UNIVERSITY OF WATERLOO

CS 856: Programmable Networks

Lecture 9: Applications to Transport and Network QoS

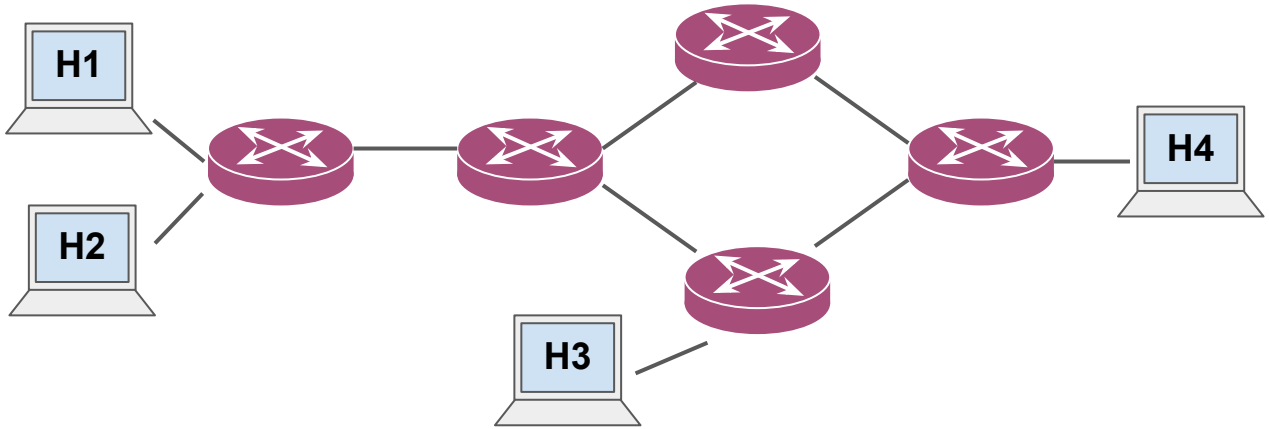
Mina Tahmasbi Arashloo

Winter 2023

Logistics

- Last lecture, Thursday March 30th
  - Any topic you'd like us to cover or discuss in more depth?
- Project presentations, April 4 and April 6
  - 20-minute presentation + 5 min Q&A
  - If you can't make it to class on either day, let me know
- Reviews due **Monday, March 20th, at 5pm.**

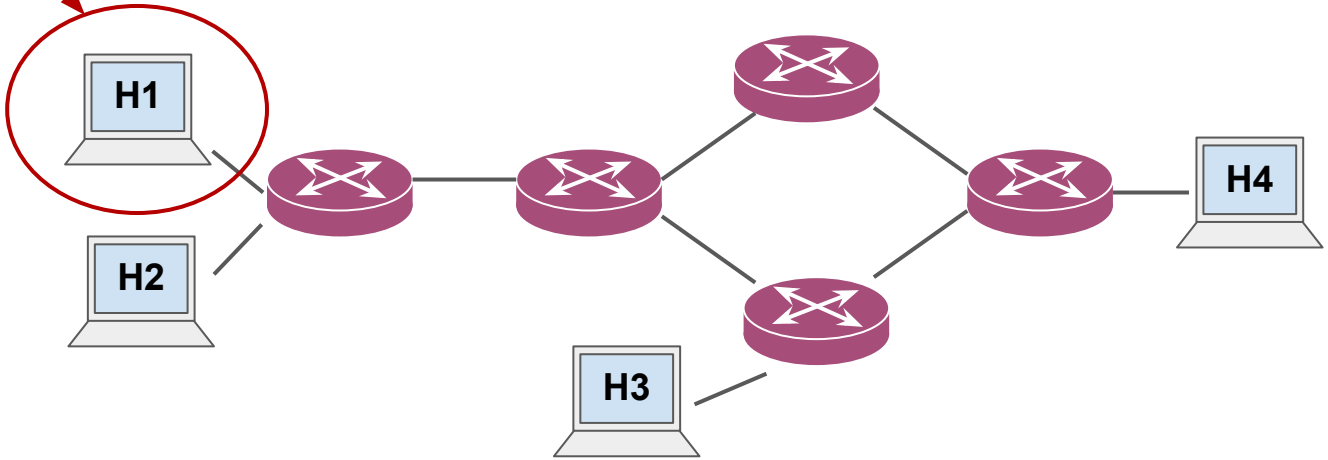
Networks are shared infrastructure



Networks are shared infrastructure

Traffic from multiple "flows" share

- The link to the network
- The memory and computational resources of the host



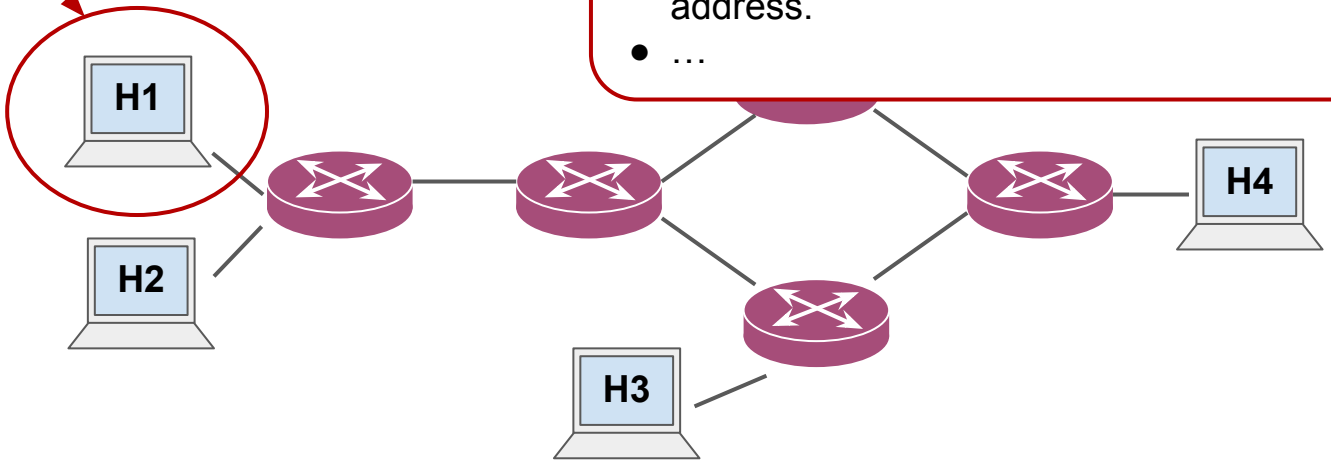
Networks are shared infrastructure

Traffic from multiple "flows" share

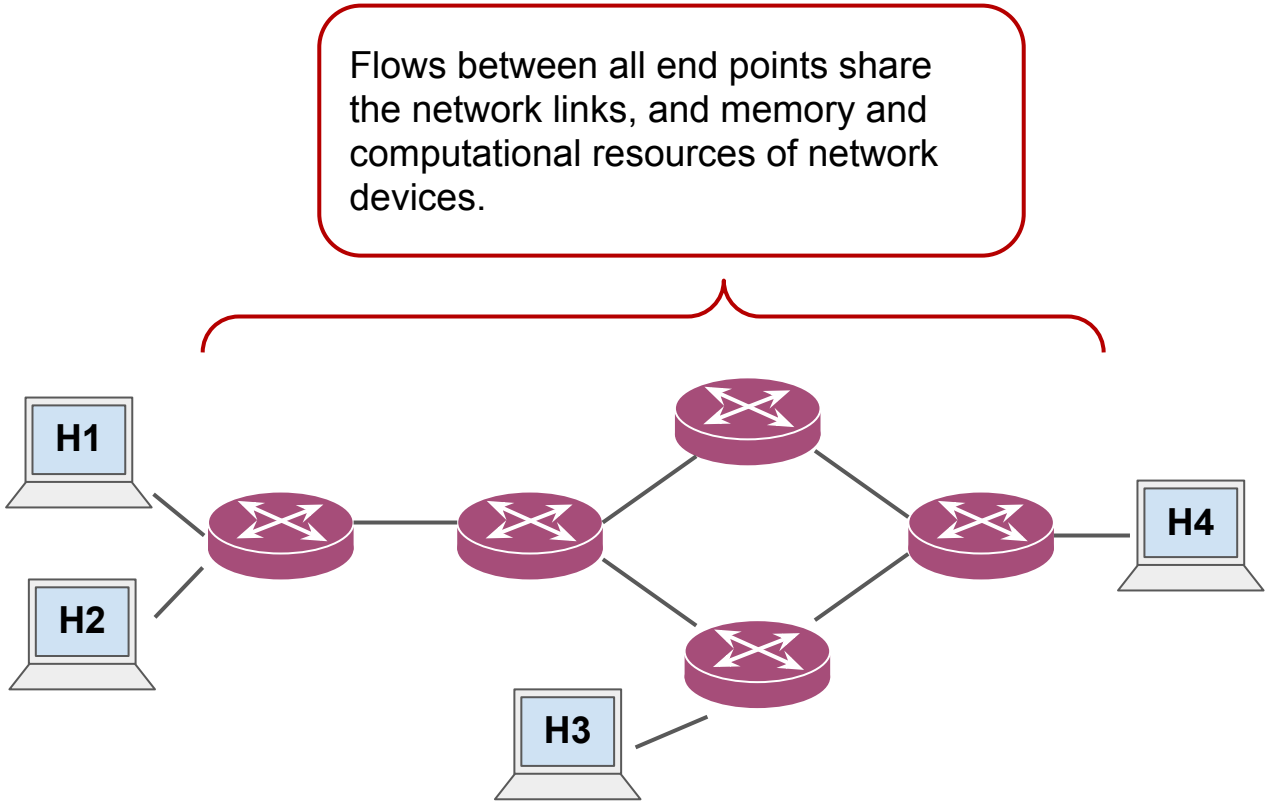
- The link to the network
- The memory and computational resources of the host

You can think of a flow in the broad sense of the term – a set of packets that are treated together as a group for network processing purposes:

- A TCP flow
- Packets originating from the same VM
- All the DNS packets from the same IP address.
- …

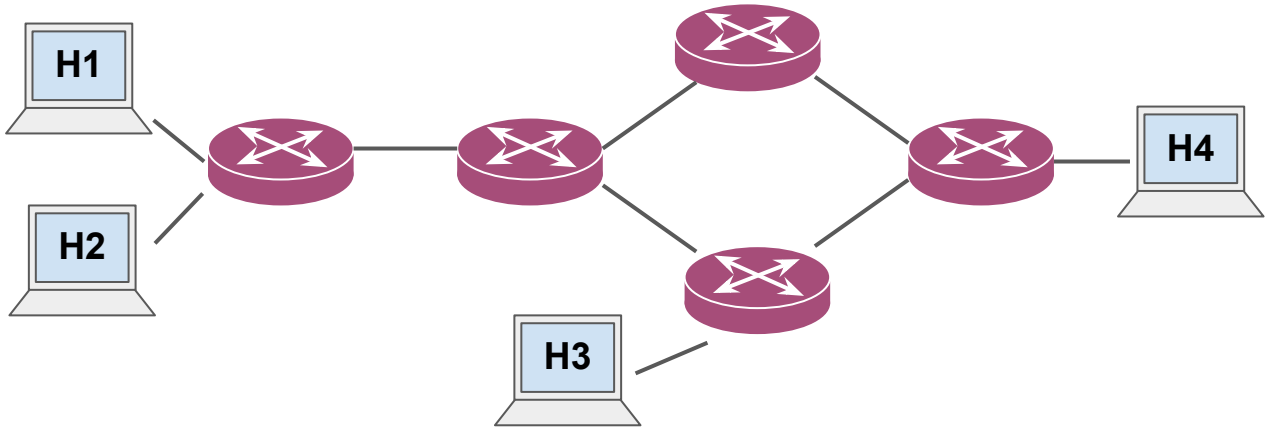


Networks are shared infrastructure



Networks are shared infrastructure

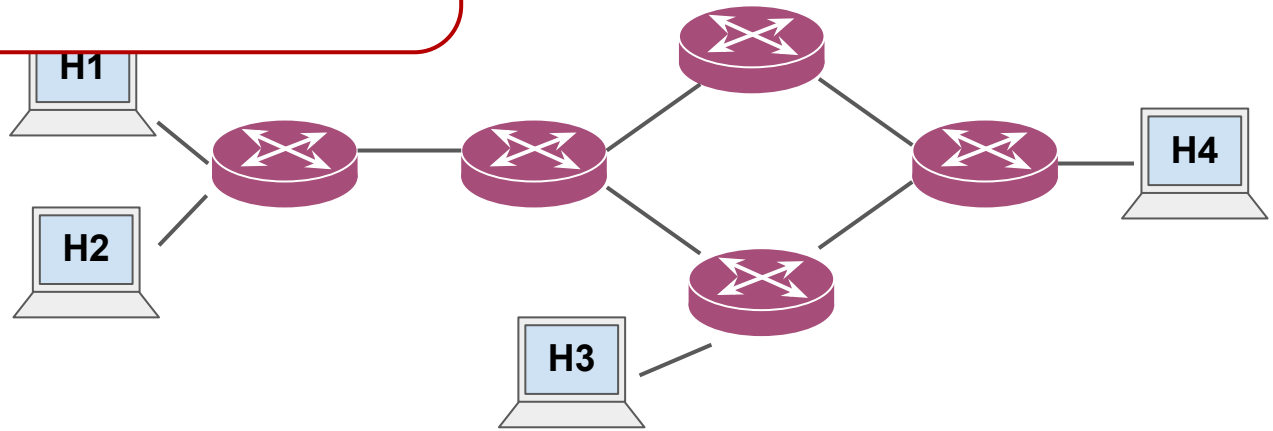
IP only provides *best-effort packet delivery*



Networks are shared infrastructure

IP only provides *best-effort packet delivery*

- Packets can get lost
- No performance (e.g., throughput, latency, jitter) bounds for a flow or classes of flows
- No notion of fairness
- …

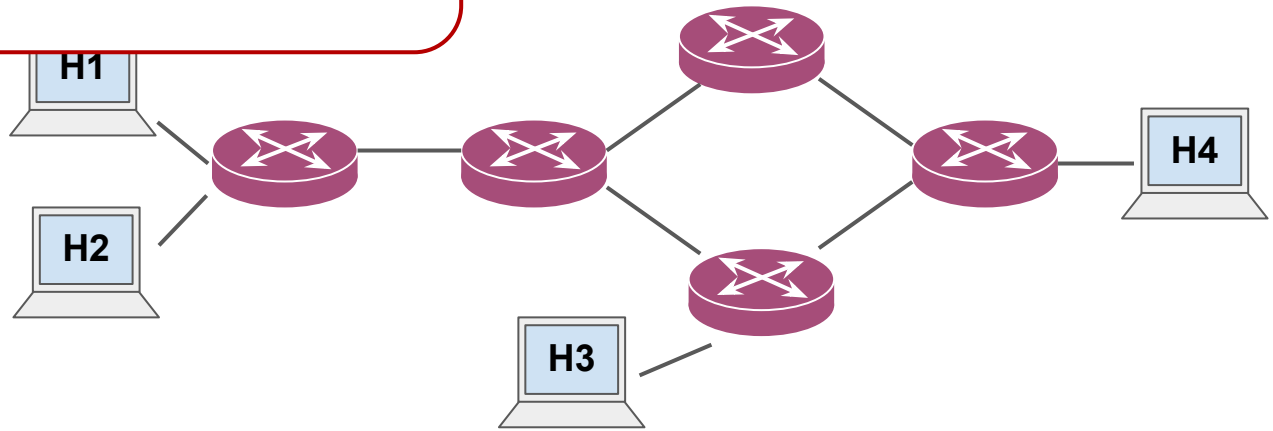


Networks are shared infrastructure

IP only provides *best-effort packet delivery*

- Packets can get lost
- No performance (e.g., throughput, latency, jitter) bounds for a flow or classes of flows
- No notion of fairness
- …

Doesn't have a notion of (or care about) flows

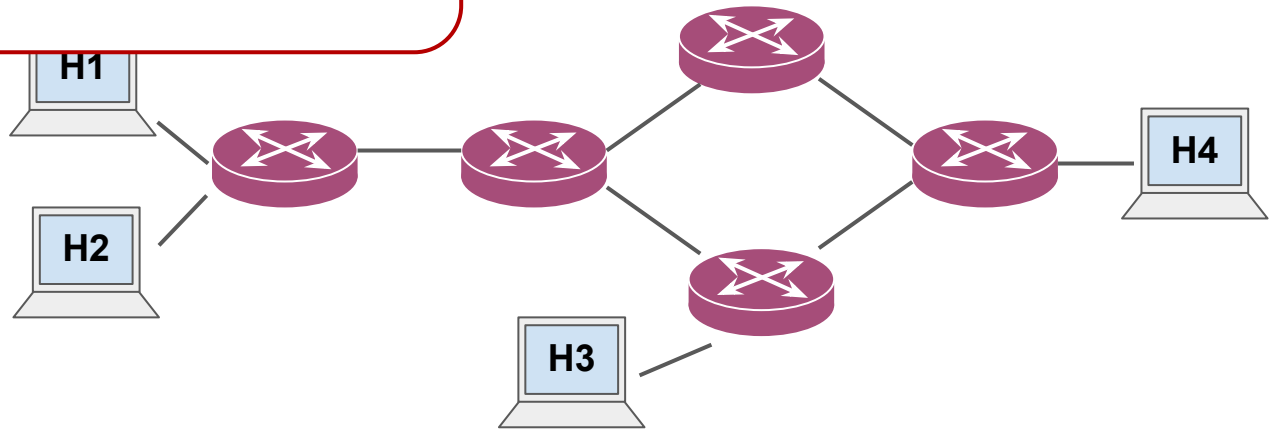


Networks are shared infrastructure

IP only provides *best-effort packet delivery*

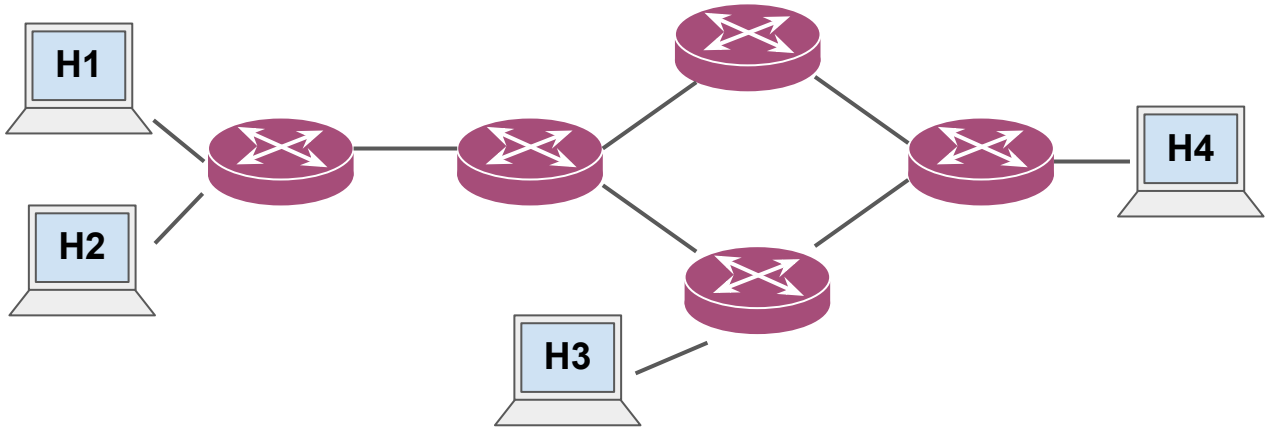
- Packets can get lost
- No performance (e.g., throughput, latency, jitter) bounds for a flow or classes of flows
- No notion of fairness
- …

Doesn't have a notion of (or care about) flows



Networks are shared infrastructure

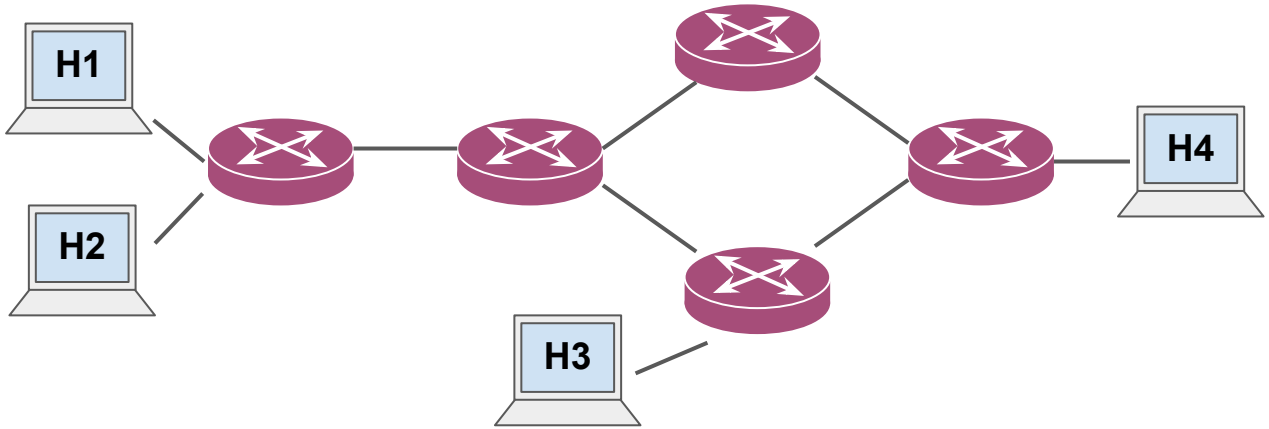
IP only provides *best-effort packet delivery*



Networks are shared infrastructure

IP only provides *best-effort packet delivery*

There are other mechanisms to control/customize how different flows share network resources.

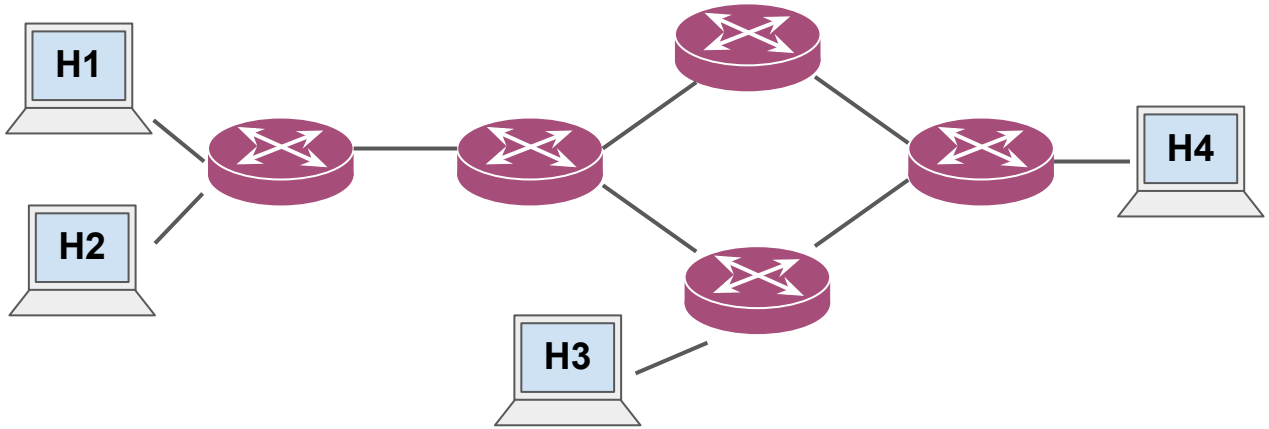


Networks are shared infrastructure

IP only provides *best-effort packet delivery*

There are other mechanisms to control/customize how different flows share network resources.

- end-to-end congestion control
- packet scheduling
- active queue management
- …

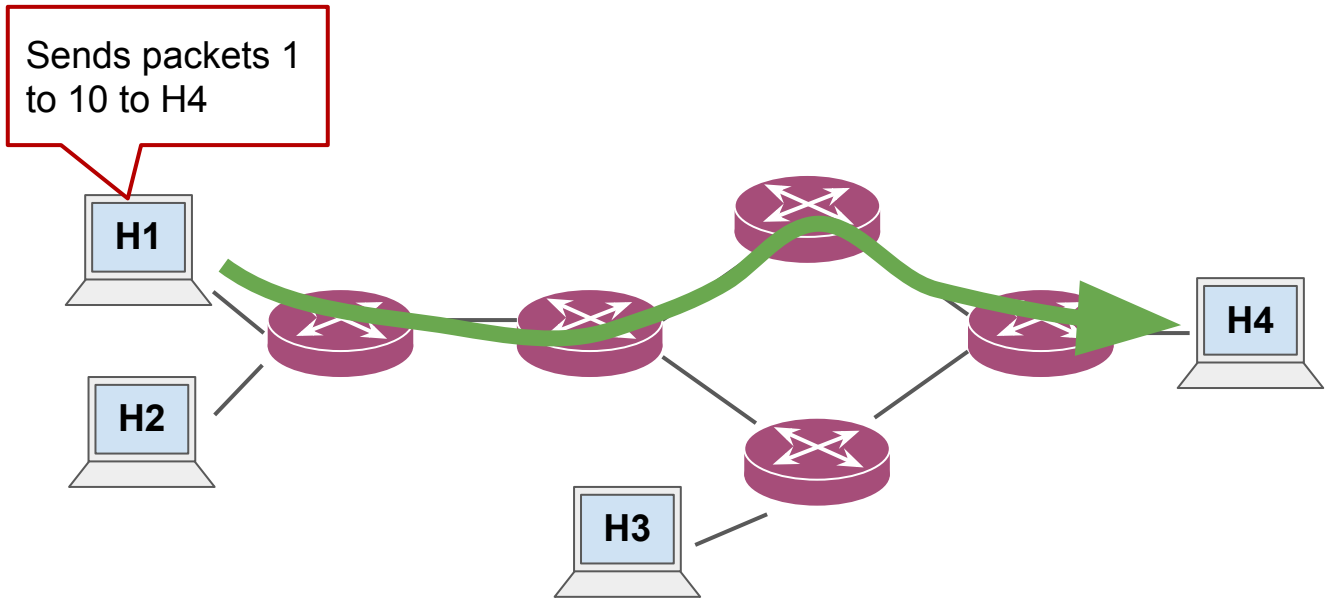


End-to-End Congestion Control

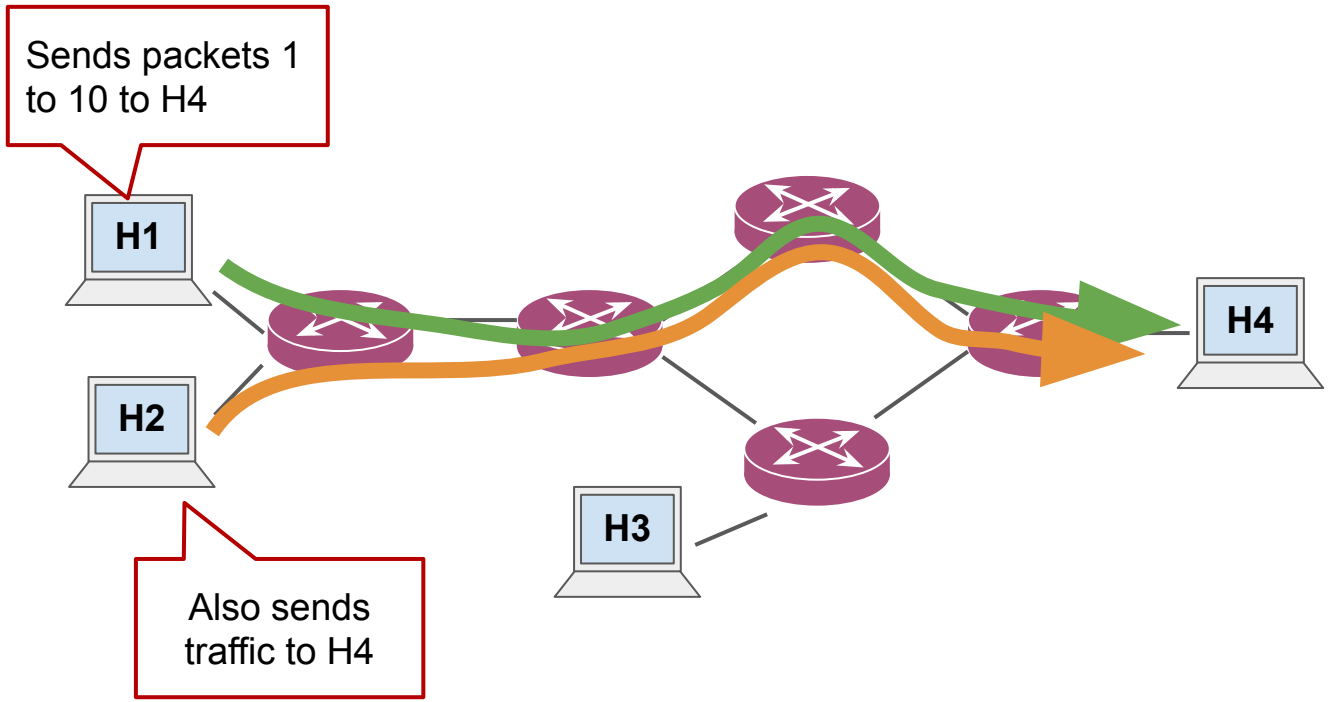
For every flow, at the sender

- Send some packets out.
- Use some signals to detect congestion in the network:
  - packets getting lost
  - packets taking longer to get to the receiver
  - network/receiver telling you it is congested
  - …
- Adjust sending rate accordingly.

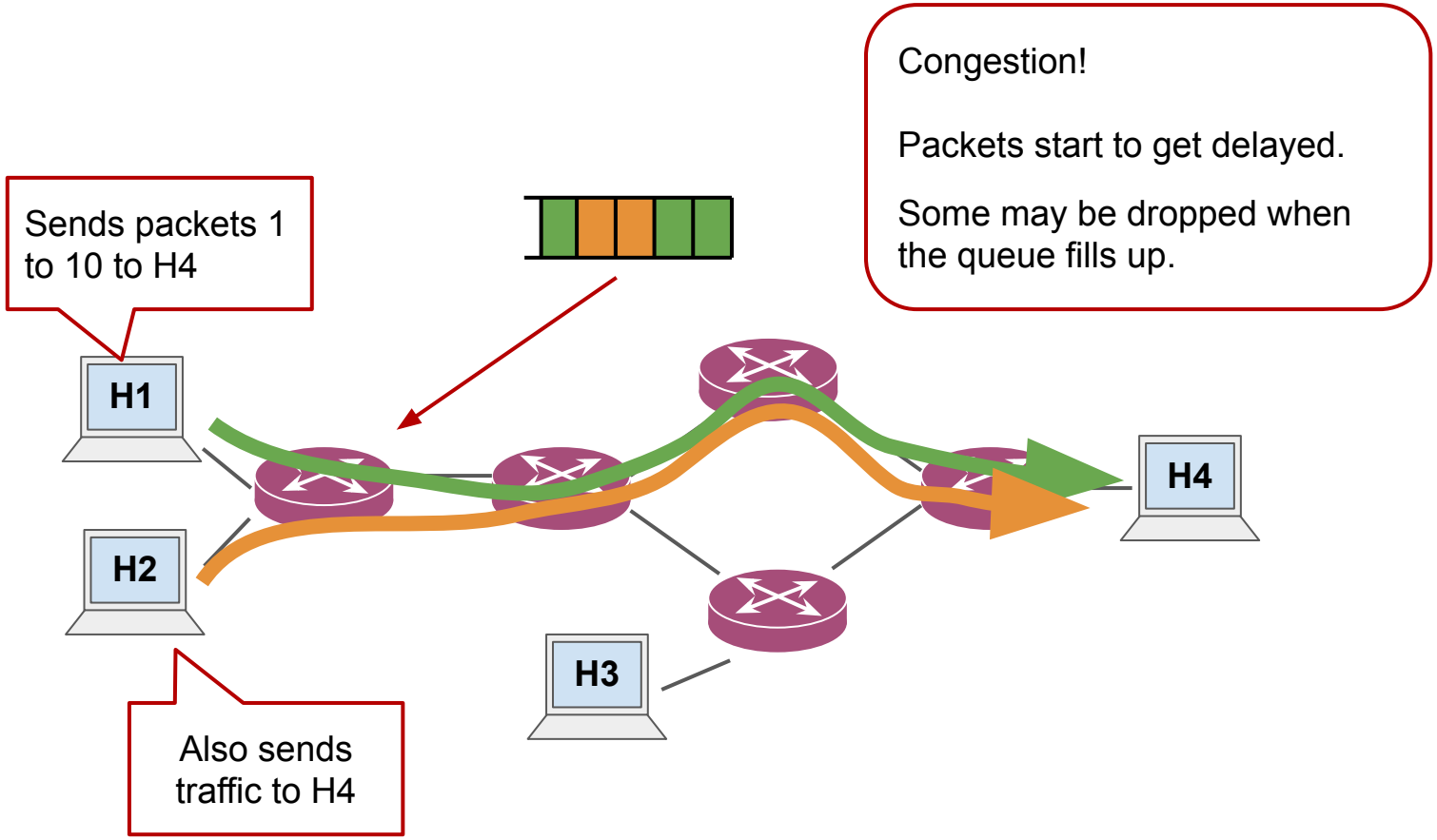
End-to-End Congestion Control



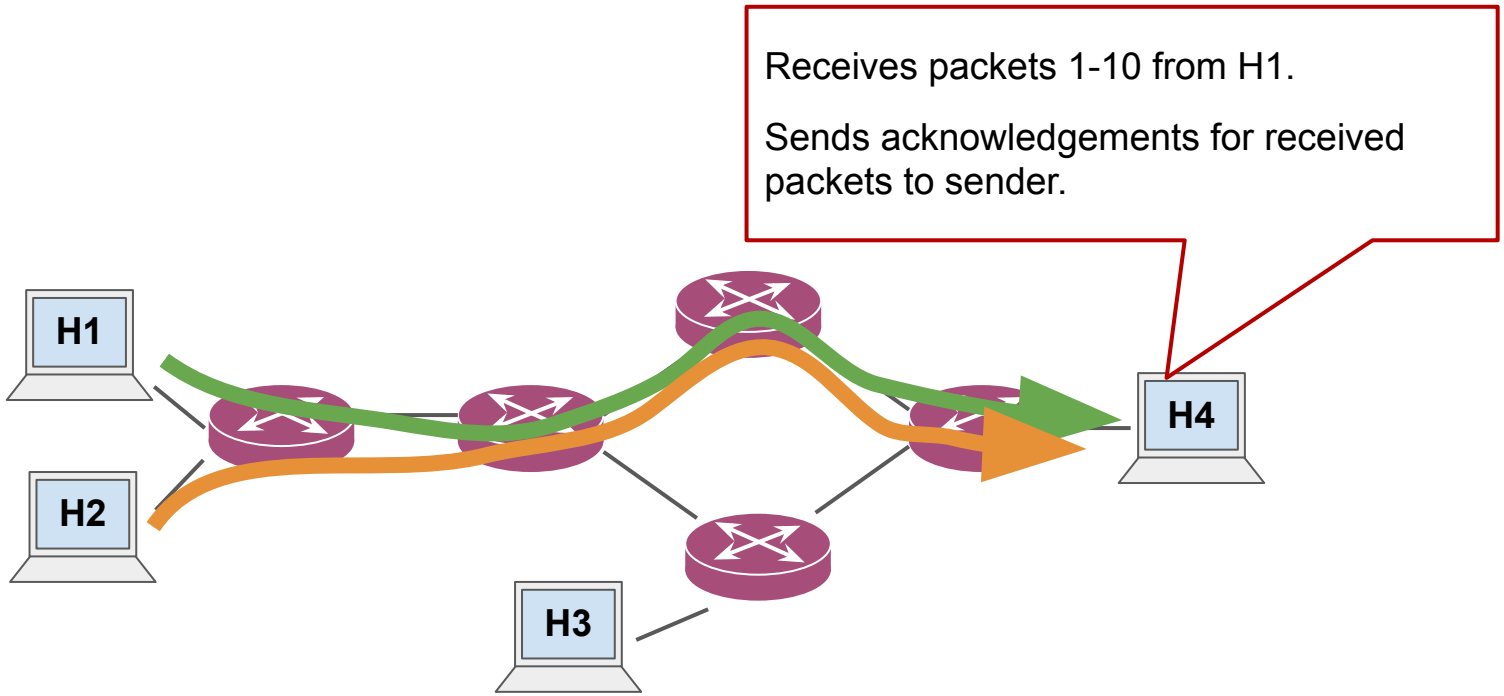
End-to-End Congestion Control



End-to-End Congestion Control



End-to-End Congestion Control



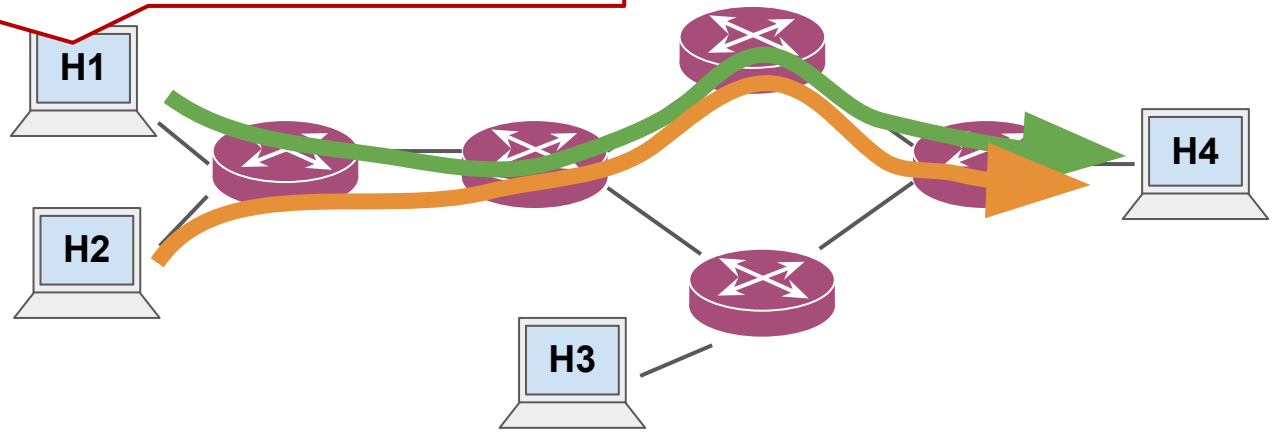
End-to-End Congestion Control

Receives acknowledgements for packets 1-10.

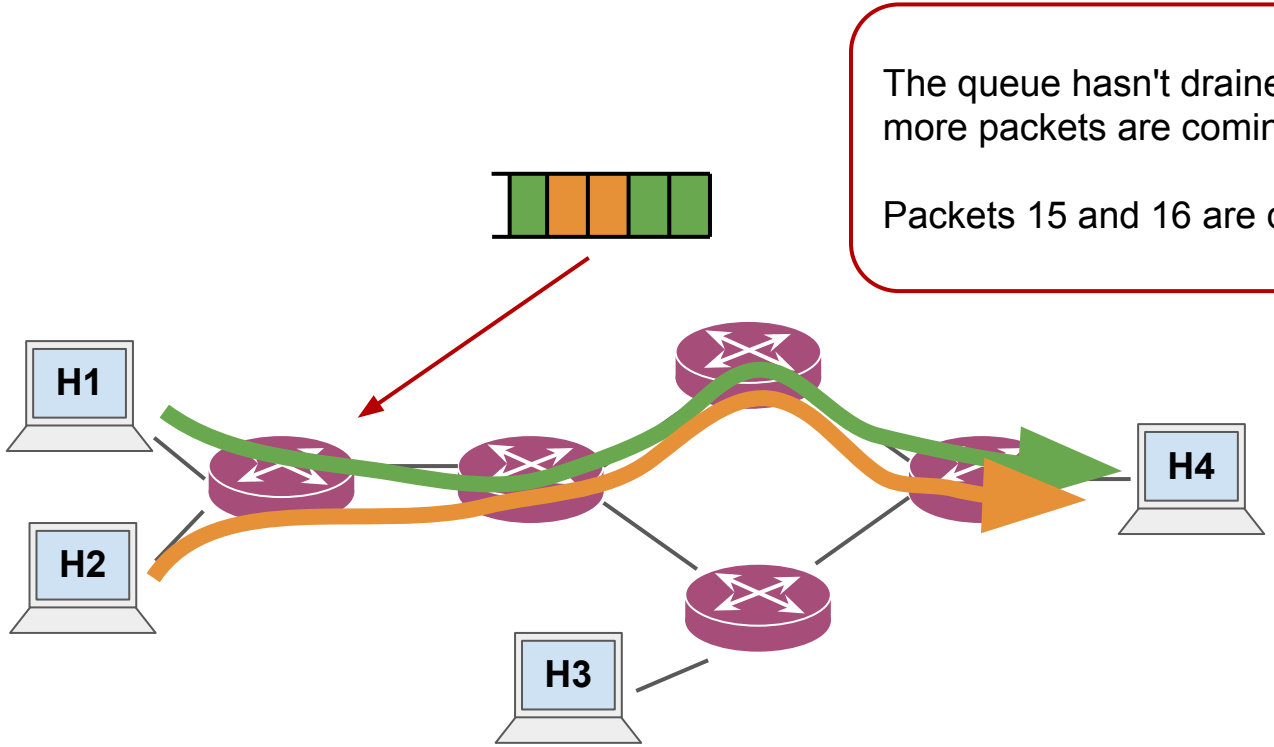
Notices packets 8-10 took longer to get acknowledged.

Maybe there is a minor congestion?

Next time, only sends 7 packets out.



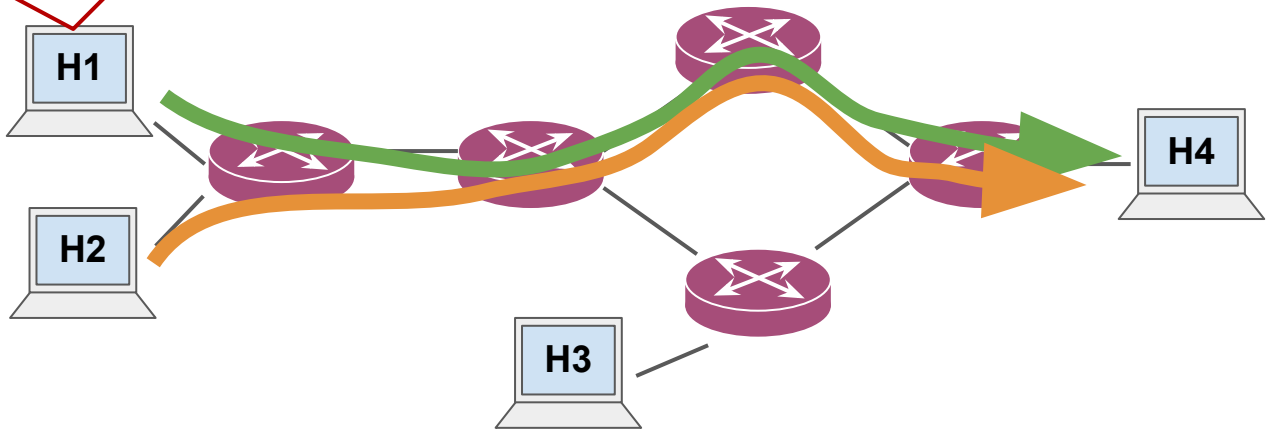
End-to-End Congestion Control



End-to-End Congestion Control

When it doesn't receive acks for packets 15 and 16, decides there is severe congestion going on.

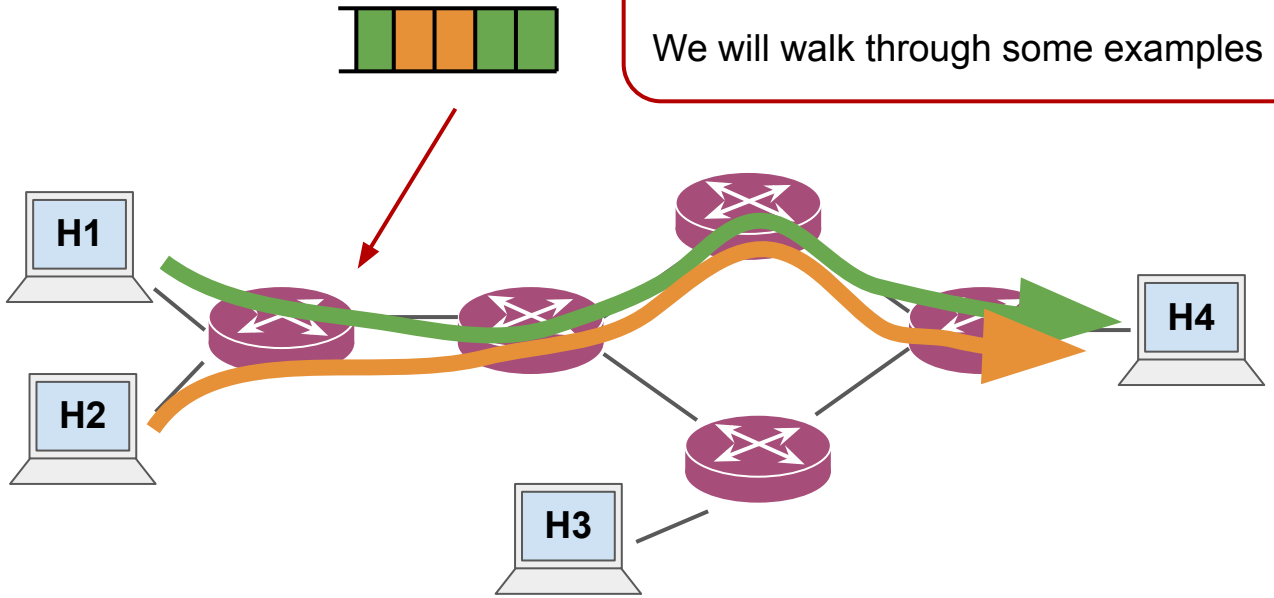
Only sends 2 packets out next time.



End-to-End Congestion Control

When the queue starts building up, the switch can signal the end-points to slow down.

We will walk through some examples later.

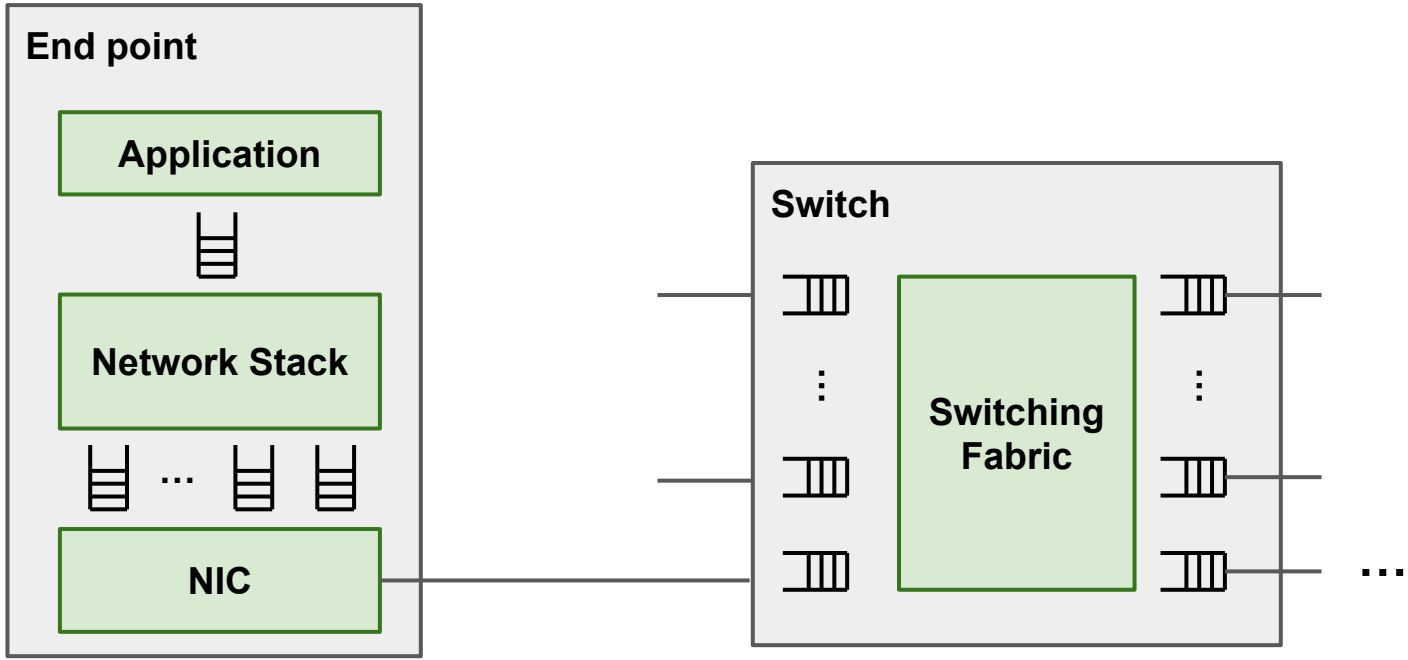


How does it affect resource sharing?

- The algorithm deciding when and how much to change the pace for each flow affects
  - how different flows in the network interact with each other, and
  - how they share the bottleneck.
- E.g., the flows that back off quickly and to larger extents can end up with a lower share of the bottleneck bandwidth.

Packet Scheduling

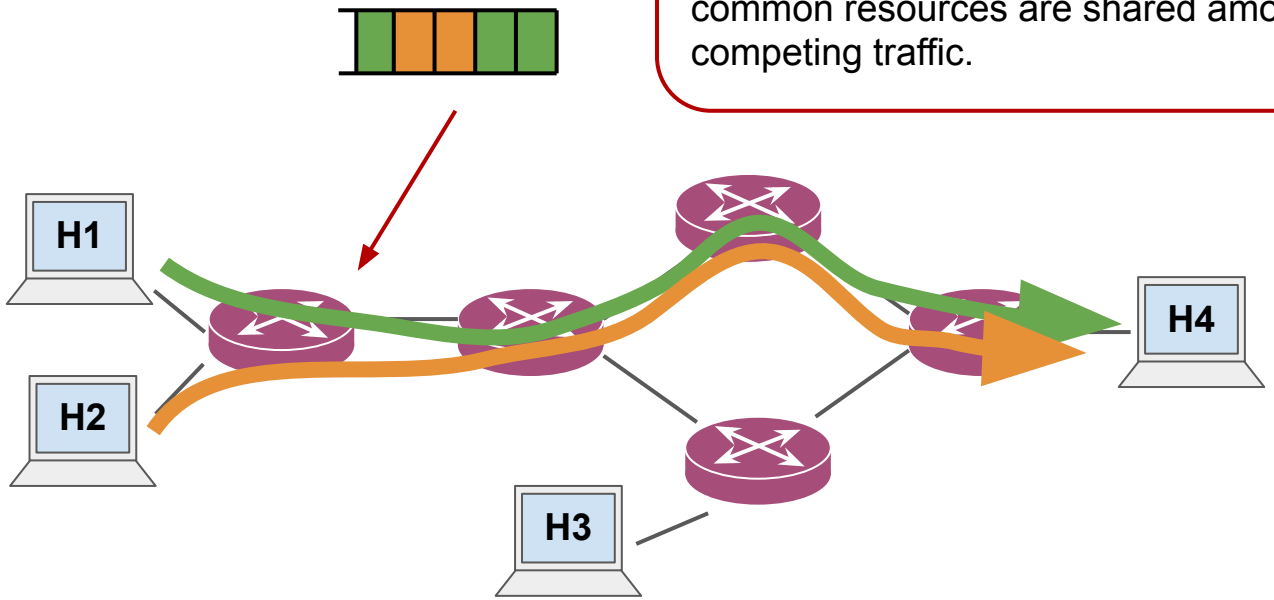
Queue are an integral part of networks



Packet Scheduling

Packet schedulers decide how to line up packets in queues and who gets to go next.

When there is contention, they can affect how common resources are shared among competing traffic.



Packet Scheduling

- The simplest "scheduler" is First In First Out (FIFO)
  - Packets are queued up in the order they arrive and exit in the same order
- There are many other, more complex, schedulers
  - A single priority queue (a packet's priority is decided on enqueue)
  - A set of FIFOs, each with its own priority
  - A set of FIFOs, serviced in round robin fashion
  - A FIFO, but packets can only be dequeued at a specific rate
  - A hierarchy of schedulers
  - …

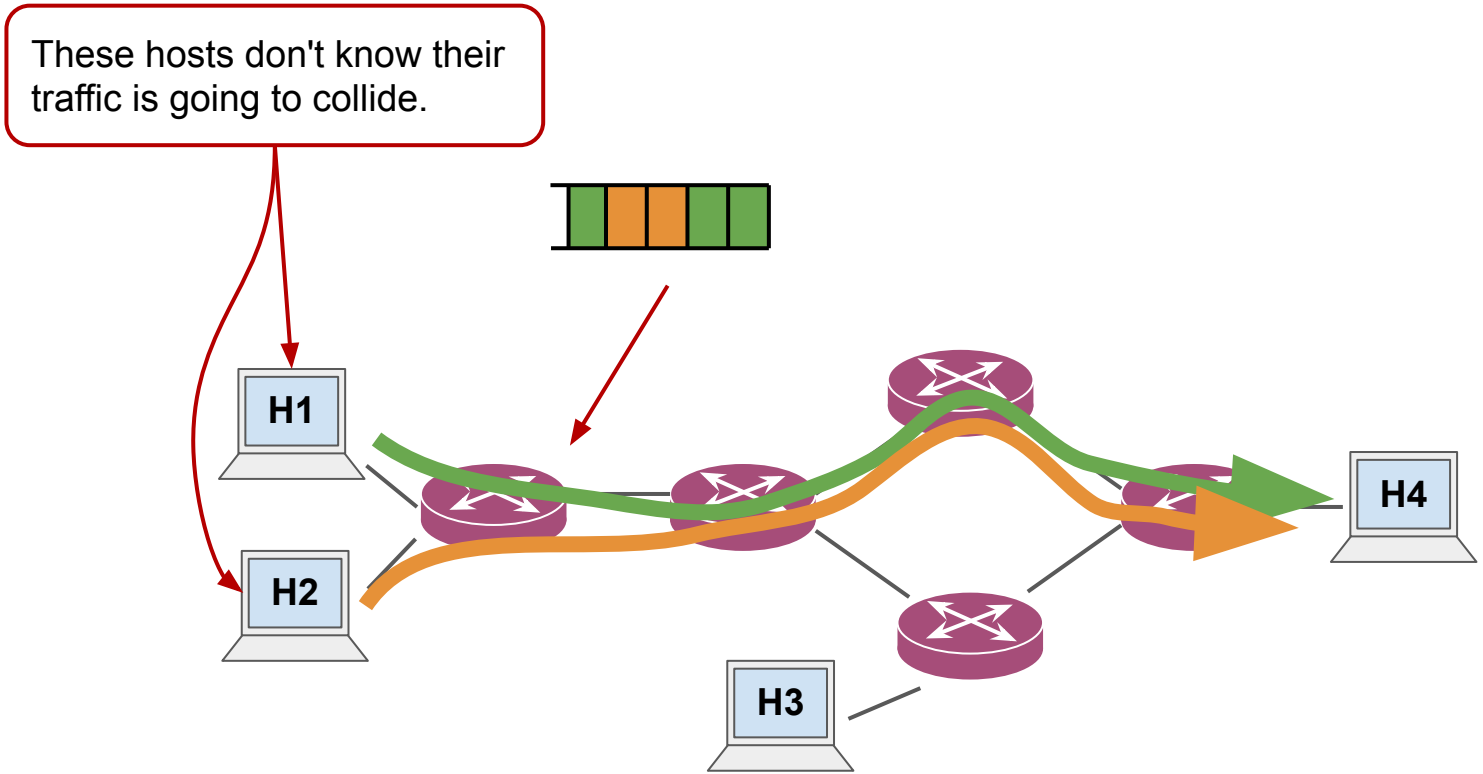
Active Queue Management (AQM)

- AQM algorithms manage the occupancy of a single queue
- They try to drop/mark packets before the queue is full
- With the goal of keeping the queue occupancy, and therefore, latency, within desirable bounds.
- Different algorithms have different ways of deciding when to start dropping/marking, whether to drop or mark, and which packets to drop/mark.

How are these mechanisms used in traditional networks?

- Most networks rely on end-to-end congestion control algorithms
- keeping the functionality in the network quite simple
  - No explicit signals or a simple fixed set of signals for end-to-end congestion control algorithms
  - A few FIFO queues and a few schedulers (e.g., priorities, DRR, etc.)
  - No AQM or a simple fixed set of AQM algorithms
- Why?
  - end-to-end principle
  - Keeping network devices simple and fast

But a little help from the network can go a long way

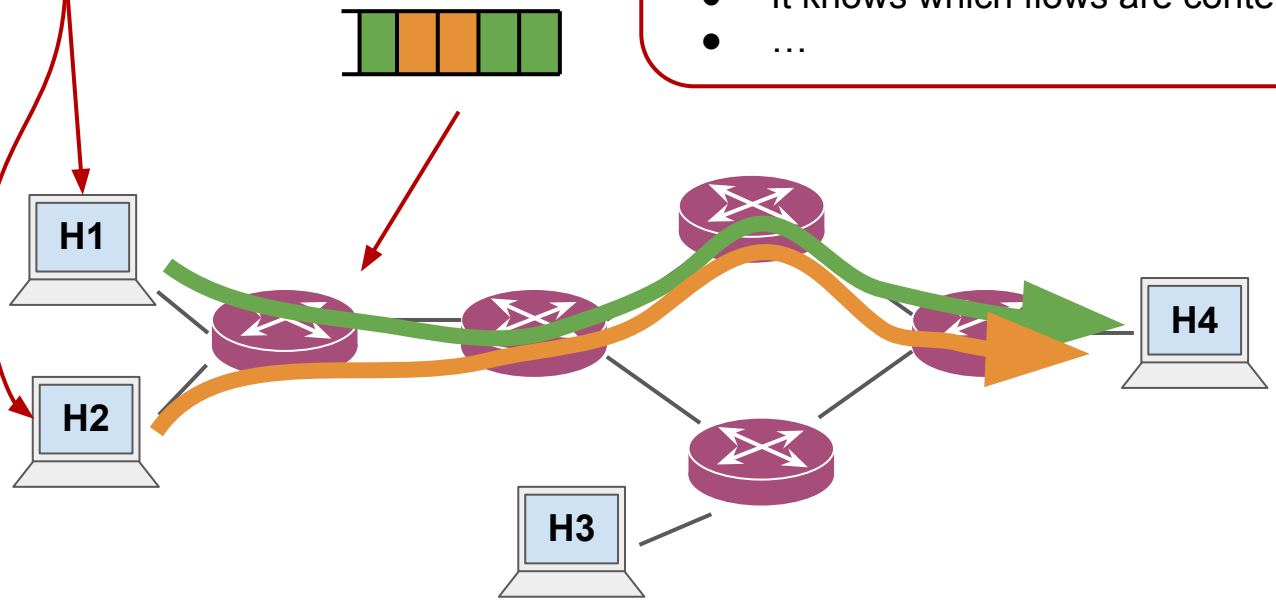


But a little help from the network can go a long way

These hosts don't know their traffic is going to collide.

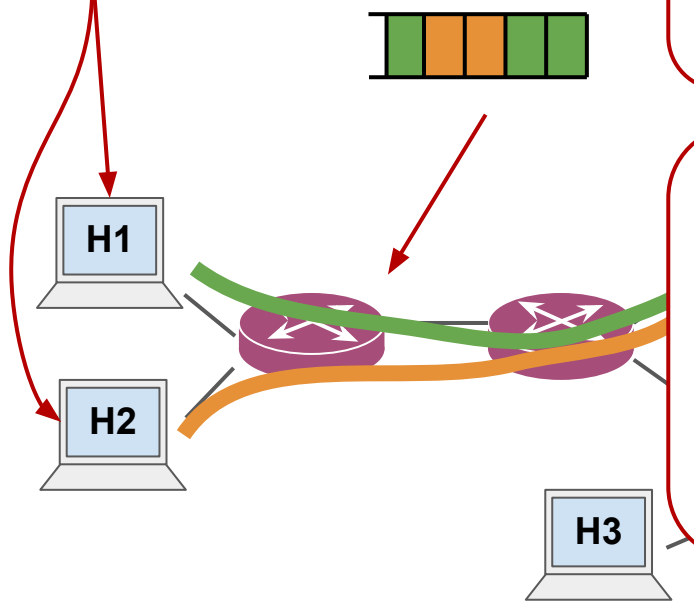
The switch knows a lot more about the contention

- It is where the contention is happening
- It sees the queue building up
- It knows which flows are contending
- …



But a little help from the network can go a long way

These hosts don't know their traffic is going to collide.



The switch knows a lot more about the contention

- It is where the contention is happening
- It sees the queue building up
- It knows which flows are contending
- …

In traditional networks, there are ways for switches to convey existence of congestion to end points by marking certain bits on packets.

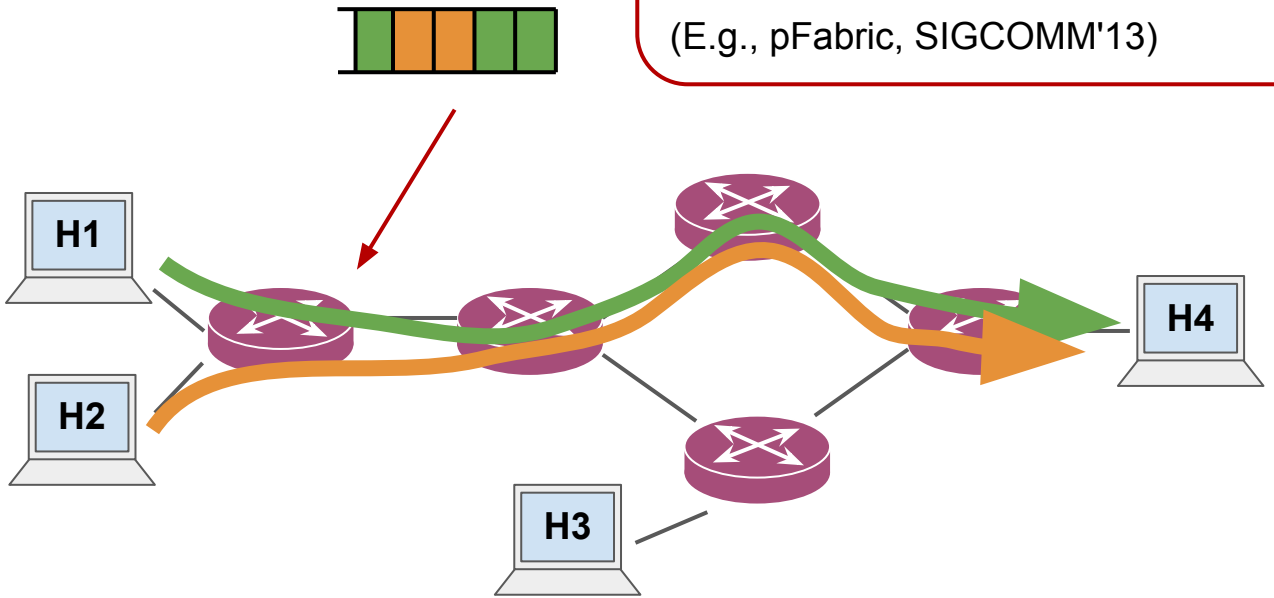
But this is more of an indirect signal

The sender has to infer all the other information about congestion (e.g., location, extent, etc.).

But a little help from the network can go a long way

Instead, a careful design of scheduling primitives in the network can significantly help with performance objectives such as minimizing flow completion times.

(E.g., pFabric, SIGCOMM'13)



We may need to do more in the network

- End-to-end congestion control relies on sending packets out, getting signals, and adjusting it sending strategy.
- It works best when flows can take a couple of round-trip-times (RTTs) to complete
    - They can send packets for a few rounds, and get an accurate picture of the available network capacity.
- However, this may not happen in certain networks

We may need to do more in the network

- For example, in data centers:
  - Link speeds are increasing
  - Flows are getting shorter
- Flows can take fewer RTTs (even just one or two) to complete.
- Not enough time for the end-to-end congestion control algorithms to "figure out" the right rate.

How has network programmability helped?

- Customizing the signals to e2e congestion control, scheduling, AQM, etc. to the each network and the requirements of its applications.
- Motivating new signaling, scheduling, AQM, etc. techniques
  - Implement it in a programmable switch
  - show it can run at line rate
  - show it provides significant benefits
  - so you can convince vendors to include it in their switches

How has network programmability helped?

- Better signals for congestion control algorithms
  - e.g., use INT to add information about queue lengths to the packets (HPCC, SIGCOMM 2019)
- More complex (and flexible) packet scheduling
  - e.g., fair queuing is hard to implement at line-rate but you can implement and approximation on programmable switches (AFQ, NSDI'18).
  - a programmable hardware architecture for packet scheduling, so we can configure the switch for different scheduling algorithms (PIFO, SIGCOMM'16)

How has network programmability helped?

- Targeted fine-grained measurements
  - can help provide better signals to congestion control algorithms
  - can help create more effective AQM schemes
  - e.g., if we could detect which flow(s) contribute most to the queue build up, we can mark/drop those packets in our AQM scheme (Conquest, CoNEXT'19)

Paper 1: HPCC: High Precision Congestion Control

- Uses INT to provide more accurate signals to congestion control algorithms
- Demonstrates the benefit in low-latency high-bandwidth RDMA networks.

Paper 2: Approximating Fair Queueing on Reconfigurable Switches

- Fair queuing ensure all flows sharing a link get a fair share of the bandwidth
    - providing the illusion that each flow has its own separate queue with a "round-robin" service across queues
- It has been deemed too complex to implement in switches.
- This paper shows that an approximate version can be implemented in programmable switches.

Additional Resources

- Back-pressure Flow Control (BFC) (NSDI'22)
- Programmable packet scheduling at line-rate (SIGCOMM'16)
- Loom: Flexible and Efficient NIC Packet Scheduling (NSDI'19)
- Fine-Grained Queue Measurement in the Data Plane (CoNEXT'19)
- ABM: Active Buffer Management in Datacenters (SIGCOMM'22)