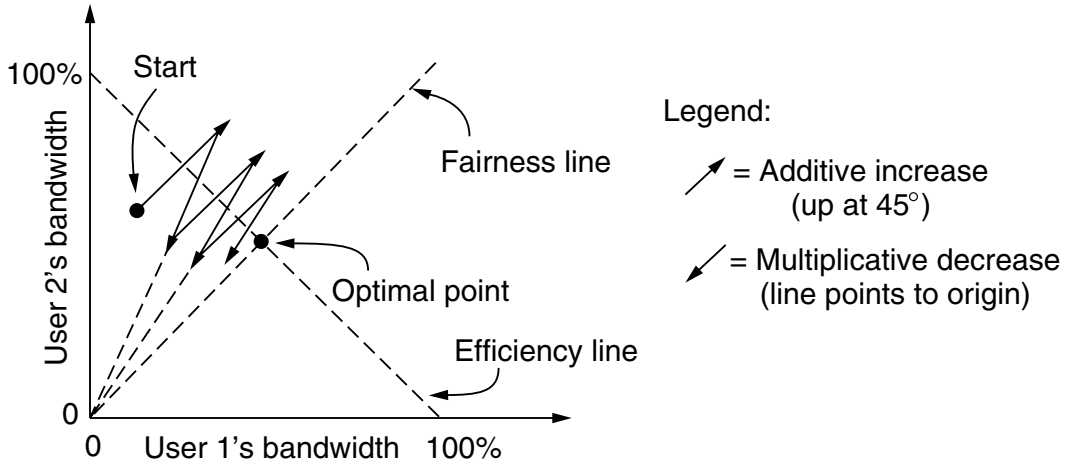
- AIMD=
  additive increase, multiplicative decrease
- approach: senders can increase sending rate until packet loss (congestion) occurs, then decrease sending rate on loss event
- why AIMD?
  - distributed, asynchronous algorithm
  - optimize congested flow rates network wide
  - have desired stability properties



why is AIMD fair

- example: two competing data sessions



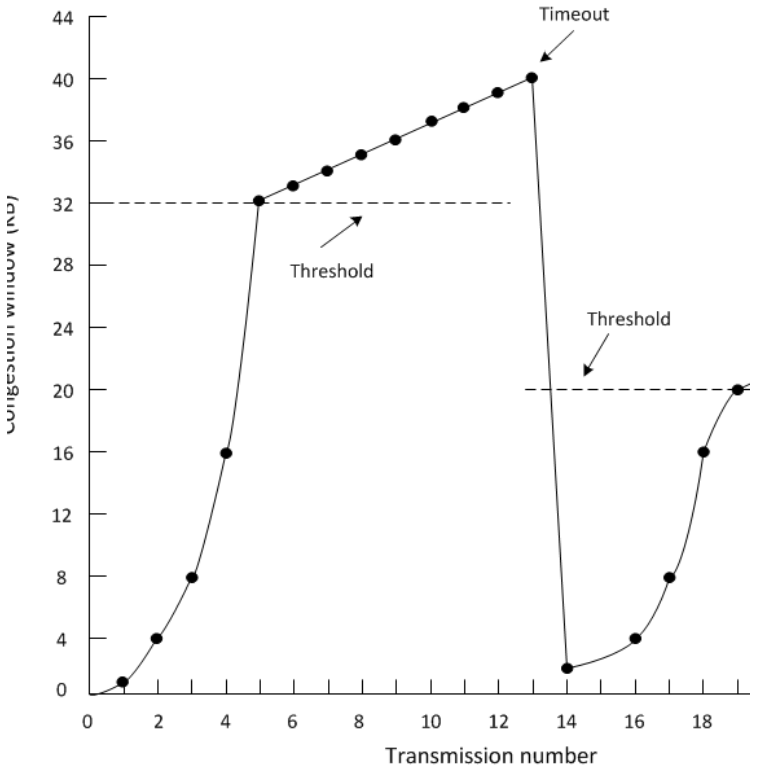


TCP congestion control: AIMD (contd.)

- how much can be sent is regulated by the size of congestion window (cwnd)

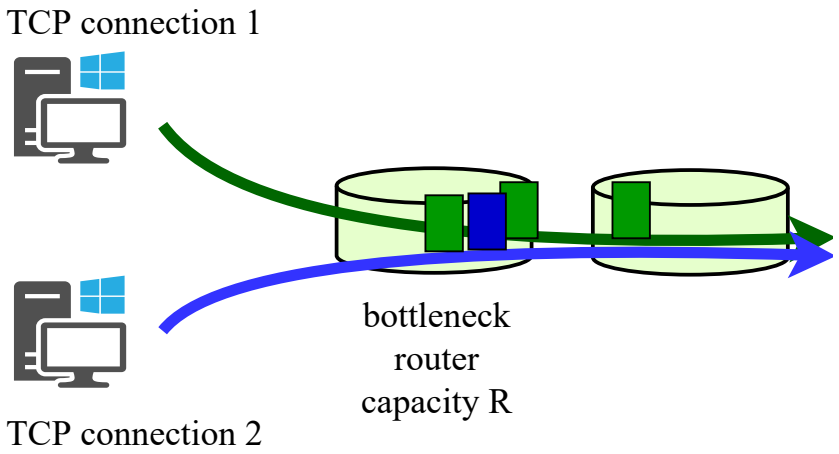
- slow start
  - exponential increase of congestion window size
  - doubling the amount of packets that we are sending

- multiplicative decrease
  - cut to 1 MSS (maximum segment size) when loss is detected by timeout (TCP Tahoe)
  - cut in half on loss detected by triple duplicate ACK (TCP Reno)



TCP fairness

- fairness goal:
  if *K* TCP sessions share same bottleneck link of bandwidth R,
  each should have average rate of *R/K*
- *max-min fairness:*
  bandwidth of one flow cannot be increased without decreasing the one of another flow



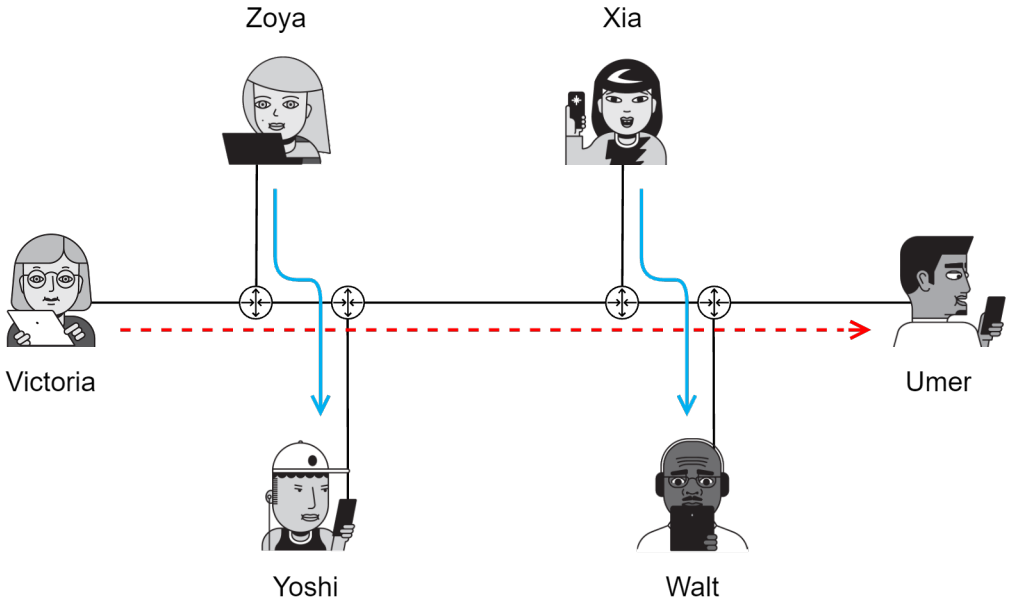
fairness

- multimedia apps often do not use TCP
- there is no “Internet police” policing use of congestion control
- no one prevents to use own unfair transport protocol,

  that will not “back off” in case of congestions
- parallel TCP connections
  - application can open multiple parallel connections between two hosts
  - they will be treated as separate TCP flows

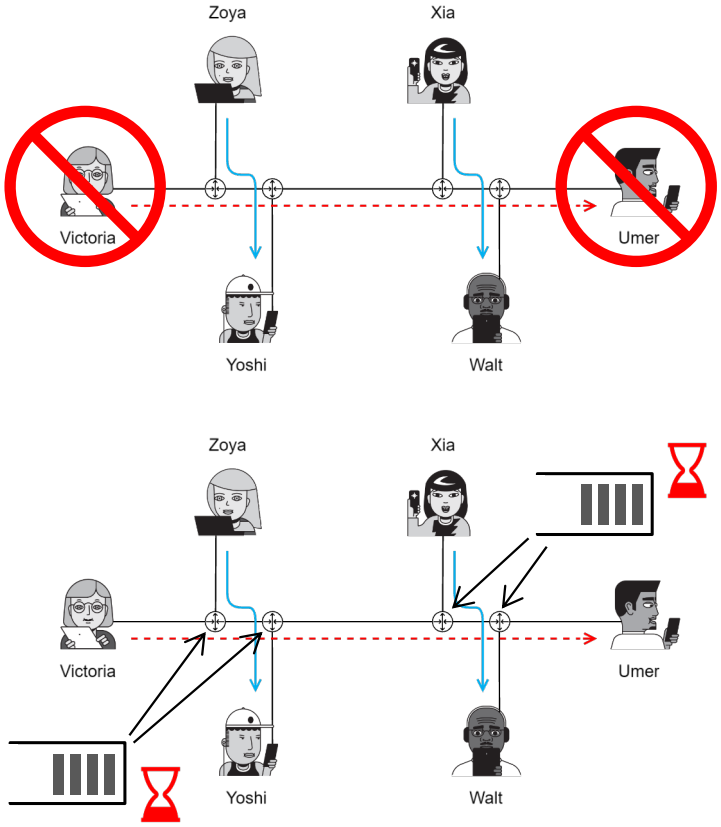
a conflict between efficiency and fairness (1)

- 3 data flows
- Zoya-Yoshi and Xia-Walt flows
  - high-resolution video call
  - saturate the horizontal links
- Victoria-Umer flow
  - audio call
  - lower data volume
- how to optimize the network?



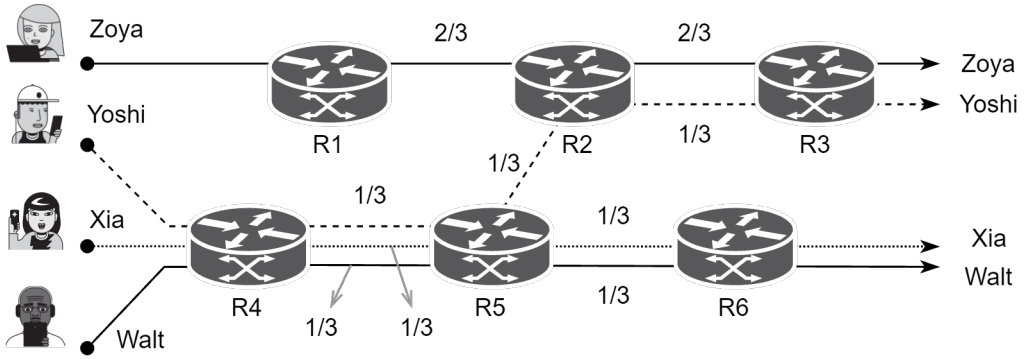
a conflict between efficiency and fairness (2)

- global efficiency
  - shut down Victoria-Ulmer flow
  - unfair network for Victoria and Ulmer
- balancing efficiency and fairness
  - routers queue and forward data packets
  - increase packet delay for Zoya and Xia
  - fair network for Victoria and Ulmer
- trade-off control
  - queueing and forwarding policies
  - adaptive bitrate streaming



max-min fairness

- 4 data flows competing for bandwidth
- each link has a capacity equal to 1



- max-min fair bandwidth allocation
  - R4-R5: all flows get 1/3 of the bandwidth
  - R1-R2: Zoya's flow get 2/3 of the bandwidth

- why is the allocation max-min fair?
  - Yoshi's get 1/2 of the bandwidth at R2-R3
  - this increases Yoshi's bandwidth at R4-R5
  - to implement this change, we need to decrease Xia's and Walt's bandwidth

summary and outlook

- networking layer and two main routing concepts
  - Distance Vector Routing
  - Link State Routing

- transport layer
  - basics of how TCP works and why it was designed in this way
  - connection and flow control
  - congestion control and fairness