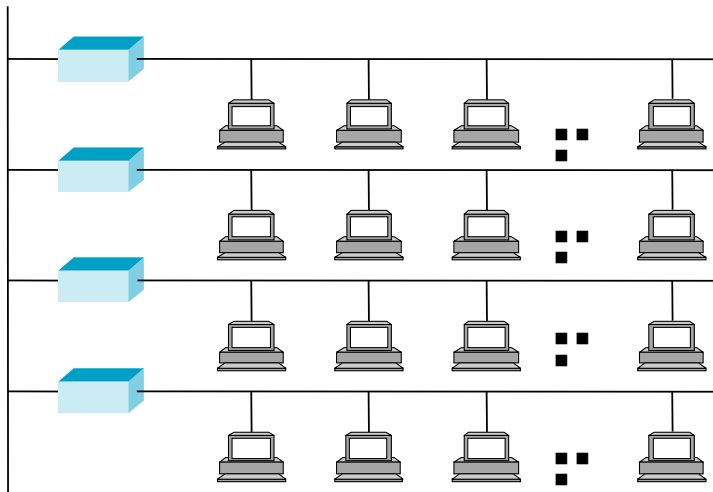


CSC 525: Computer Networks

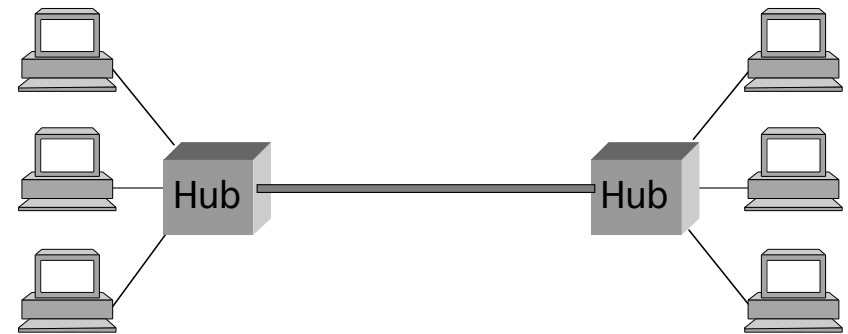
Ethernet

- The most popular LAN technology
 - developed by Xerox PARC in mid-1970s
 - roots in Aloha packet-radio network (shared wireless medium)
 - 10Mbps standard by Xerox, DEC, and Intel in 1978
 - Evolved over years, from 10Mbps to 10Gbps, from bus topology to star



 Repeater

 Host



CSMA/CD

- Collision domain
 - Frames propagate to everywhere in a collision domain, meaning if two hosts are sending at the same time, their frames will collide on the wire.
 - How do hosts in the same collision domain share the medium?
- CSMA: carrier sense multiple access
 - Listen before transmit
 - If channel is idle: transmit
 - If channel is busy, wait until idle.
 - Collisions still possible:
 - Multiple nodes may send simultaneously
- CD: collision detection
 - compare transmitted signals with received, if not same, abort transmission and wait and try again.
 - How long to wait? Exponential backoff.

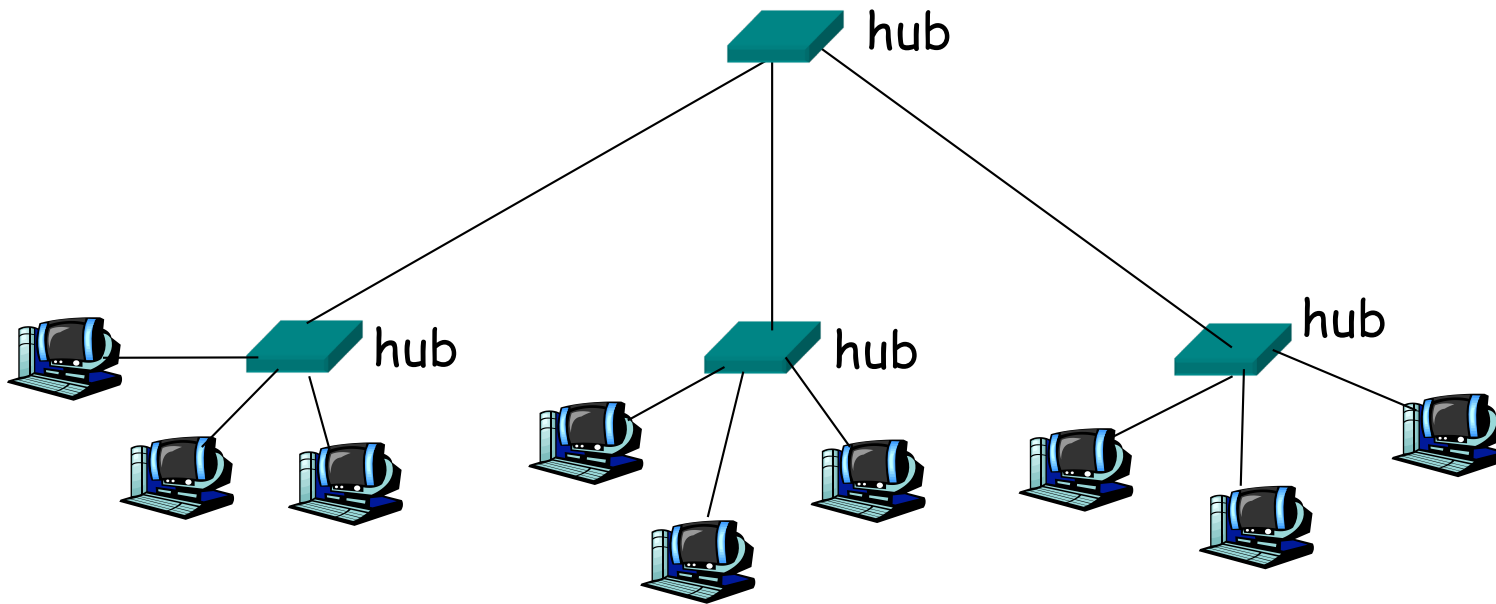
Why is Ethernet successful?

- Simplicity:
 - Easy to administer and maintain
 - Inexpensive
- Performance?
 - Good under light traffic load ($< 30\%$)
 - Bad under heavy traffic load (too much collision)
 - Get improved over time, basically by moving from sharing a link to switching among dedicated links.
- How to scale Ethernet?

Ethernet Hubs

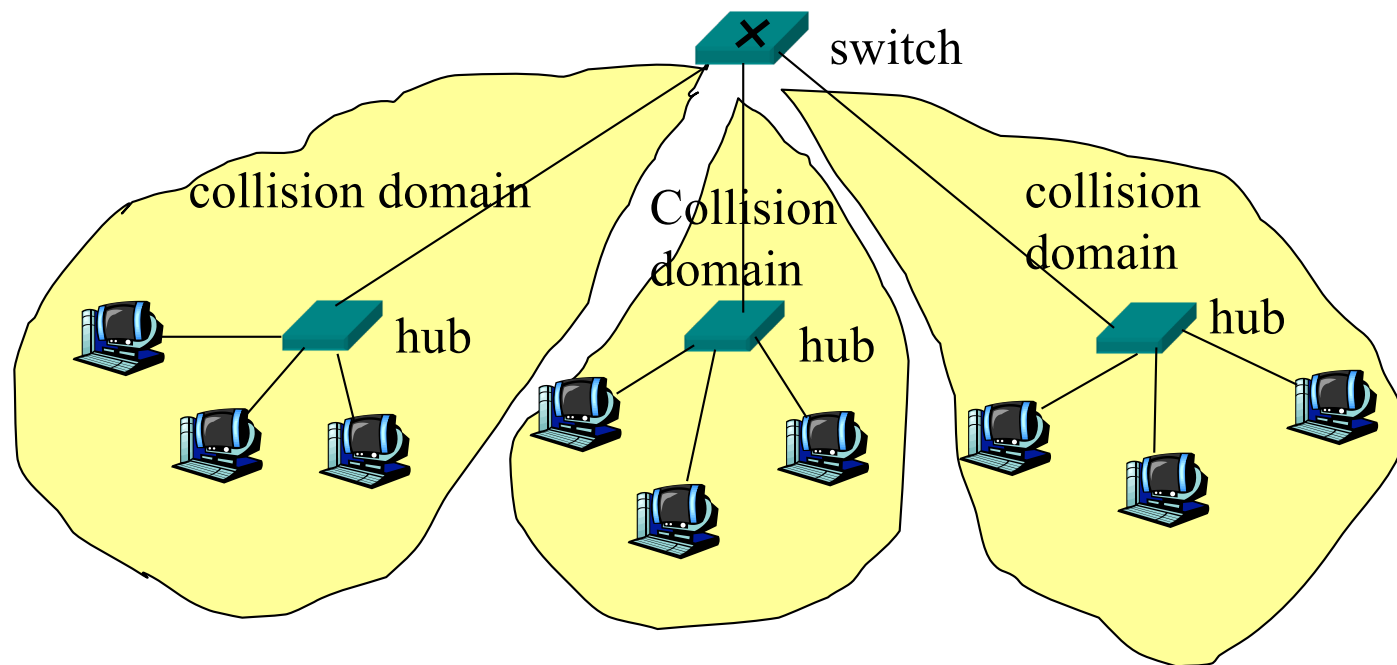
Hub: *physical layer* repeaters

- repeat bits received on one interface to all other interfaces
- All nodes are still in the same collision domain.
- More nodes lead to poorer performance due to increasing collisions.



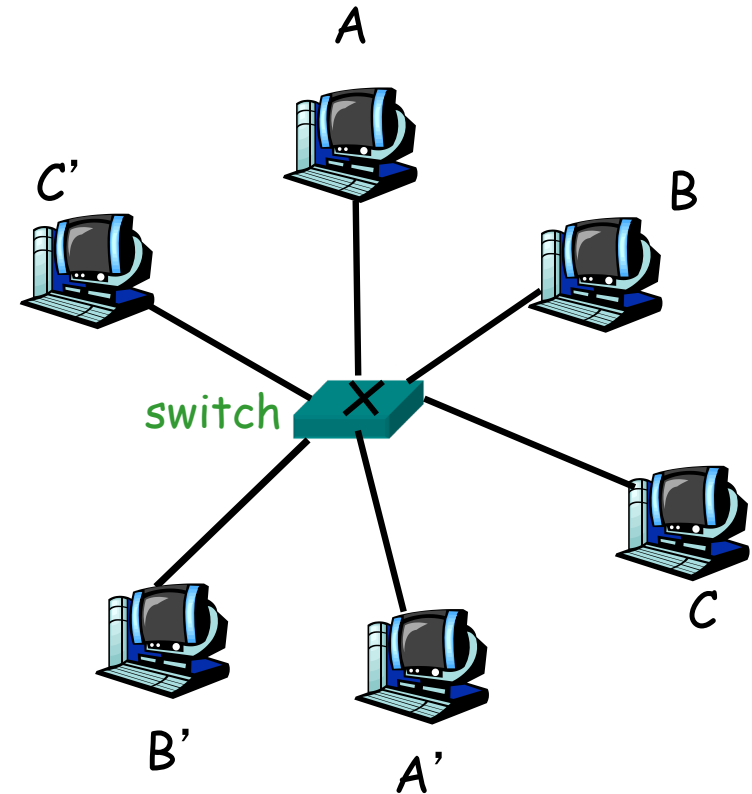
Ethernet Switch

- A switch will forward traffic only to the port that will lead to destination, not all the ports.
- It thus separates the network into multiple collision domains.
- In the case that all hubs are replaced by switches, each collision domain will have only one end-host.



Dedicated Access

- Switch with many interfaces; Hosts have direct connection to switch
- No collisions; full duplex.
- **Switching:** A-to-A' and B-to-B' simultaneously, no collisions, get full link bandwidth.

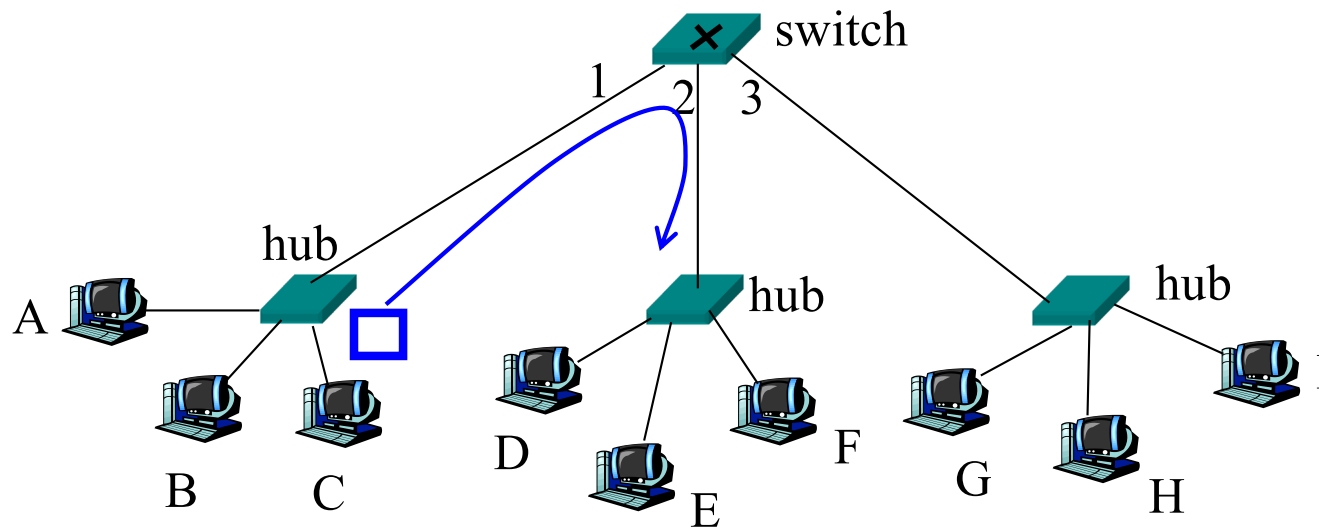


Self Learning


- How does a switch know to which port a packet should be forwarded?
- A switch has a **switch table**
 - Entry: (MAC Address, outgoing Interface, Time Stamp)
 - stale entries will time out (usually in minutes)
- Switch *learns* which hosts can be reached through which interfaces
 - when a frame is received, the switch learns that the sender host can be reached via the incoming interface of this frame.
 - records sender/location pair in switch table
 - The first frame is flooded.

Switch Example


Suppose C sends a data frame  to D

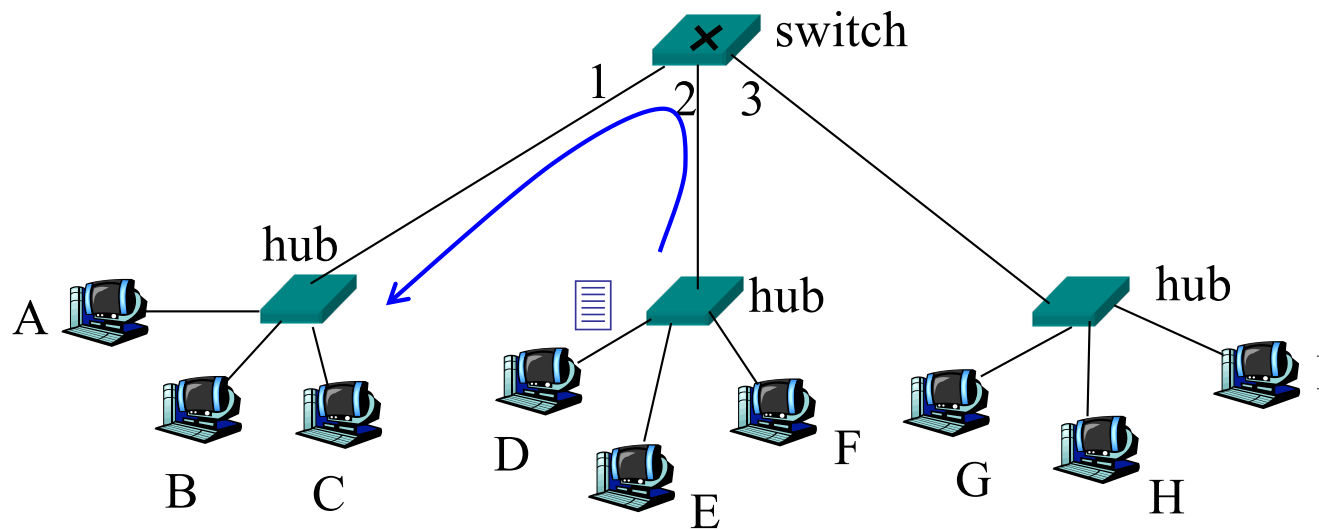


address	interface
A	1
B	1
E	2
G	3
C	1

- Switch receives the frame from C
 - Add to forwarding table: C is on interface 1
 - D is not in table: forwards  to interfaces 2 and 3
- Frame received by D


Switch Example

Suppose D replies a data frame  to D



address	interface
A	1
B	1
E	2
G	3
C	1
D	2

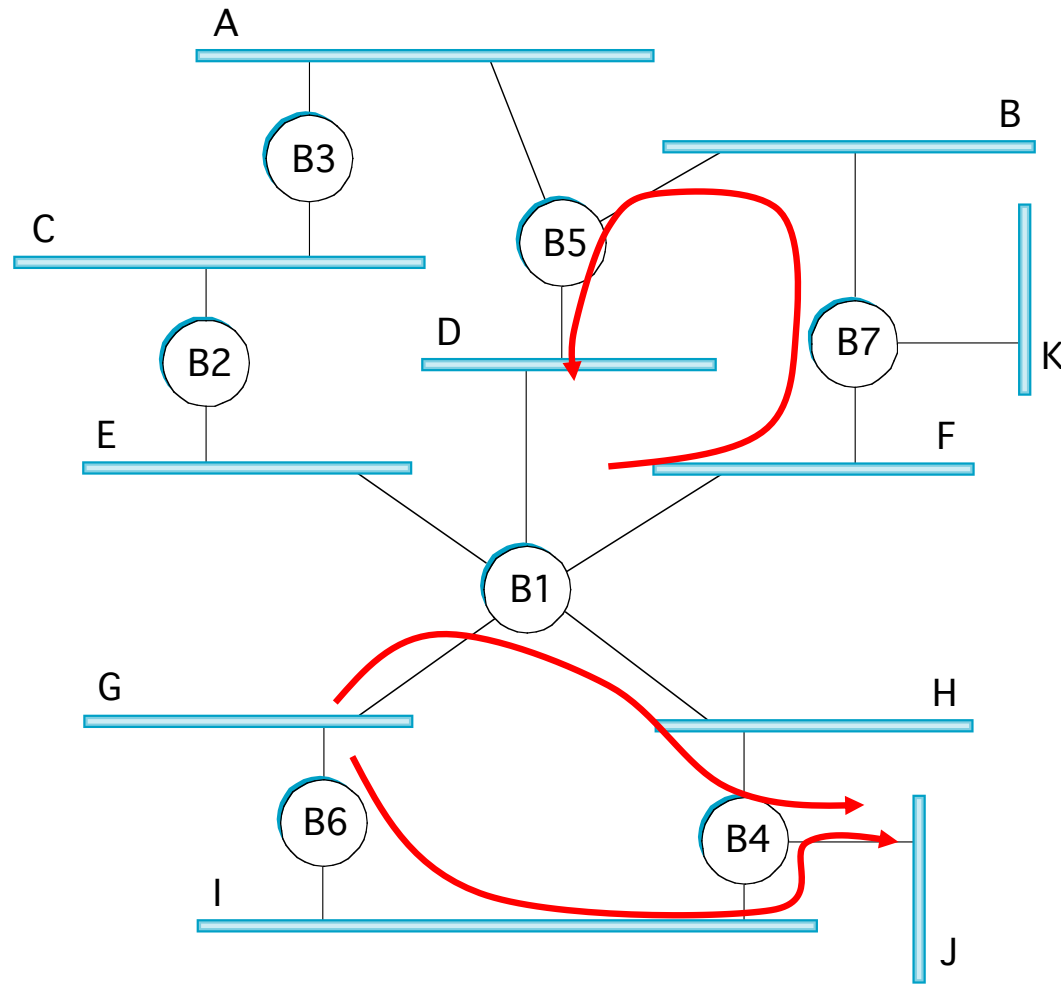
When D replies back with a frame  to C

- Add to forwarding table: D is on interface 2
- Forward  to C on interface 1

Pros of self-learning switches

- Plug-and-play
 - no configuration needed
- Host mobility
 - Hosts can move to another port in the same Ethernet and self-learning switches will be able to update its switch table automatically after a small timeout period.
 - Hosts don't need to change addresses.

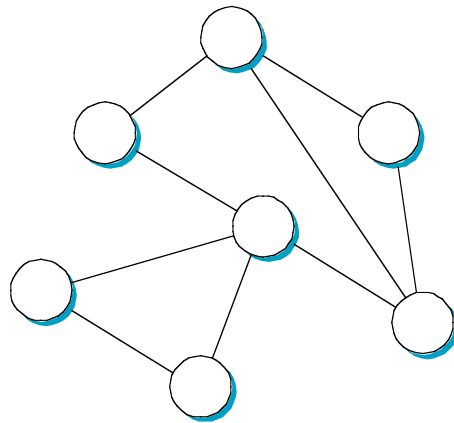
Forwarding Loops



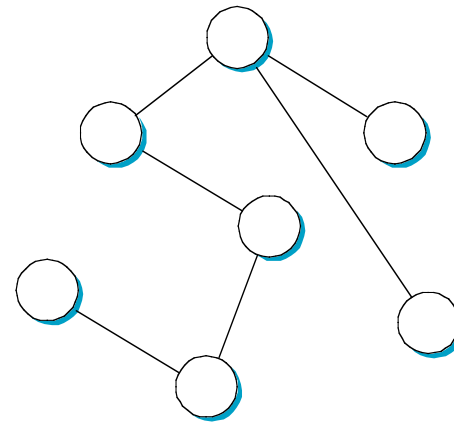
Flooding causes duplicates and loops.

Spanning Tree Protocol

- Trim the topology into a spanning tree, then run the flood-based self-learning.
- Switches run a distributed spanning tree protocol
 - select which switch actively forward to which port to ensure loop freedom.
 - developed by Radia Perlman



(a)



(b)

An “*All Ethernet*” Enterprise Network?

- “*All Ethernet*” makes network management easier
 - Zero-configuration of end-hosts and network
 - Flat addressing
 - Self-learning
 - Location independent and permanent addresses also simplify
 - Host mobility
 - Network troubleshooting
 - Access-control policies

But, Ethernet does not scale

- Flooding-based delivery

- Frames to unknown destinations are flooded



- Broadcasting for ARP, DHCP

- Bootstrapping relies on broadcasting
- Vulnerable to resource exhaustion attacks



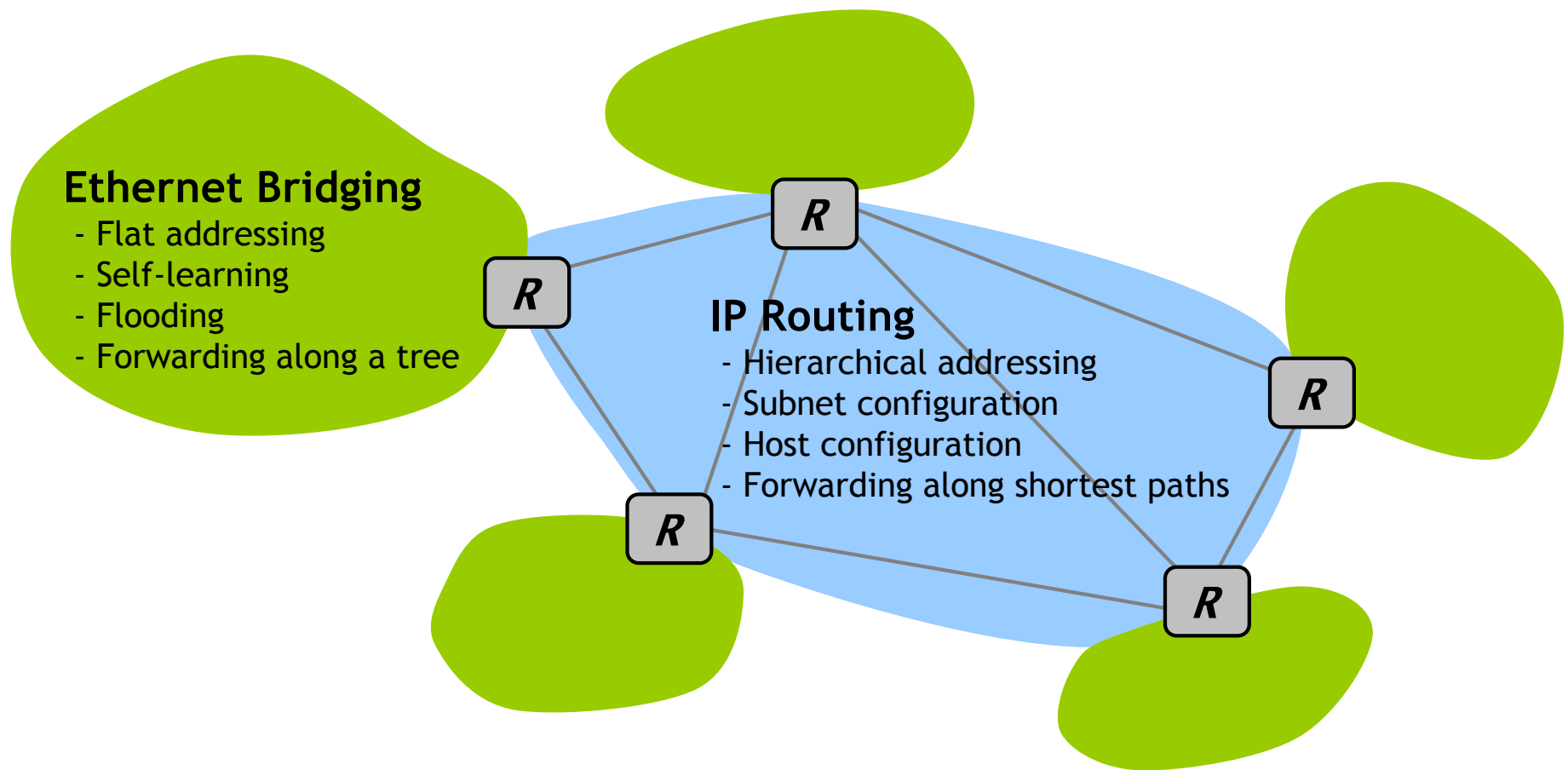
- Inefficient forwarding paths

- Loops are fatal due to broadcast storms; use the STP
- Forwarding along a single tree leads to inefficiency; cannot make use of redundant links.



State of the Practice

Enterprise networks comprised of **Ethernet-based IP subnets** interconnected by IP routers.



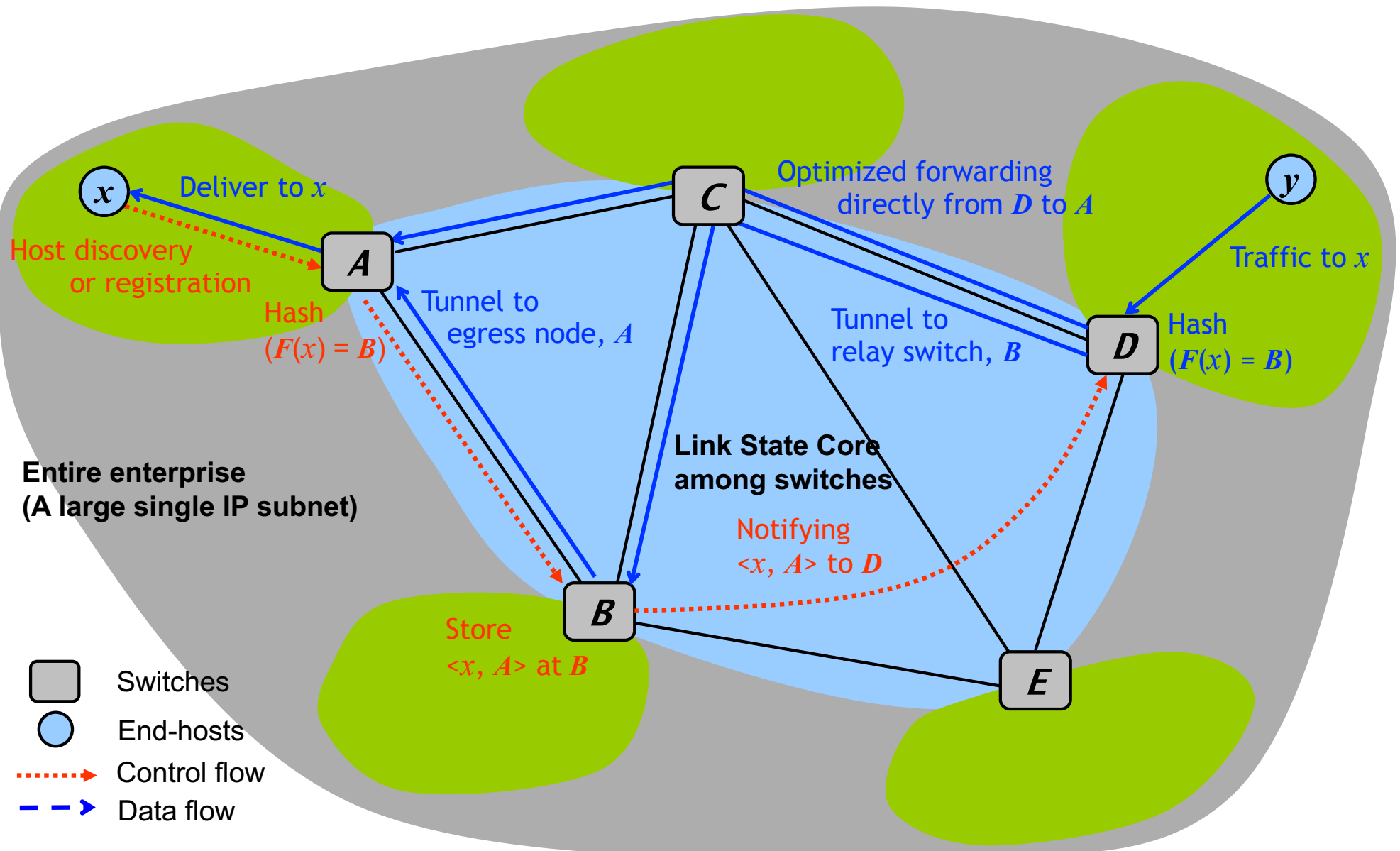
Can we get the best of both?

- Objectives
 - Avoiding flooding of data packets
 - Restraining broadcasting
 - Keeping forwarding tables small
 - Ensuring path efficiency
 - Support for host mobility
 - Plug-and-play
 - Backward compatible to hosts.

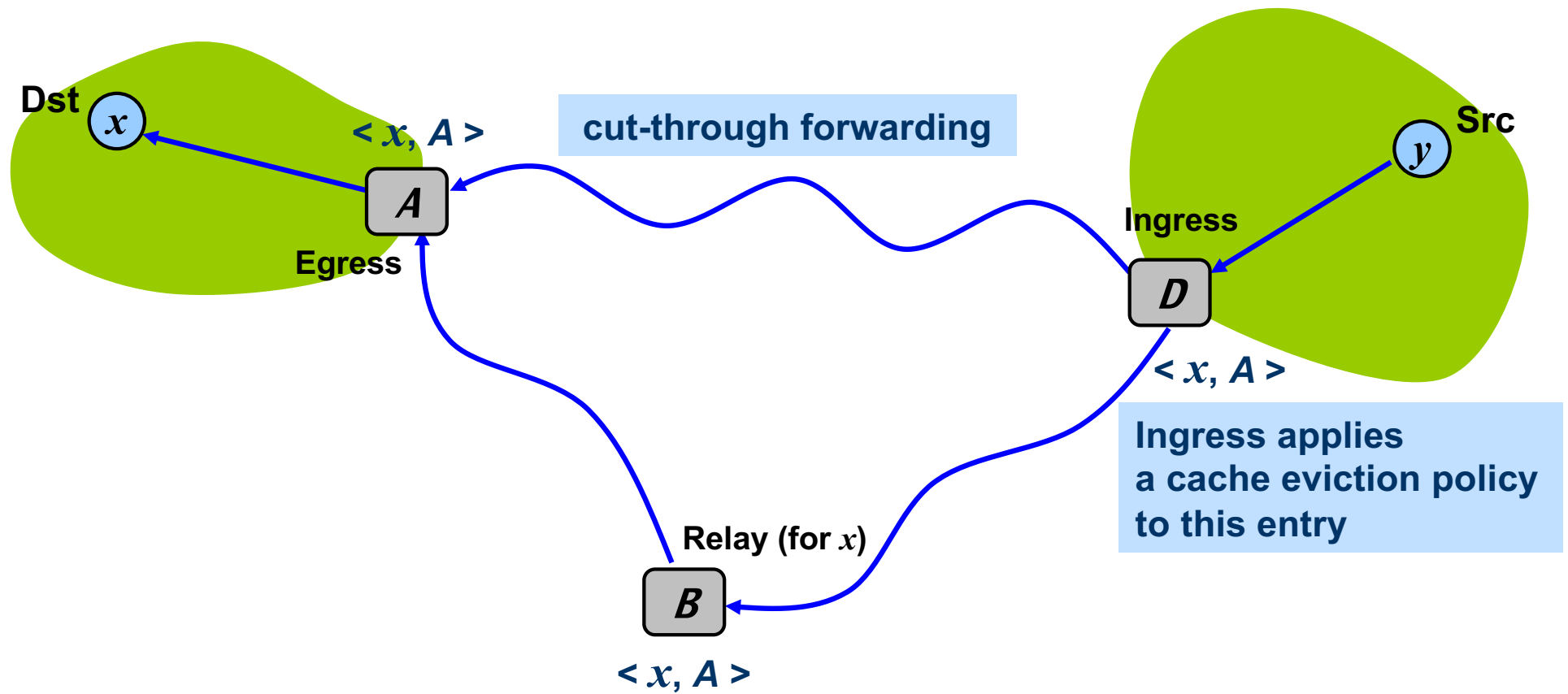
Seattle in a nutshell

- **Shortest-path forwarding between switches**
 - Switches run link-state routing with only their own connectivity info (not including subnets of end hosts)
 - Ensures data-plane efficiency
- **Flat addressing of end-hosts**
 - Switches use hosts' MAC addresses for routing
 - Ensures zero-configuration and backward-compatibility
- **Automated host discovery at the edge**
 - Switches detect the arrival/departure of hosts
 - Obviates flooding and ensures scalability
- **Hash-based on-demand resolution**
 - Hash deterministically maps a host to a switch
 - Switches resolve end-hosts' location and address via hashing
 - Ensures scalability

How does it work?



Terminology



Unicast-based Bootstrapping

- ARP and DHCP
 - Ethernet: Broadcast requests
 - SEATTLE: Hash-based on-demand address resolution
 - Exactly the same mechanism as location resolution
 - Proxy resolution by ingress switches via unicasting

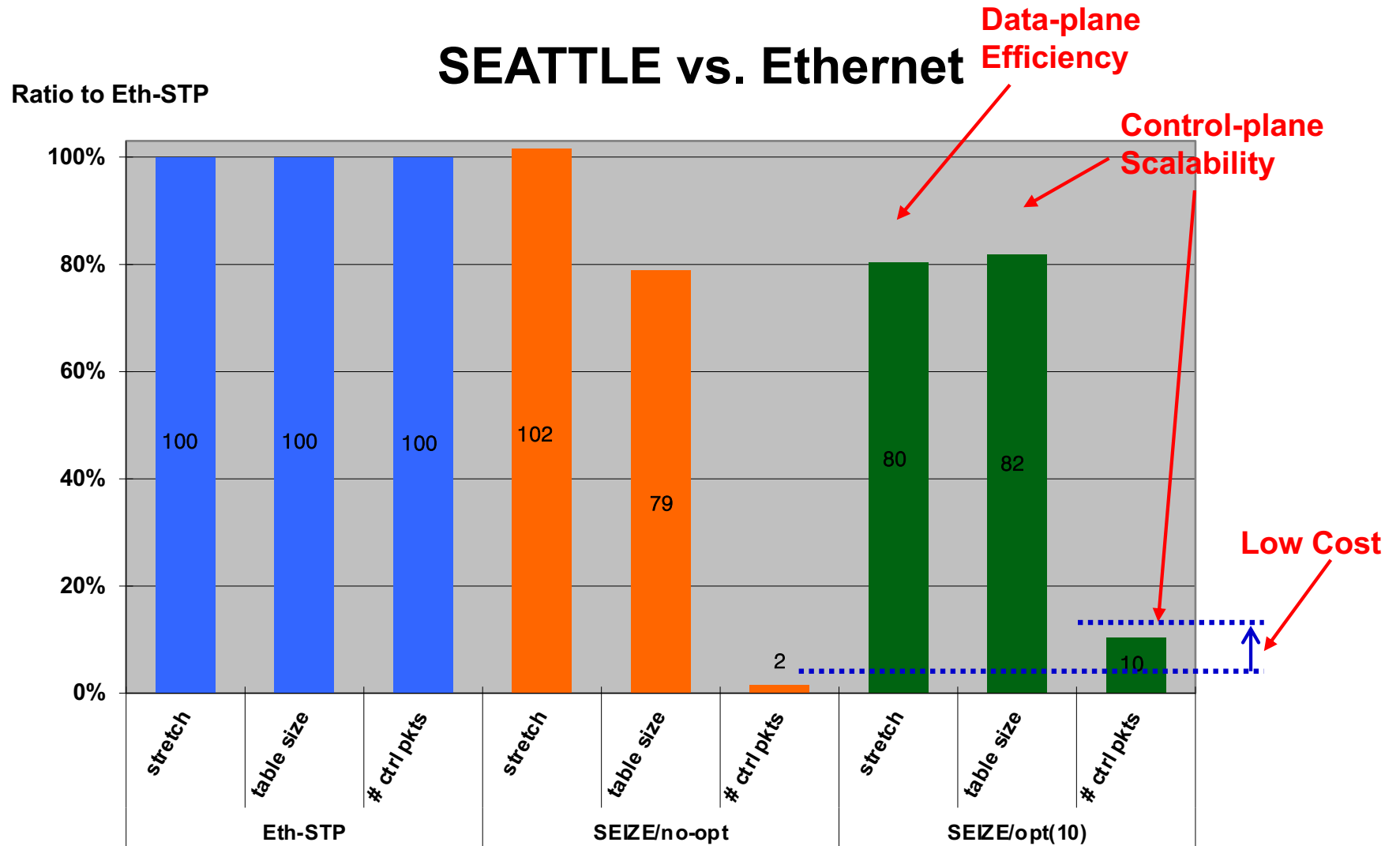
Control-Plane Scalability

- Minimal overhead for disseminating host-location information
 - Each host's location is advertised to only two switches
- Small forwarding tables
 - The number of host information entries over all switches leads to $O(H)$, not $O(S*H)$
- Simple and robust mobility support
 - When a host moves, updating only its relay suffices
 - No forwarding loop created since update is atomic

Data-Plane Efficiency

- Price for path optimization
 - Additional control messages for on-demand resolution
 - Larger forwarding tables
 - Control overhead for updating stale info of mobile hosts
- The gain is much bigger than the cost
 - Because most hosts maintain a small, static communities of interest (COIs)
 - Classical analogy: COI \leftrightarrow Working Set (WS);
Caching is effective when a WS is small and static

Overall Comparison



Conclusions

- SEATTLE is a plug-and-play enterprise architecture ensuring both scalability and efficiency
 - Link state routing within switches
 - plus one-hop DHT for lookup.
- Enabling design choices
 - Hash-based location management
 - Reactive location resolution and caching
 - Shortest-path forwarding