

UiO : Faculty of Mathematics and Natural Sciences
University of Oslo

Department of Informatics
Networks and Distributed Systems (ND) group

IN 3230 / IN 4230

Networking principles



Michael Welzl

University education

"*When I took Electrical Engineering, I didn't learn how Sony builds amplifiers, I learned Maxwell.*"
- John Day, Boston University

- Engineering training (job preparation):
  ok to focus on **what** and **how**
  - You're going to use it, you must know how it works

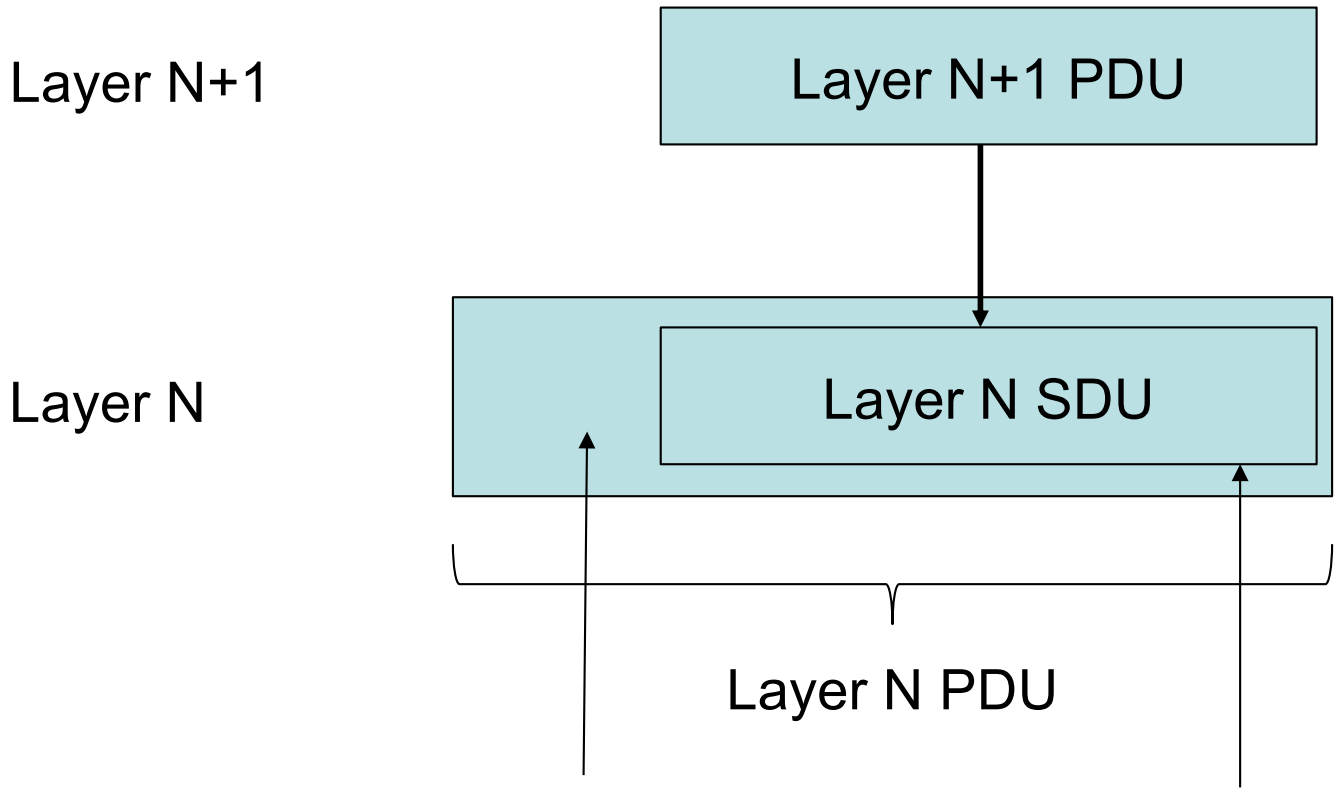
- University education: should explain **why**
  - You're going to design future systems, you need to understand why things are how they are, and what is good or bad about them

Next: communication principles

- Several slides taken/adapted from John Day
- For more details:
  John Day, “Patterns in Network Architecture: A Return to Fundamentals”. Prentice Hall, 2007.
  - First and foremost, a different way to think about layers
- We start very simple and gradually build it

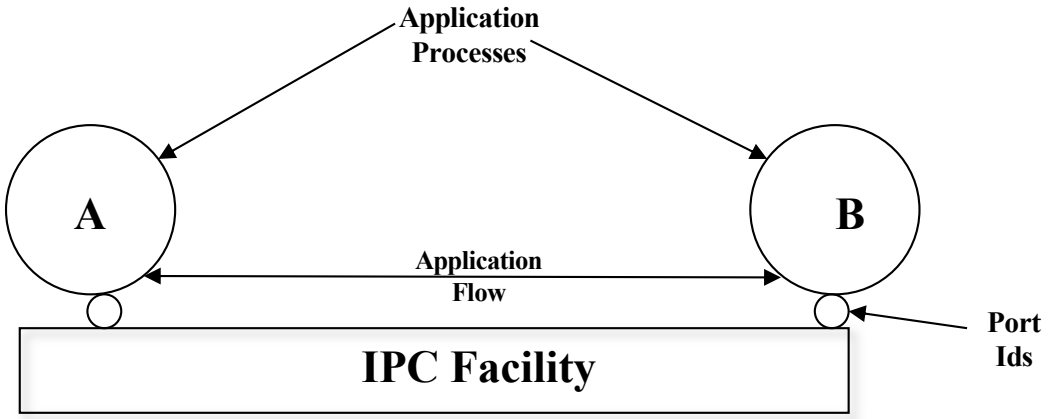
…but first: an OSI terminology reminder

PDU: Protocol Data Unit, **SDU**: Service Data Unit



Common non-OSI terminology: **header** **payload**

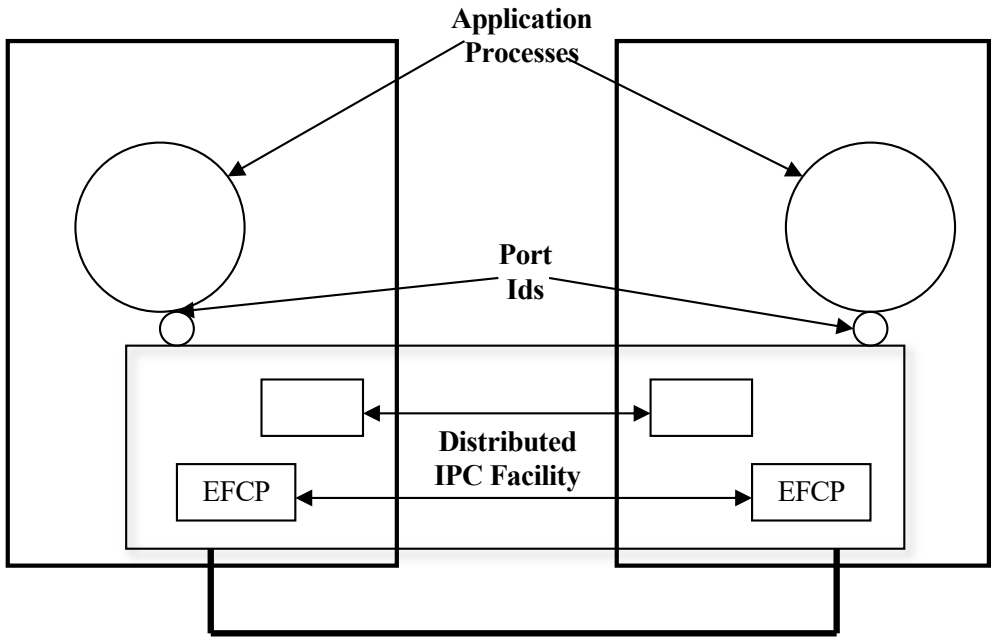
Two application processes on one system



Inter Process Communication (IPC)
within a single system

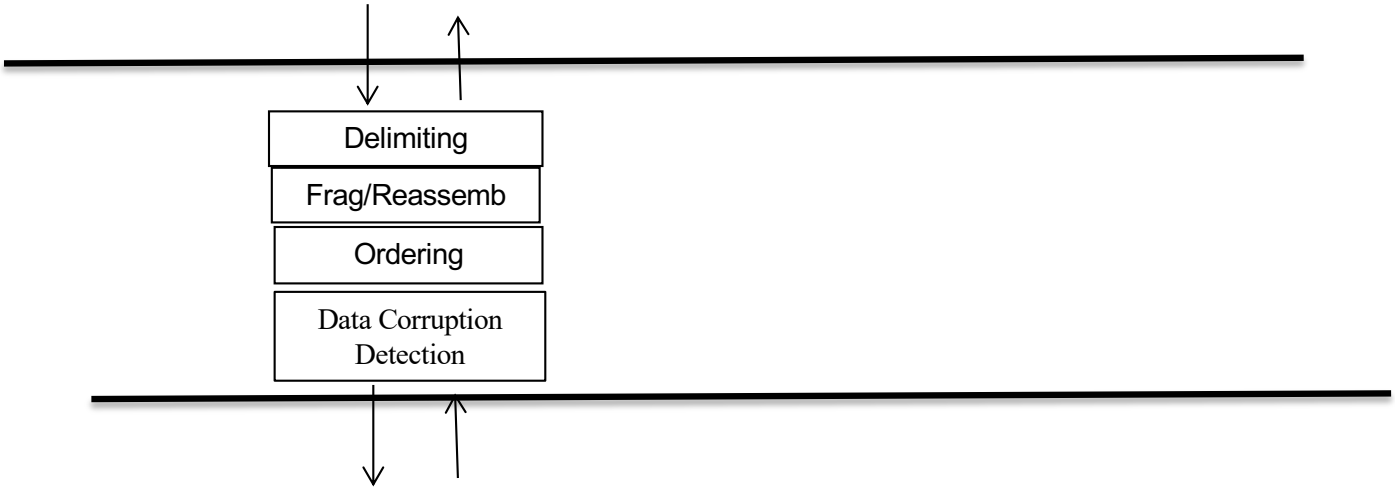
The application should not be able to distinguish a slow local correspondent from operating over the network.

Two systems: how?



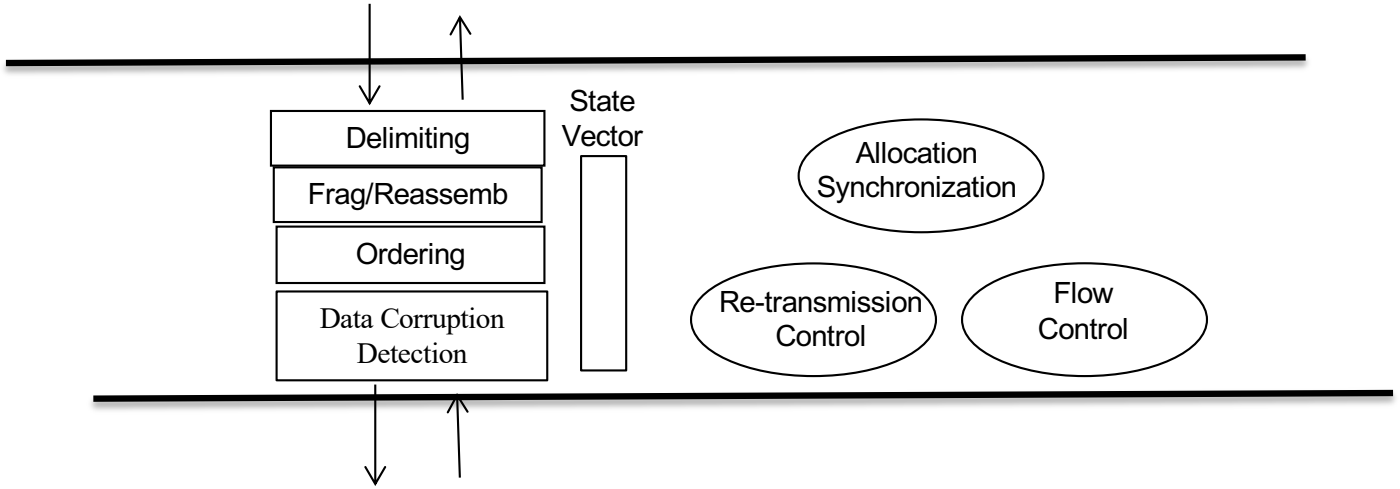
- Communication over a network link: errors can occur
- We need some sort of error and flow control protocol (EFCP)
  - What are the functions this protocol needs?

Functions of a Protocol



- **Data Corruption Detection/Correction** – Making sure that the bits in the packet are correct (within some probability)
- **Delimiting** – Packing SDUs into the User Data of PDUs and marking the boundaries of an SDU, so we can find them again
- **Fragmentation/Reassembly; Combining/Separation** – Last step of Delimiting, creating User-Data fields of the PDUs
- **Ordering/Sequencing** – Putting the PDUs in order, or putting sequence numbers on them for sending
- **Loss and Duplicate Detection** – Detecting entire lost PDUs, or duplicates due to retransmission

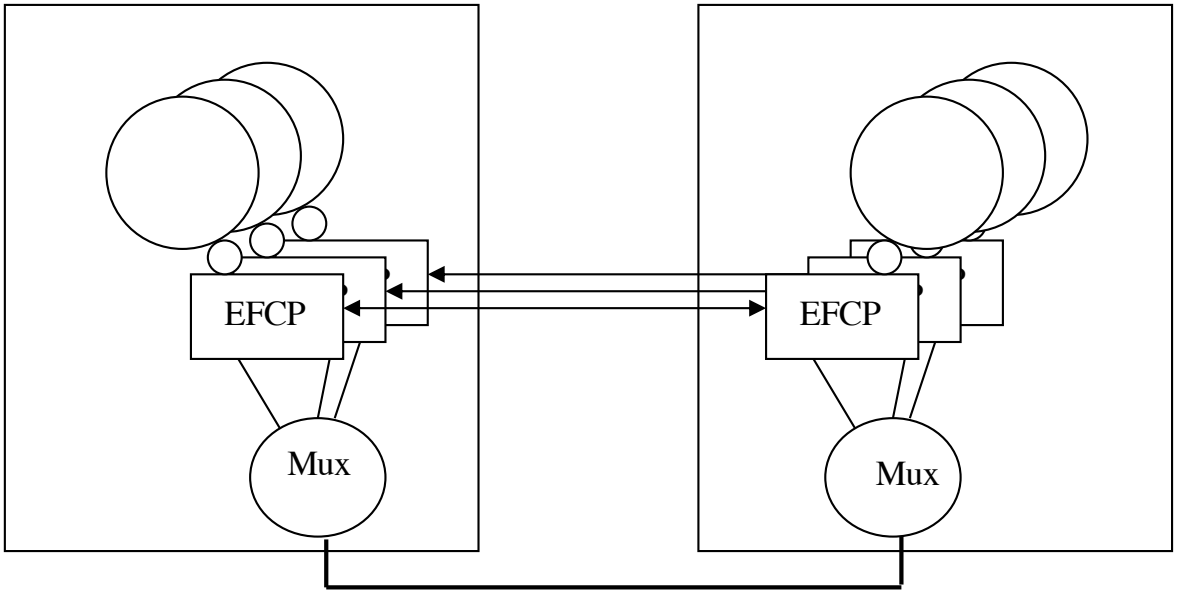
Functions of a Protocol /2



- **Retransmission Control (Acknowledgement)** – Recovering from lost PDUs.
- **Flow Control** – Keeping the sender from overrunning the receiver, using either a Sliding Window or rate-based.
- These two are feedback mechanisms, therefore we need:
- **Synchronization** – Ensuring that the sender and receiver maintain consistent state relative to each other.
- **Allocation/Deallocation** – Allocating/Deallocating ports and creating and terminating synchronization.

More than one application per system at the same time

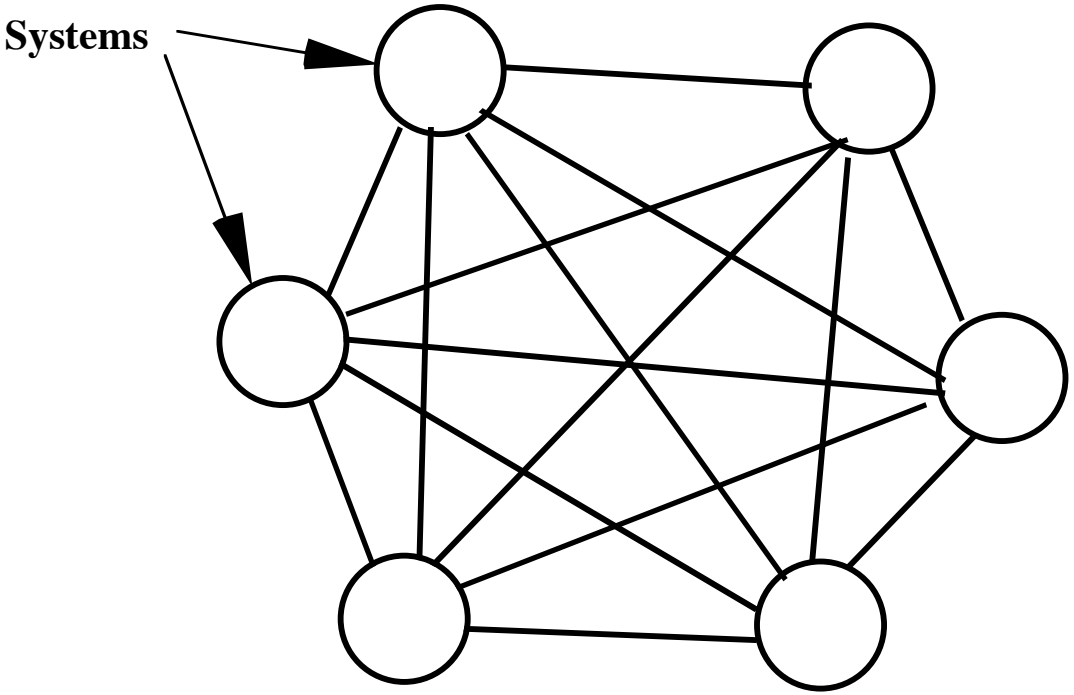
- Requires two new capabilities
- EFCP must be able to distinguish one flow from another
  - Typically use the port-ids of the source and destination



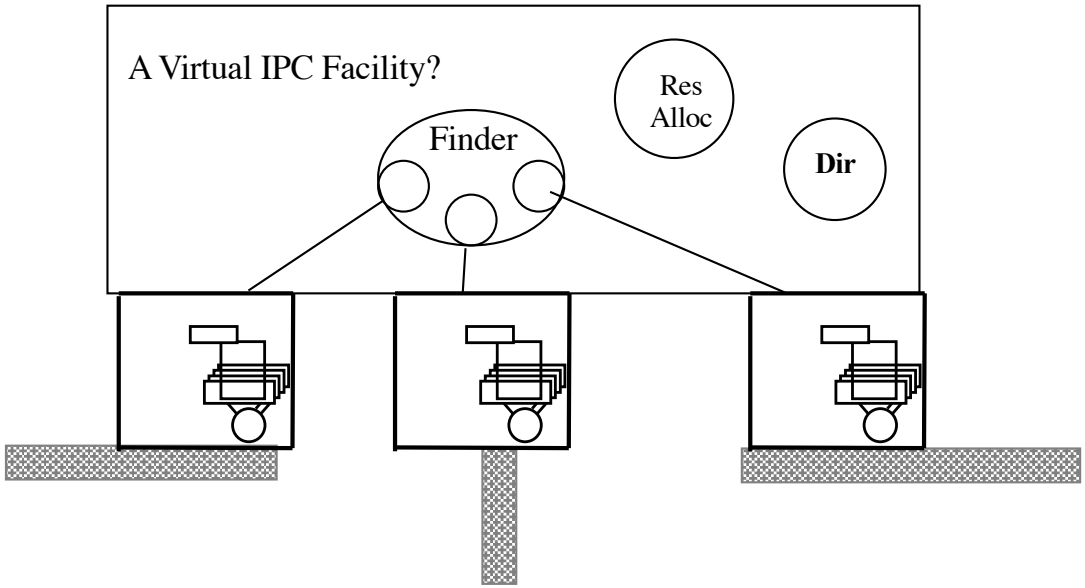
Connection-id

Dest-port	Src-port	Op	Seq #	CRC	Data

Communication with N systems

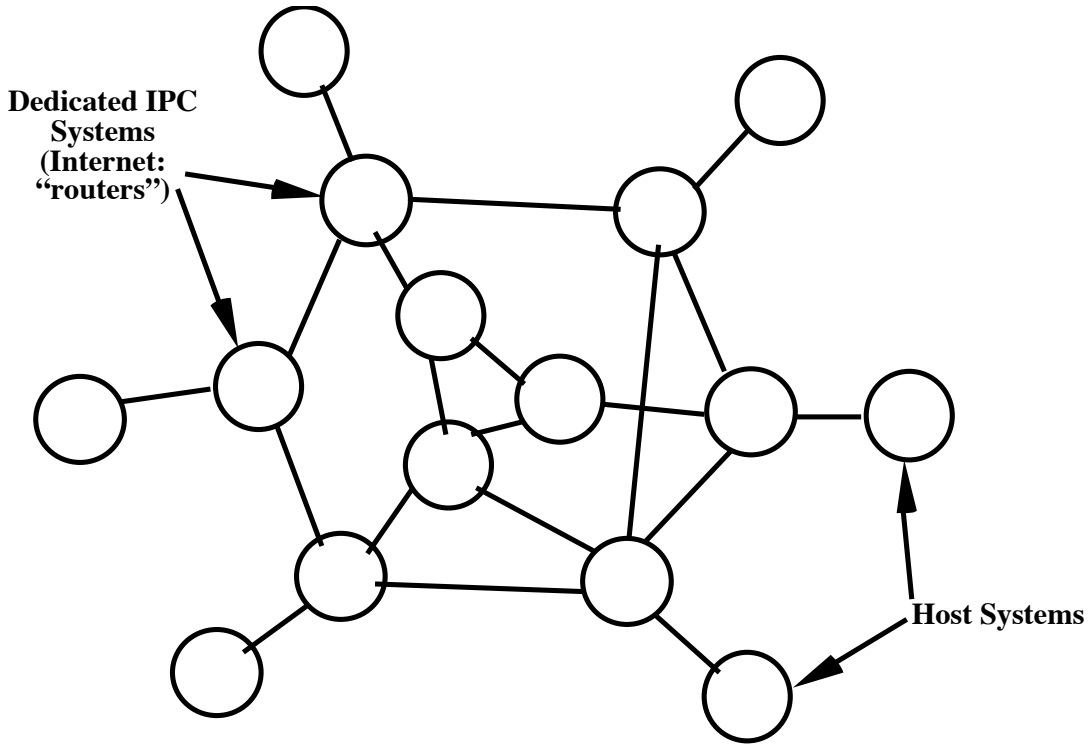


Communication with N systems /2: A little re-organizing



- So we have a Distributed IPC Facility (DIF) for each interface, then to maintain the API we need an application over all of them.
  - The user shouldn't have to figure out which wire to use.
- But this isn't going to scale and get expensive fast. We need a cheaper way to do this.

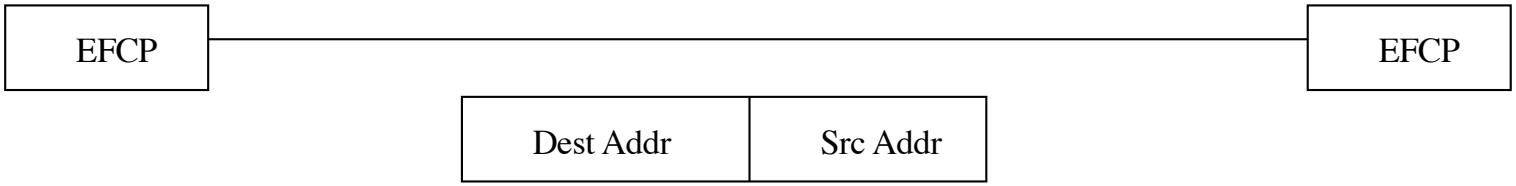
Communication with N systems /3



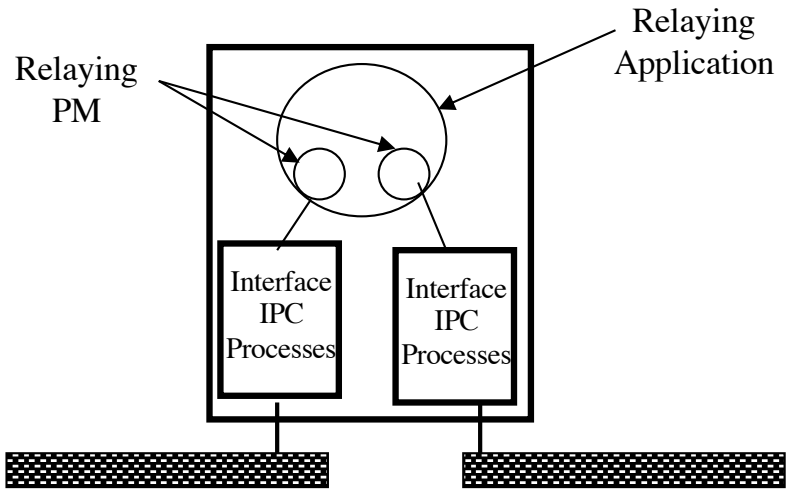
By dedicating systems to IPC, reduce the number of lines required and even out usage by recognizing that not everyone talks to everyone else the same amount.

Communication with N systems /4

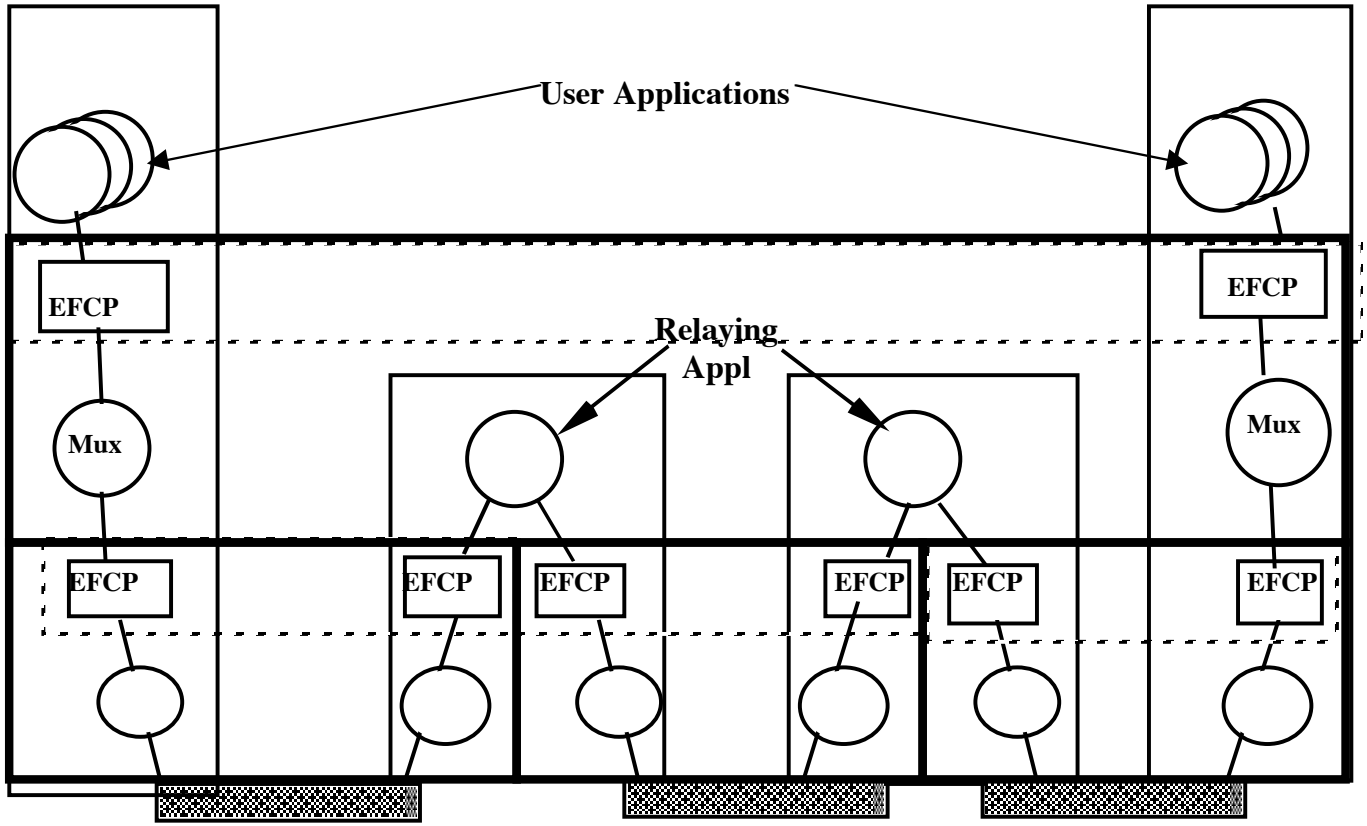
- Relaying systems over a wider scope requires carrying addresses
- Creates more problems: e.g., can't avoid transient congestion and bit errors in relay memories; what about message order? etc.
  - Will have to have an EFCP operating over the relays



Common Relaying and Multiplexing Application Header



A recursive structure!

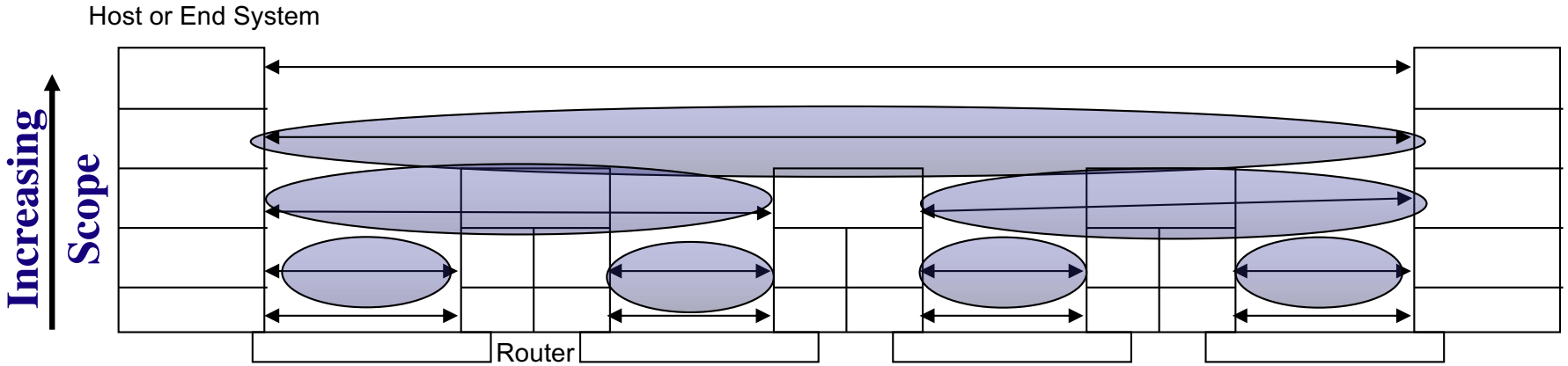


Implications

- The essential role of layers is *not* to be a collection of a specific set of functions but instead to define a communication *scope*
  - *Different from what OSI and DoD models say!*
- Some essential functions exist at every layer, although they might be instantiated differently depending on the environment
  - Even if the functions are exactly the same, it's not a repetition: they only occur once *per scope*
- Layers can be stacked as needed, there should be no dependencies and there can always be the same interfaces between layers
- “Cross-layer communication” is a hack that may be necessary when the vertical interfaces aren’t well enough defined

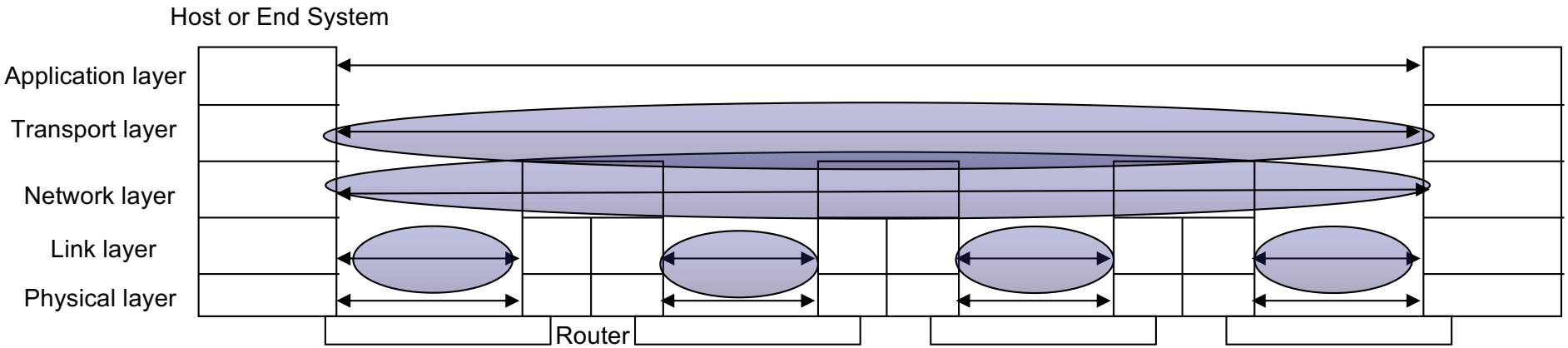
Layers

- Networks have loci of distributed shared state with different scopes
- At the very least, differences of scope require different layers
- If layers have the same scope, their functions should be completely independent



- Does this picture show the Internet?

The Internet

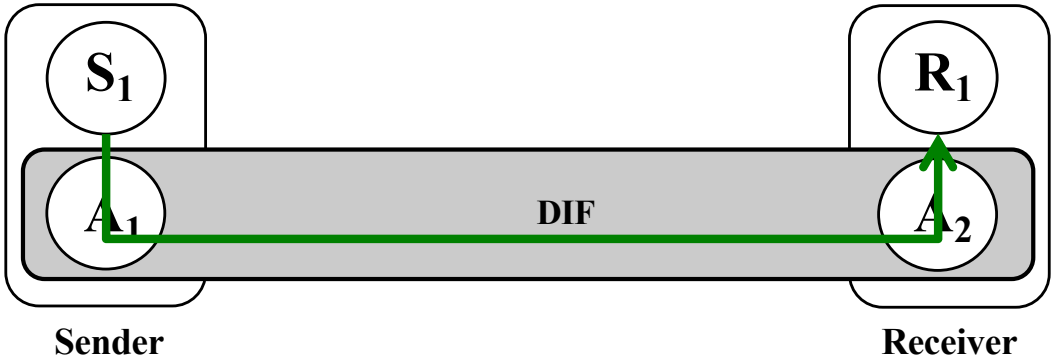


- What does the recursive structure tell us, what can we learn?
- Next: examples

Example problem: IP fragmentation
Dependencies between layers with the same scope

- IP must get all fragments of a packet to reassemble
  => What if a fragment is lost?
  - Early 70's: **IP was split from TCP**
  - TCP's retransmissions are distinct, not recognized by IP
  - Must hold fragments for a long time, in vain => wastes buffer space
- Fix: Path MTU Discovery (PMTUD)
  - Disable fragmentation & find the largest possible packet that can be sent: start big, reduce size upon incoming ICMP error
  - Actually this doesn't work either: many nets filter ICMP
- Solution: PAcketization Layer PMTUD (PLPMTUD)
  - start small, assume ok when ACKs arrive
  - ...can you see (layering) problems with this?

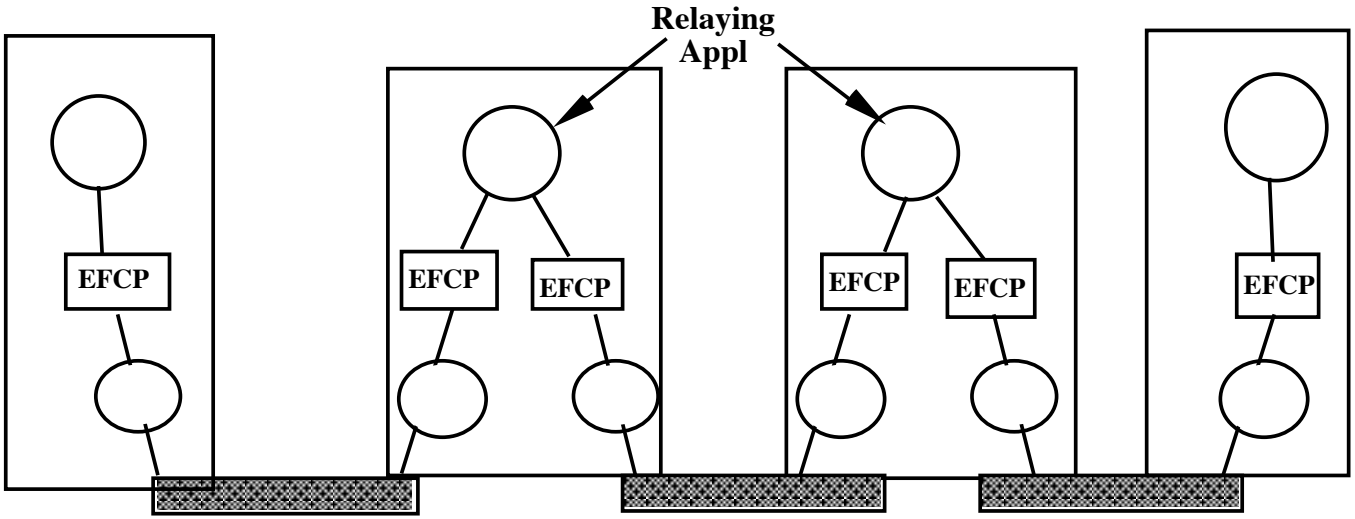
Developing flow + congestion control



- How fast can we send?
  - Vertical interface should probably include a “rate”
  - EFCP should implement flow control

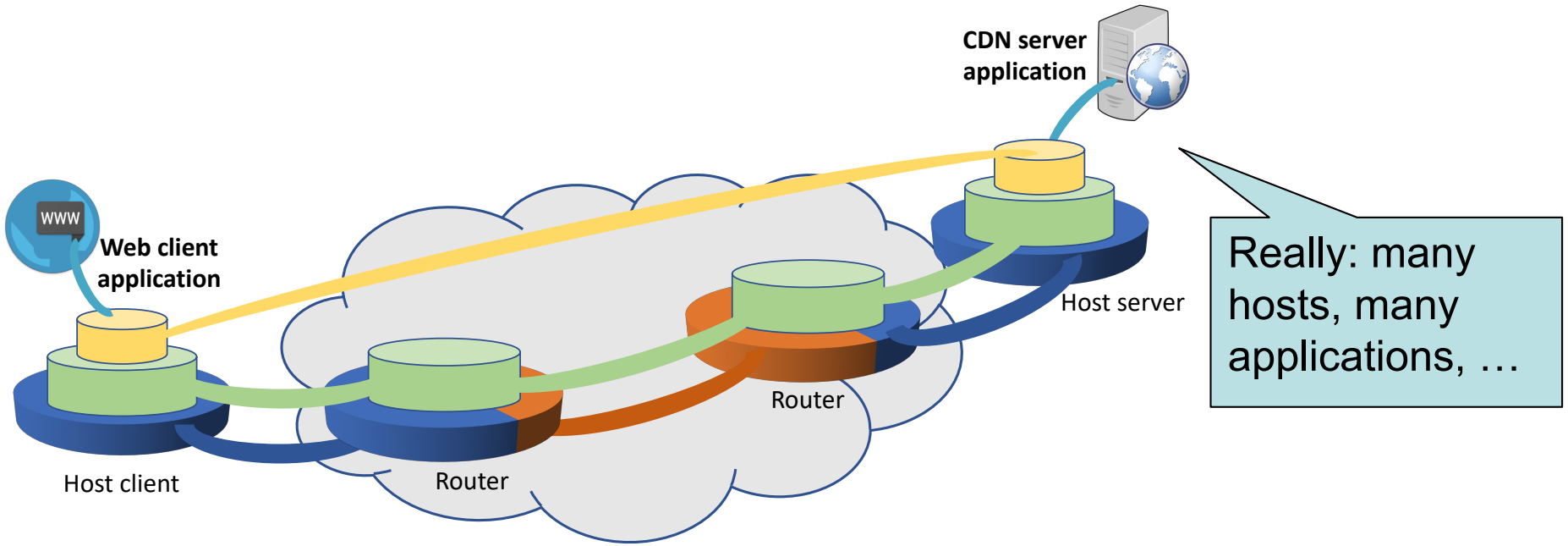
Or something similar...e.g. a credit (window)

Developing flow + congestion control /2



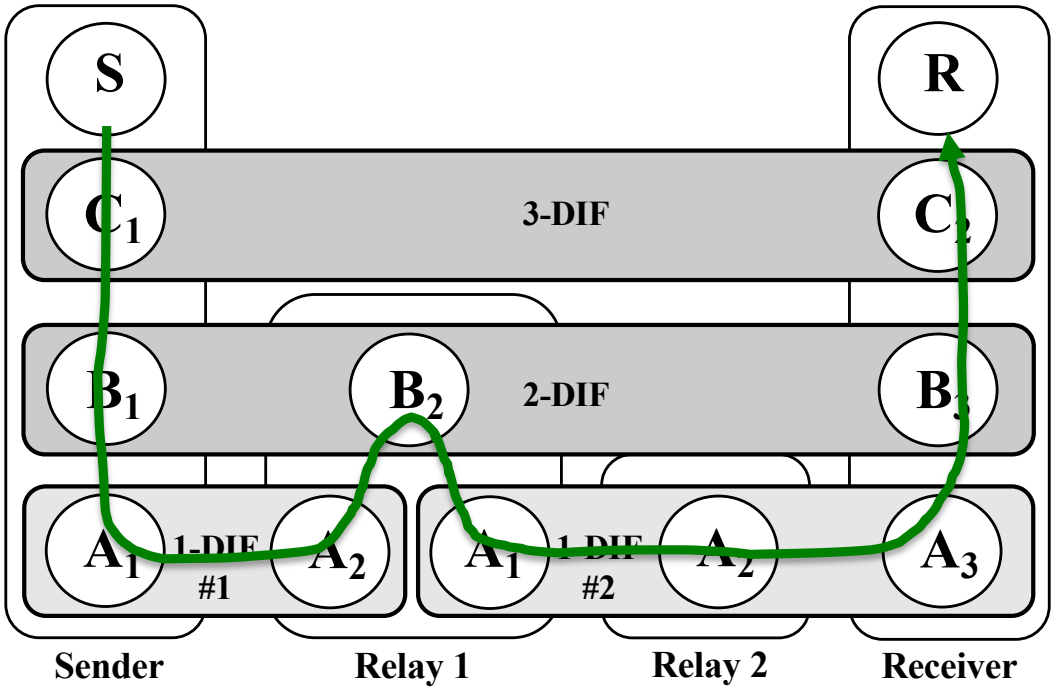
- How fast can we send?
  - Vertical interface should probably include a “rate”
  - EFCP should implement flow control

Developing flow + congestion control /3

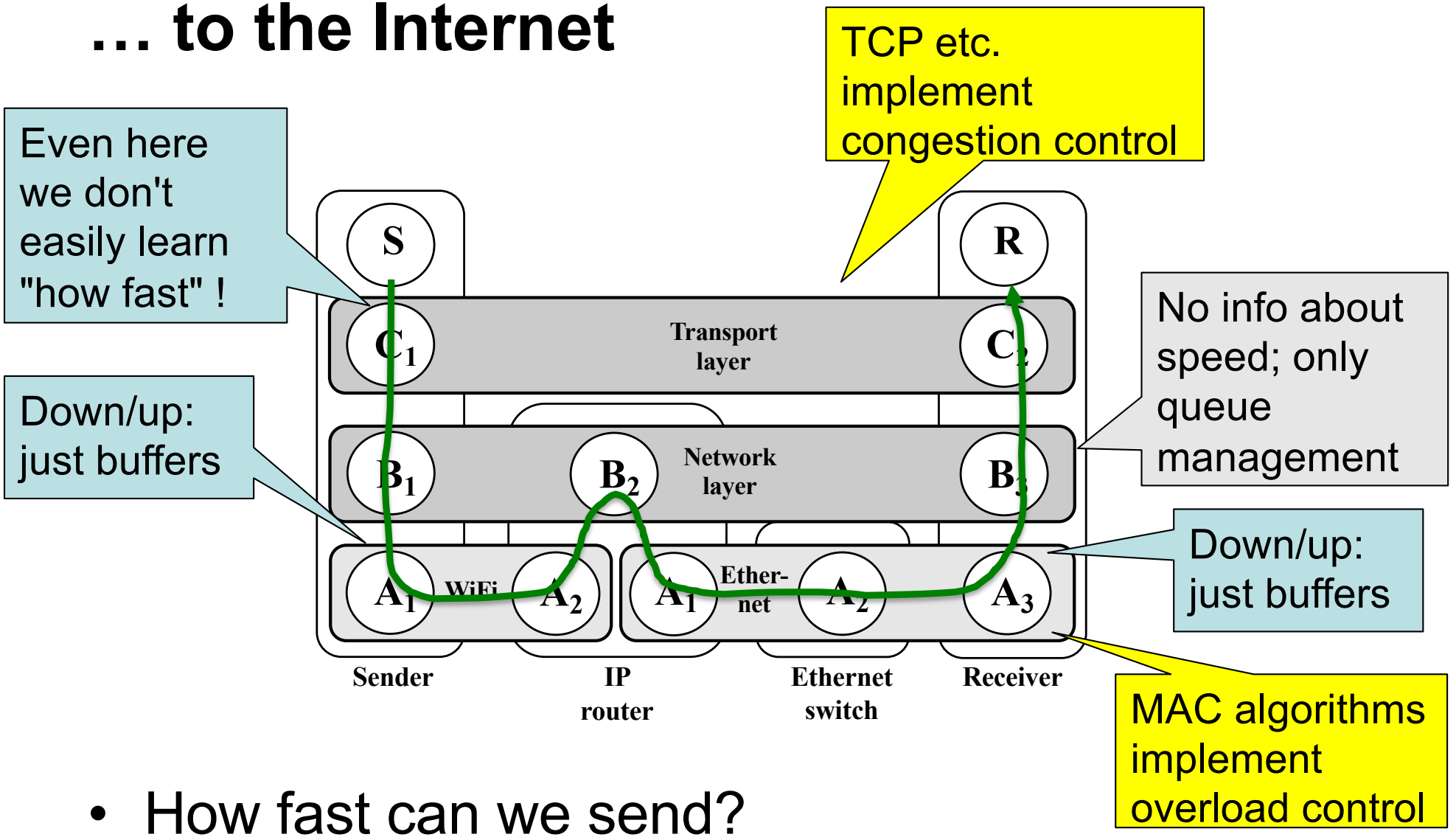


- How fast can we send?
  - Vertical interface should probably include a “rate”
  - EFCP should implement flow + congestion control !
  - Probably at least in the green DIF, because this is where MUXing occurs (in hosts and routers)
  - As we will see, Internet: really only in the yellow DIF…

Context switch: from RINA…



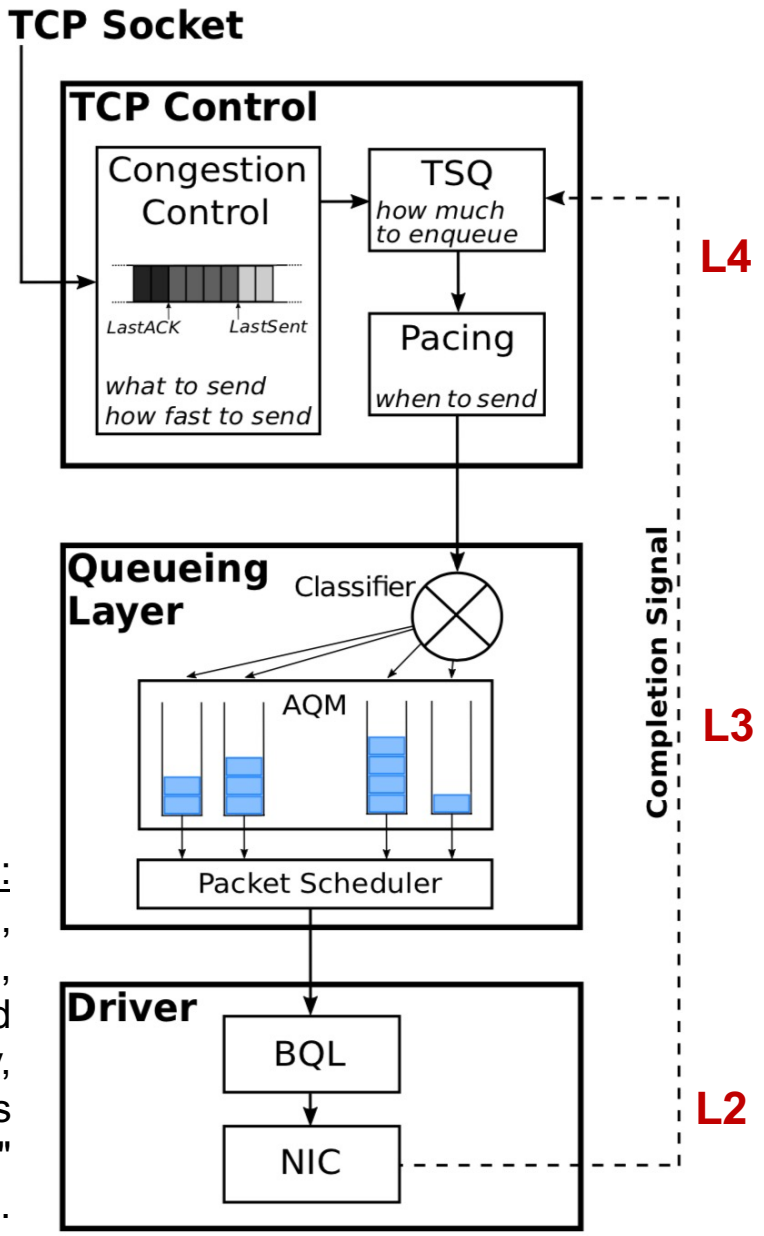
… to the Internet



- How fast can we send?
  - Who's deciding what, what are the interfaces?

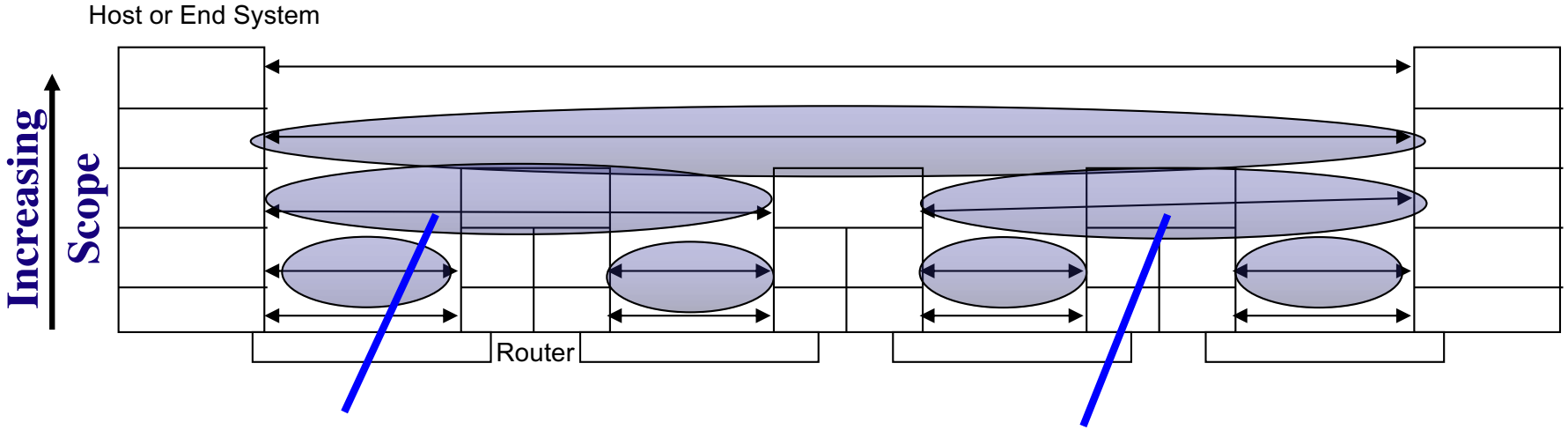
E.g. Linux

- Useful TCP "speed" feedback: "low water mark"
  - Patch with "TCP_NOTSENT_LOWAT" socket option, 2013
  - Apple computers also offer this



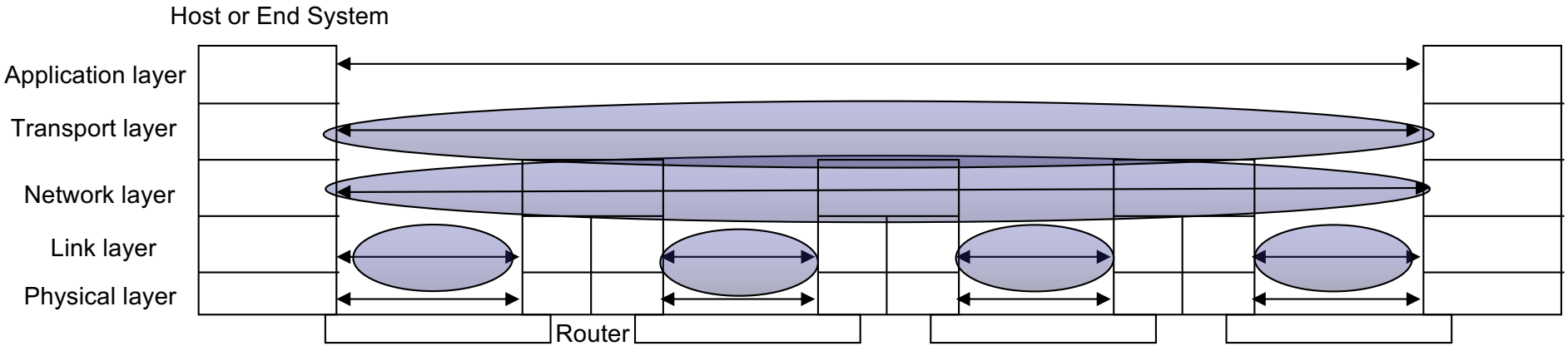
From:
C. A. Grazia, N. Patriciello, T. Høiland-Jørgensen, M. Klapez, M. Casoni and J. Mangues-Bafalluy, "Adapting TCP Small Queues for IEEE 802.11 Networks," PIMRC 2018, Bologna.

Addresses



- Network 1 is completely unaware of addresses of type 1, 2, 3,..
  - Networks and their addresses could dynamically be registered (attached)
  - This is also more secure

Internet addresses



- No intermediate layer of smaller scope
  - IP addresses are long, routers need to store a lot of information
- Also, running out of addresses: IPv6 is the "solution"



- Network Address (and Port) Translators (NA(P)Ts): more practical?
  - Store mapping: "local IP address + TCP port ⇔ global IP address + TCP port"
  - Problem because depends on transport protocol, inside the network... e.g., need to send keep-alives to keep state up
  - Peer-to-peer communication, both peers behind NATs: how to talk? Complex solutions (STUN, TURN, ICE protocols)

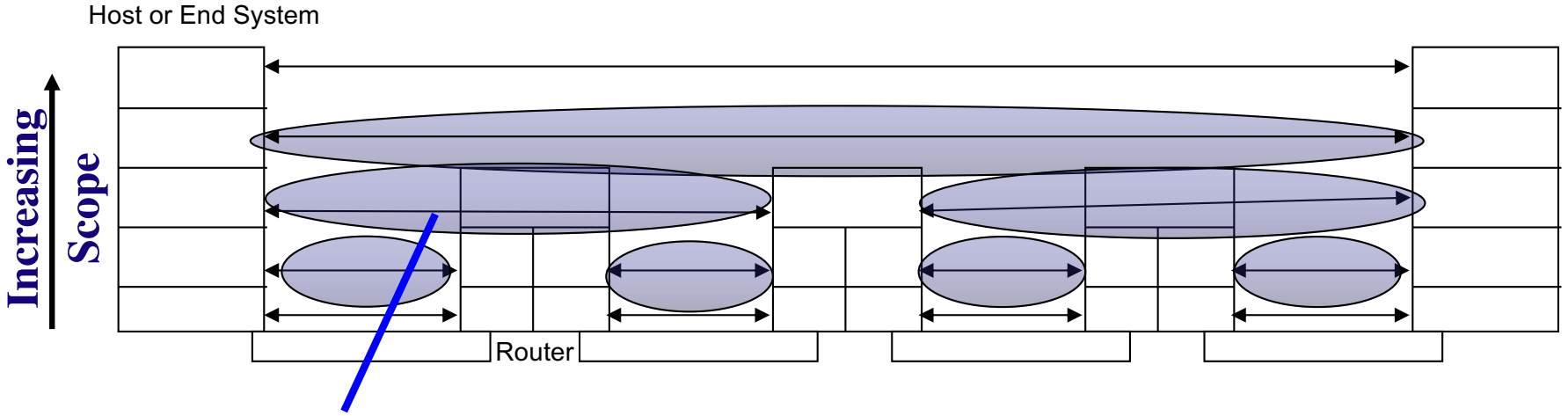
IP mobility

- Long ago, I moved from Bogstadveien to Thorvald Meyers Gate
  - I registered to have mail forwarded for a month for free (postens flyttehjelp)
  - This gave me time to inform everyone about my new address (posten offers an easy service to do that)
  - After this month, I had no more business with the old address
- Imagine a world without addresses
  - But where, instead, my name is Bogstadveien
  - Moving: must use “postens flyttehjelp” forever!
- How does the Internet solve this today?
  - Multipath TCP (MPTCP): hosts share all their IP addresses with each other, can switch without interruption
  - Applications should bind to names, not IP addresses (e.g., with the new Transport Services (TAPS) API)

or 129.240.118.130, for example...

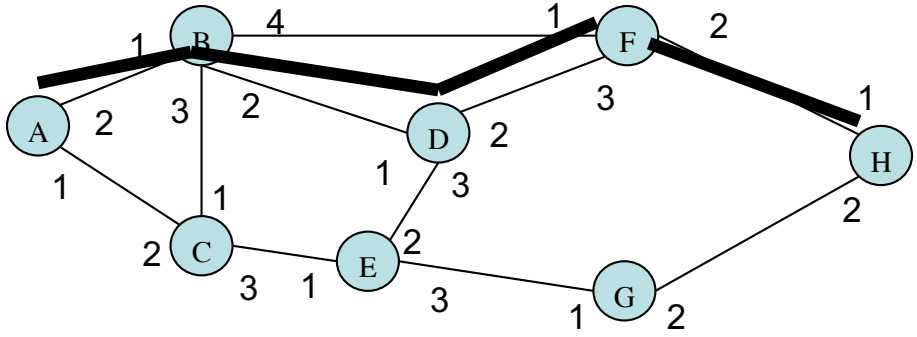
Mobile IP does exactly this.

Routing



- How would one route between the addresses on network 1?
- Next: an example with multi-homing and link failure

Using node addresses



- Since we want to emphasize that we are naming nodes, lets just use the letters for addresses. But we still have to say which wire to send them on.
  - There are two cases in general:
    - Point-to-point Wire: No need for lower layer addresses, use local identifiers.
    - Multi-access wired or wireless: Here we need addresses, use MAC addresses.
  - We have only wires, so lets assign small integers as the local interface identifiers.
- Now lets say that A wants to send a packet to H, so it goes to DNS and looks up the address and gets H. Now these nodes have been running a routing algorithm and the route is A, B, D, F, H.
- So what do the router tables look like? (see next slide)

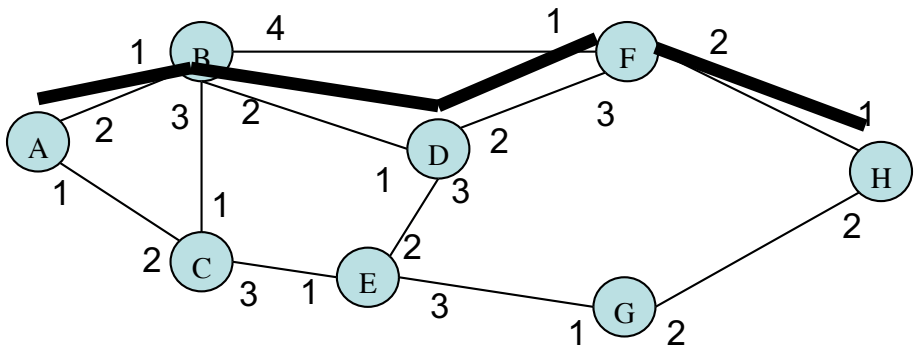
Forwarding Tables for Each Router
Using Node Addresses
Outgoing Interface

Destination Address	A	B	C	D	E	F	G	H
A	-	1	2	1	1	3	1	1
B	2	-	1	1	1	3	1	1
C	1	3	-	1	1	3	1	1
D	2	2	1	-	2	3	1	1
E	1	2	3	3	-	3	1	1
F	2	2	1	2	2	-	1	1
G	1	2	3	3	3	3	-	2
H	2	2	1	2	2	2	2	-

Forwarding Table for Each Router
Using Next Hop

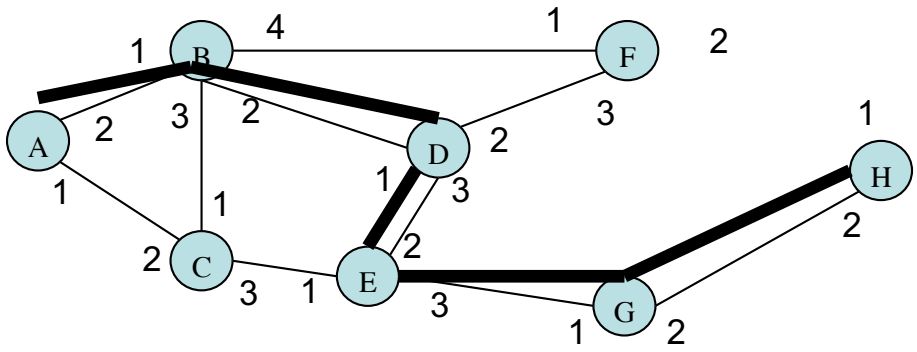
Destination Address	A	B	C	D	E	F	G	H
A	-	A	A	B	C	D	E	F
B	B	-	B	B	C	D	E	F
C	C	C	-	B	C	D	E	F
D	B	D	B	-	D	D	E	F
E	B	D	E	E	-	D	E	F
F	B	D	B	F	D	-	E	F
G	B	D	E	E	E	D	-	G
H	B	D	B	F	D	H	H	-

Sending a packet from A to H



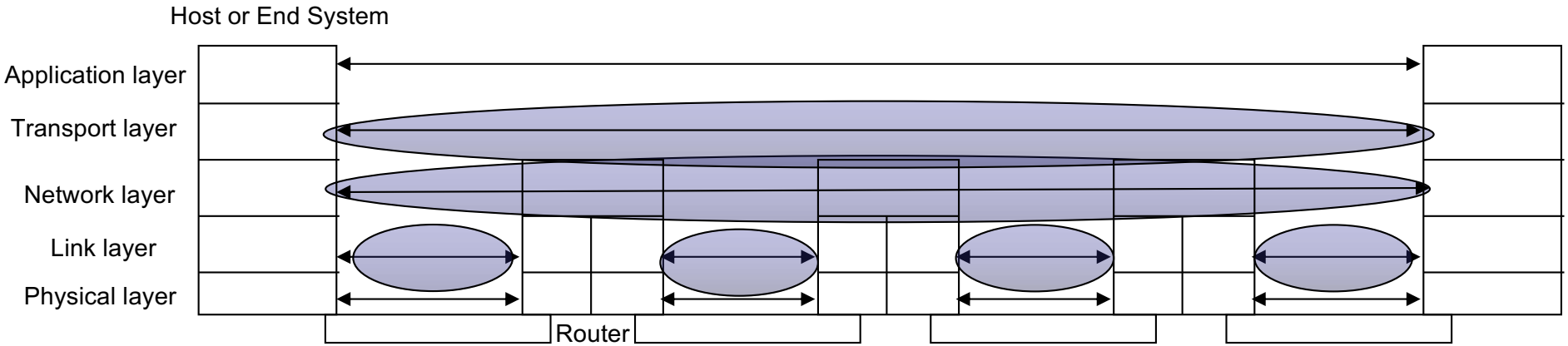
- A has a packet addressed to H:
  - A consults its Forwarding Table and sends the packet on interface 2 to B.
  - B consults its Forwarding Table and sends the packet on interface 2 to D.
  - D consults its Forwarding Table and sends the packet on interface 2 to F.
  - F consults its Forwarding Table and sends the packet on interface 2 to H.

What happens when link F - H fails?



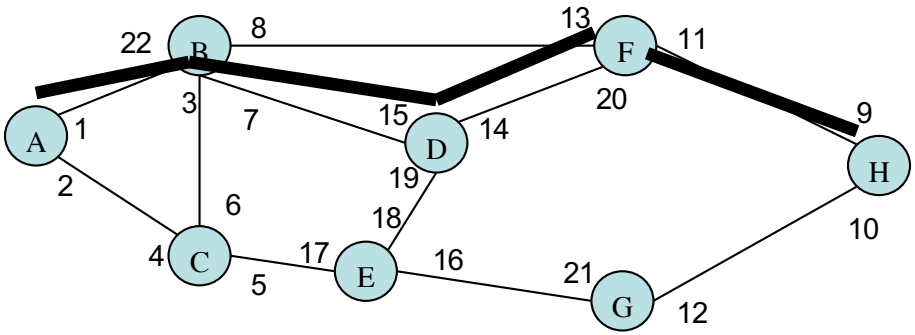
- What happens if just after sending, the F-H link fails?
  - In a few 10s of ms, a routing update is done, and a new routing table is generated.
  - The packet gets to D after the routing update has concluded and is delivered to H as if nothing happened.
  - Packets that might have been between B, D, and F might get re-routed. Only packets on the wire from F to H would be lost.

Internet routing



- Always between IP addresses
- What if a node has multiple interfaces (= is multihomed) ?
  - Need to use an IP address per interface to distinguish them
  - Hence, IP address identifies the interface, not the node
- Next: example, using “normal” letters and numbers for clarity

Internet multihoming



- Internet addresses name the interfaces. Let's see what this does
- Assume: A wants to send a packet to H, so it goes to DNS and looks up the address and gets 9. Now these nodes have been running a routing algorithm and the route is A, B, D, F, H.
- So what do the router tables look like? (next slide)

Forwarding Tables for Each Router

Using Interface Addressing

Outgoing Interface

Destination Address	A	B	C	D	E	F	G	H
1	-	22	6	15	18	20	21	9
2	-	22	4	19	17	20	21	10
3	1	-	6	19	17	20	21	10
4	2	1	-	15	18	20	12	9
5	2	3	-	19	17	20	21	10
6	1	3	-	19	15	20	12	9
7	2	-	5	15	15	20	21	9
8	2	-	6	14	18	13	12	9
9	1	7	5	14	18	11	21	-
10	2	3	5	19	16	20	12	-
11	2	3	5	19	16	-	12	9
12	1	3	5	14	18	11	-	10
13	1	7	5	15	17	-	21	10
14	1	8	5	-	16	20	12	9
15	1	7	6	-	17	20	21	9
16	2	7	5	14	-	11	21	10
17	2	3	5	15	-	20	12	9
18	2	7	4	19	-	11	12	9
19	2	3	5	-	18	11	21	10
20	1	7	5	14	18	-	21	10
21	2	3	5	19	16	20	-	9
22	1	-	4	19	17	20	21	10

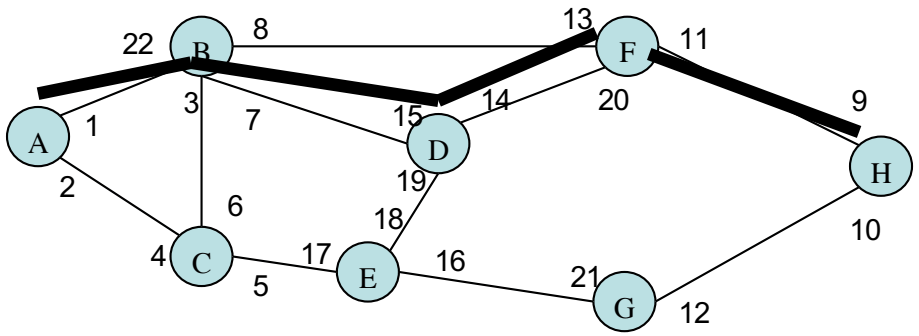
Forwarding Tables for Each Router

Using Next Hop Addressing

Next Hop

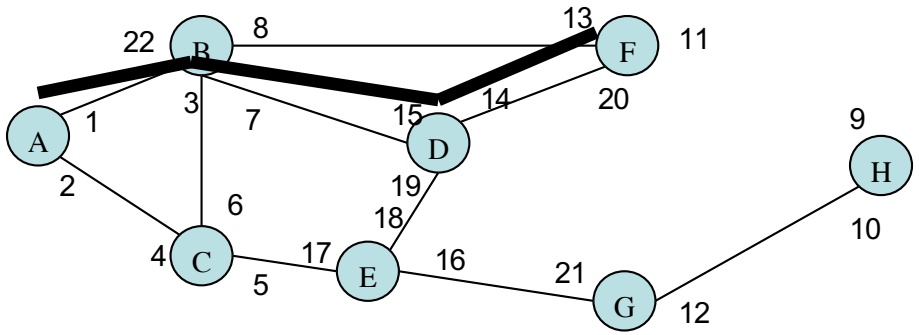
Destination Address	A	B	C	D	E	F	G	H
1	-	1	3	7	19	14	16	11
2	-	1	2	18	17	14	16	12
3	22	-	3	18	17	14	16	12
4	4	1	-	7	19	14	10	11
5	4	6	-	18	17	14	16	12
6	22	6	-	18	7	14	10	11
7	4	-	17	7	7	14	16	11
8	4	-	3	14	19	13	10	11
9	22	15	17	14	19	9	16	-
10	4	6	17	18	21	14	10	-
11	4	6	17	18	21	-	10	11
12	22	6	17	14	19	9	-	12
13	22	15	17	7	17	-	16	12
14	22	13	17	-	21	14	10	11
15	22	15	3	-	17	14	16	11
16	4	15	17	14	-	9	16	12
17	4	6	17	7	-	14	10	11
18	4	15	2	18	-	9	10	11
19	4	6	17	-	19	9	16	12
20	22	15	17	20	19	-	16	12
21	4	6	17	18	21	14	-	11
22	22	-	2	18	17	14	16	12

A packet is sent with destination address 9 from A



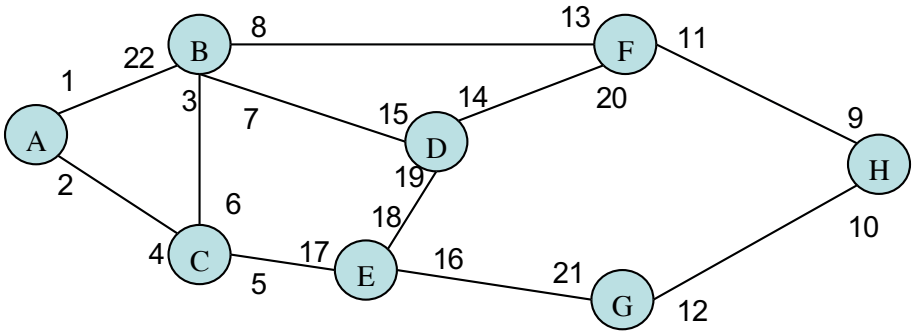
- A consults its Forwarding Table and sends it on outgoing address 1, next hop 22.
- B consults its FT and sends it on outgoing address 7, next hop 15.
- D consults its FT and sends it on outgoing address 14, next hop 20.
- F consults its FT and sends it on outgoing address 11, next hop 9.
- Now another packet is sent from A to address 9, just after it leaves, the interface goes down.

What happens when link F - H fails?



- What happens if the link with address 9 goes down?
  - In a few 10s of ms, a routing update is done, and Addresses 9 and 11 are eliminated from the forwarding table.
  - After several seconds and many retries, A determines that Address 9 is not responding,
  - All TCP connections with Address 9 are terminated.
  - All packets enroute to 9 are lost.
  - Hopefully, there is a second DNS entry that lists H as also at Address 10.
  - Connections are re-established using address 10. Several seconds have elapsed.

A modern Internet solution: MPTCP



- A and H initially tell each other about all their addresses (A: 1, 2; H: 9, 10)
- MPTCP subflows are established; number depends on path manager config.
  - E.g. fullmesh: 4 subflows (1-9, 1-10, 2-9, 2-10)
  - Then, normally: use all of them (schedule traffic, e.g. round-robin or preferring least-RTT path until cwnd full; “cautious” congestion control – be at most as aggressive as 1 TCP across a common bottleneck)
- F-H link down: 1 subflow becomes unavailable, no TCP failure for application
- This is done e.g. by iPhones since iOS 7
  https://support.apple.com/en-us/HT201373

Architectural principles of the Internet

- Yes, there are some...
- Different from recursive principles laid out so far

Two important Internet rules

- End-to-end ("e2e") argument
  - Carefully consider trade-offs between implementing functions inside the network versus end systems
  - Think twice before making the network do it
  - No application-specific functionality in the network
  - Frequently interpreted too strictly:
    "keep the network dumb, put all logic in end systems"

- Postel's robustness principle: "Be conservative in what you send, be liberal in what you accept"
  - Ensures downward compatibility (e.g. "reserved" bits)

.. and problems with them

1. E2E argument
   - Devices in the network don't seem to follow this…
   - "Pervasive Monitoring is an Attack" [RFC 7258]
   - Also: ossification: Middleboxes make assumptions, e.g. about TCP
   - **Solution: encrypt <u>everything</u>, end-to-end !**
     - Also protocol headers (QUIC)
     - Visible header fields: "grease" them!
   - Pure "e2e" restored; no more ossification; good?
   - Also, no more control by ISPs... more control to Google?

2. Postel's robustness principle
   - Ignored "reserved" fields can be used as covert channels

Critical thinking, applied to common Internet views

- NATs are evil.
  IPv6 should be deployed everywhere.
  - or... are NATs doing the right thing?
- IP fragmentation is a bad idea
  - or… does it just highlight a design flaw?
- Middleboxes are bad
  - Or… are we just missing interfaces to cleanly make them a part of the architecture?

A final note about recursive networking

- Recursive InterNetwork Architecture (RINA)
  - By John Day
  - More material: http://pouzinsociety.org

- Research prototypes exist, ... but various issues and unresolved questions
  - Just saying “everything is wrong” does not automatically solve it...
  - So, again: this was about a way of thinking

Thank you!

Questions?