CS168
Designing the Internet

Rishabh Iyer

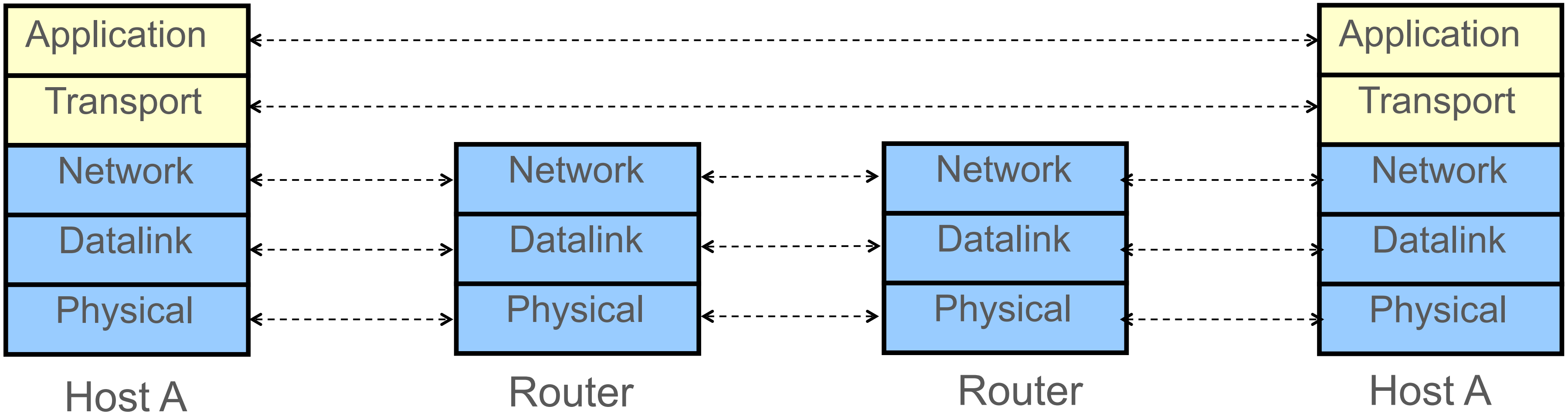
Spring 2026

Slide credits: Sylvia Ratnasamy, Rob Shakir, Peyrin Kao

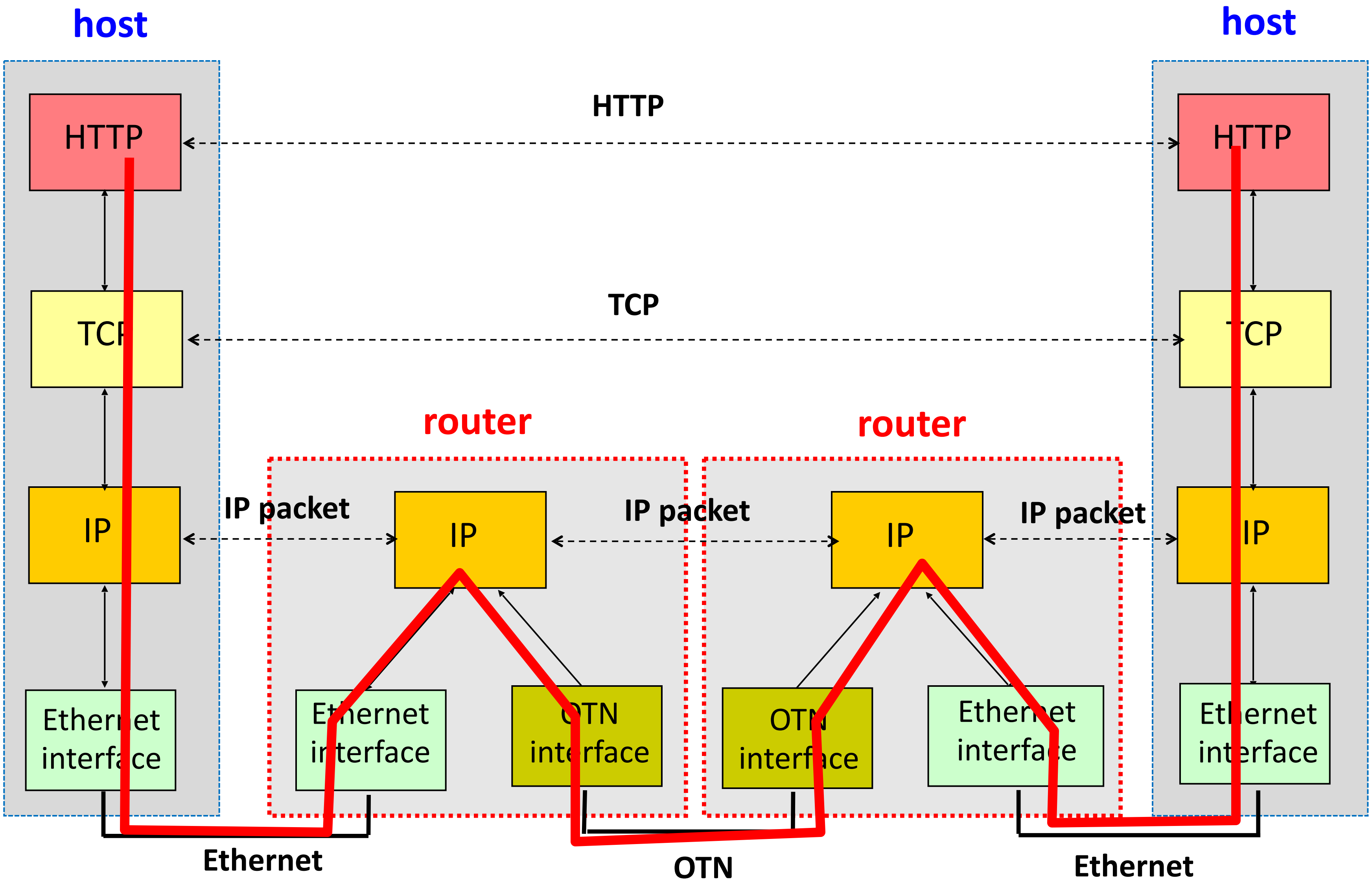
Outline

- Review: layering and the Internet
- The end-to-end argument

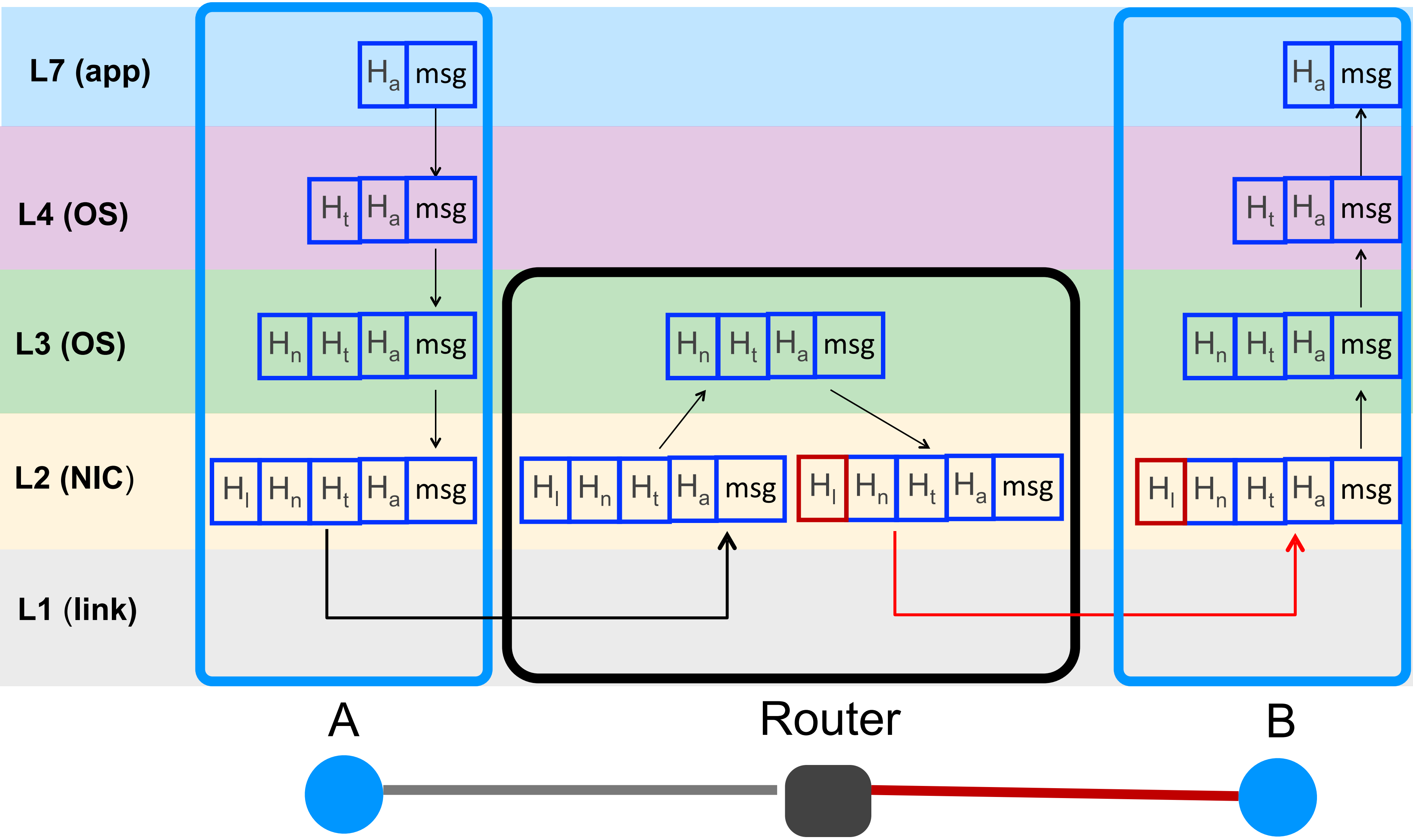
Recall: The Internet's layered architecture



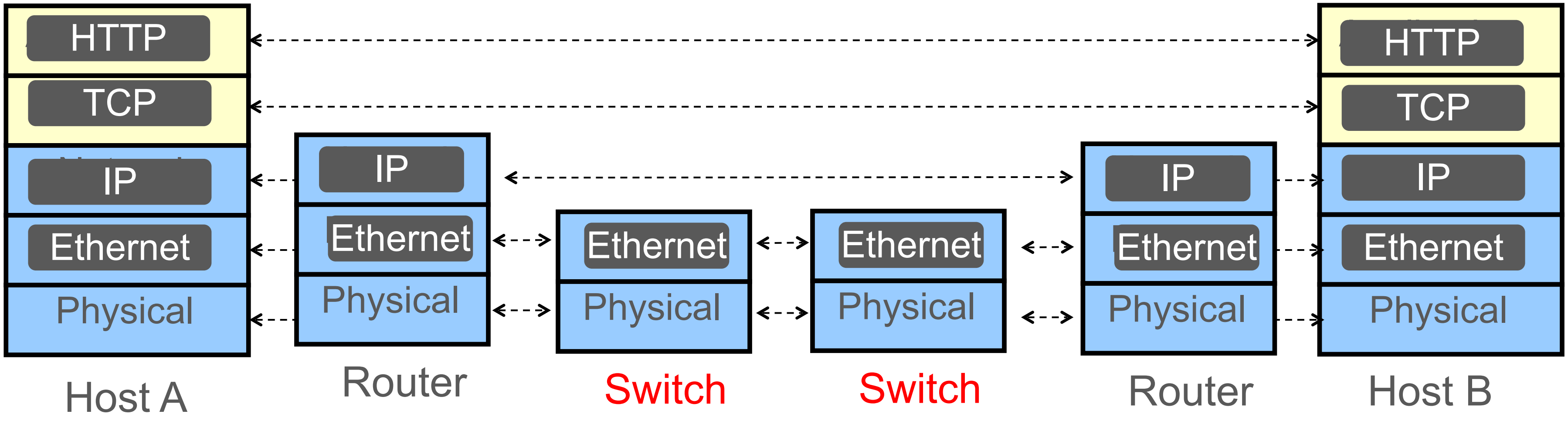
Recall: Example



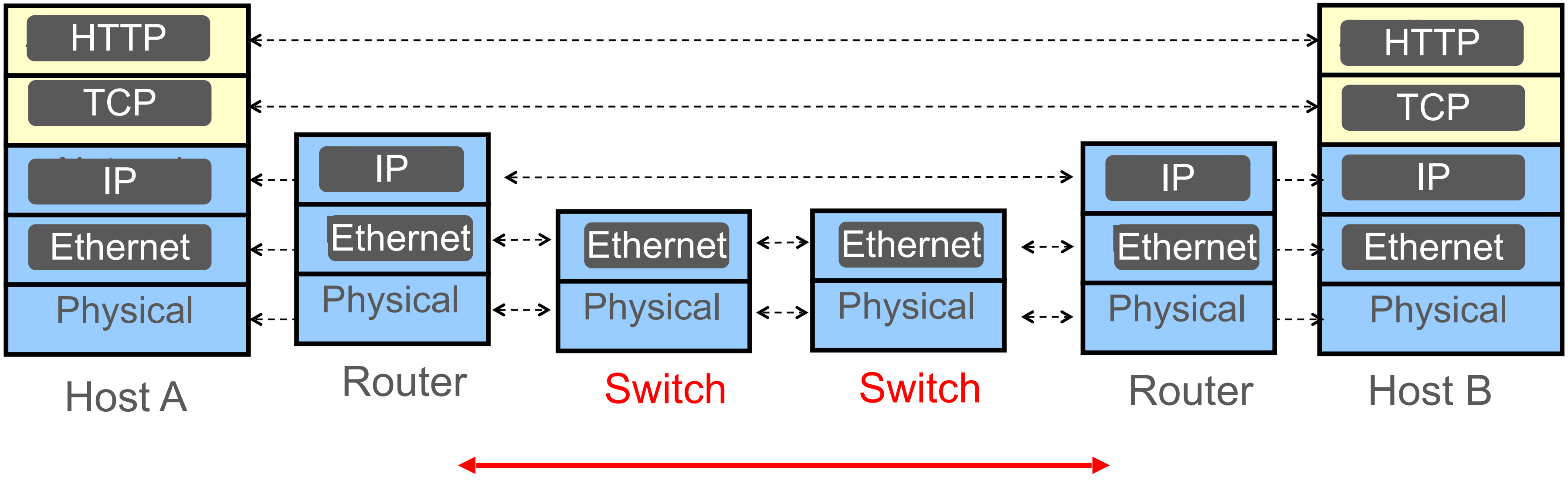
Recall: Adding/removing headers from a packet as it traverses layers



Where we left off ...



Where we left off ...



Packet traverses a local Ethernet network;
(i.e., based on its L2 header)

Local vs. Global networking

Local vs. Global networking

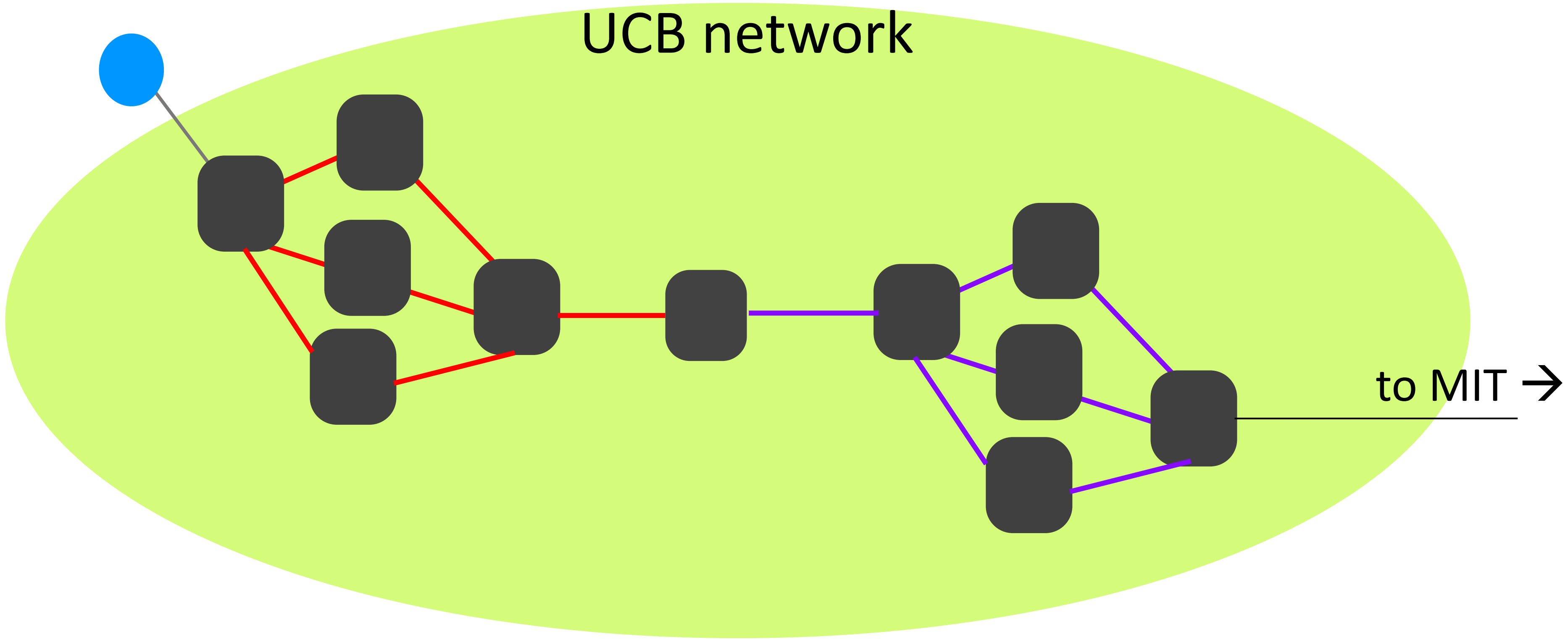
- Recall:
  - Local → relying on L2 headers (and local addresses)
  - Global → relying on L3 headers (and global addresses)

Local vs. Global networking

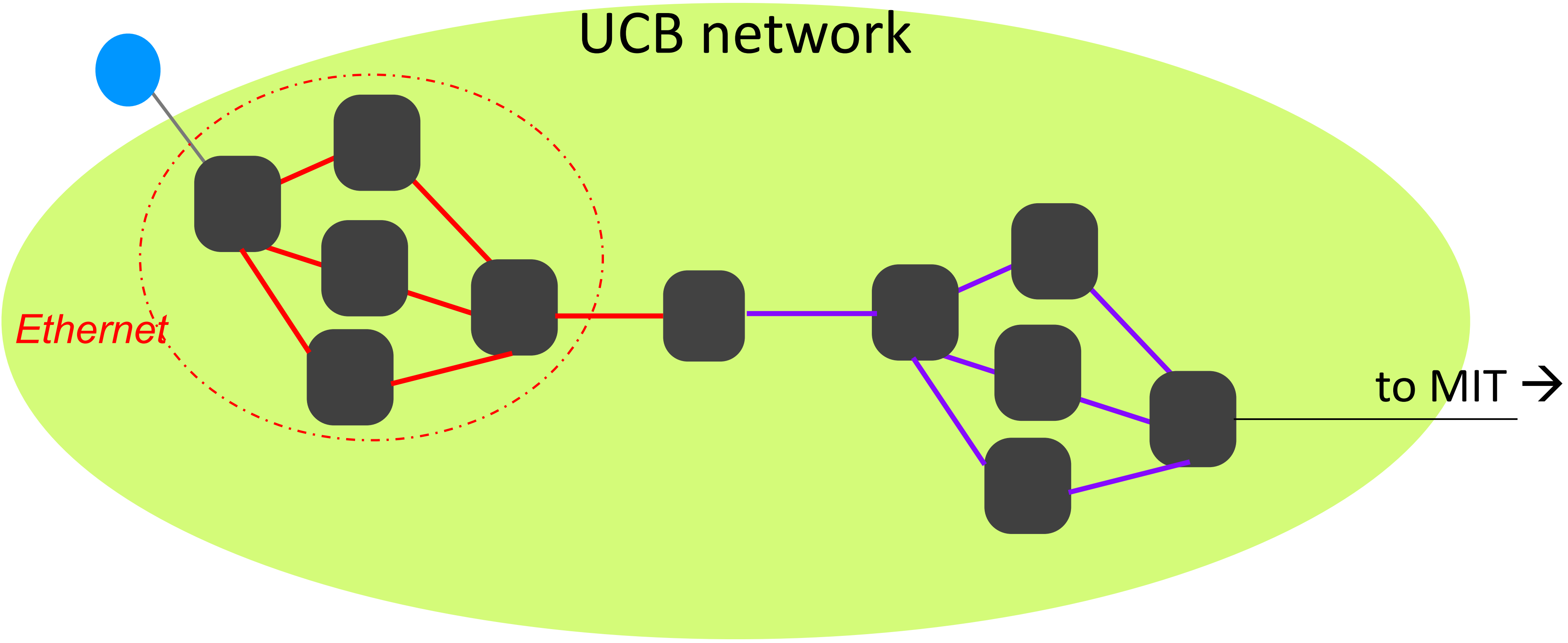
- Recall:
  - Local → relying on L2 headers (and local addresses)
  - Global → relying on L3 headers (and global addresses)

- So, when do we need L3 to interconnect L2 networks?
  1. When the L2 networks are based on different technologies
  2. When the L2 networks are operated independently (e.g., for administrative, policy, or scalability reasons)

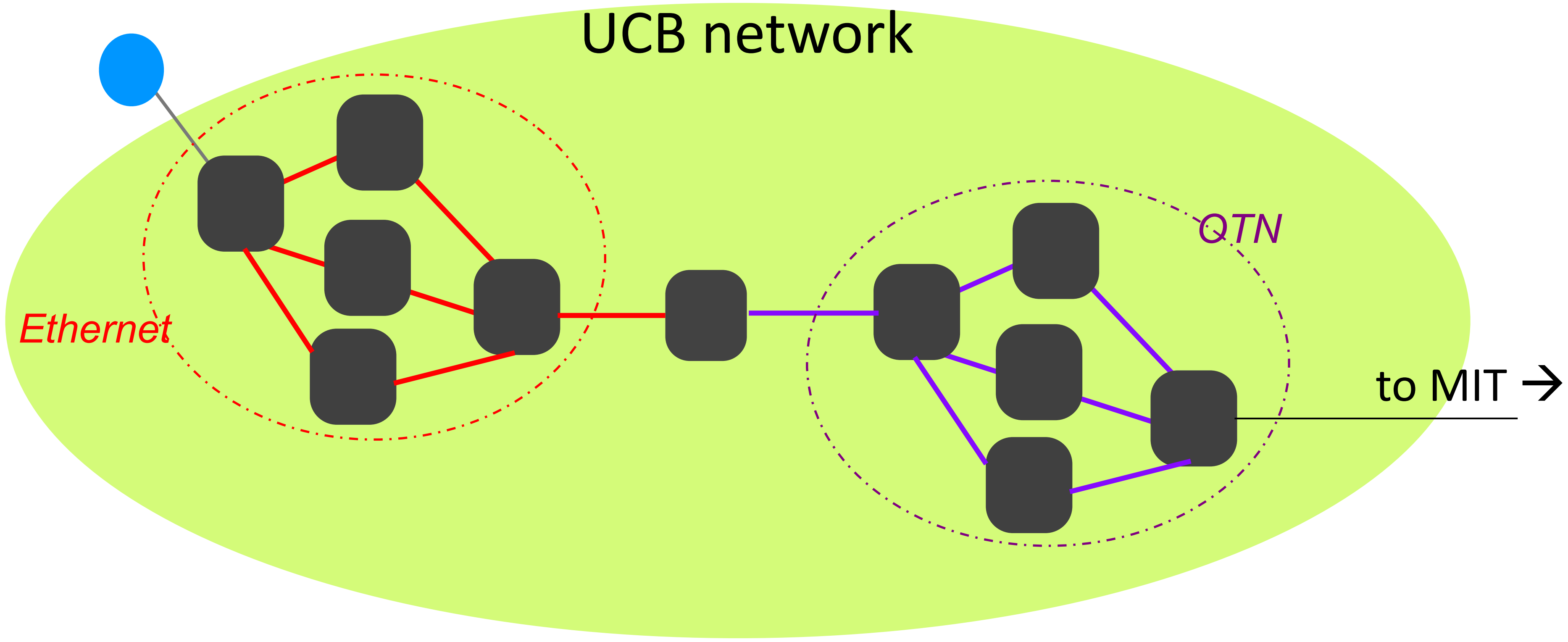
#1 Using L3 to interconnect different L2 technologies



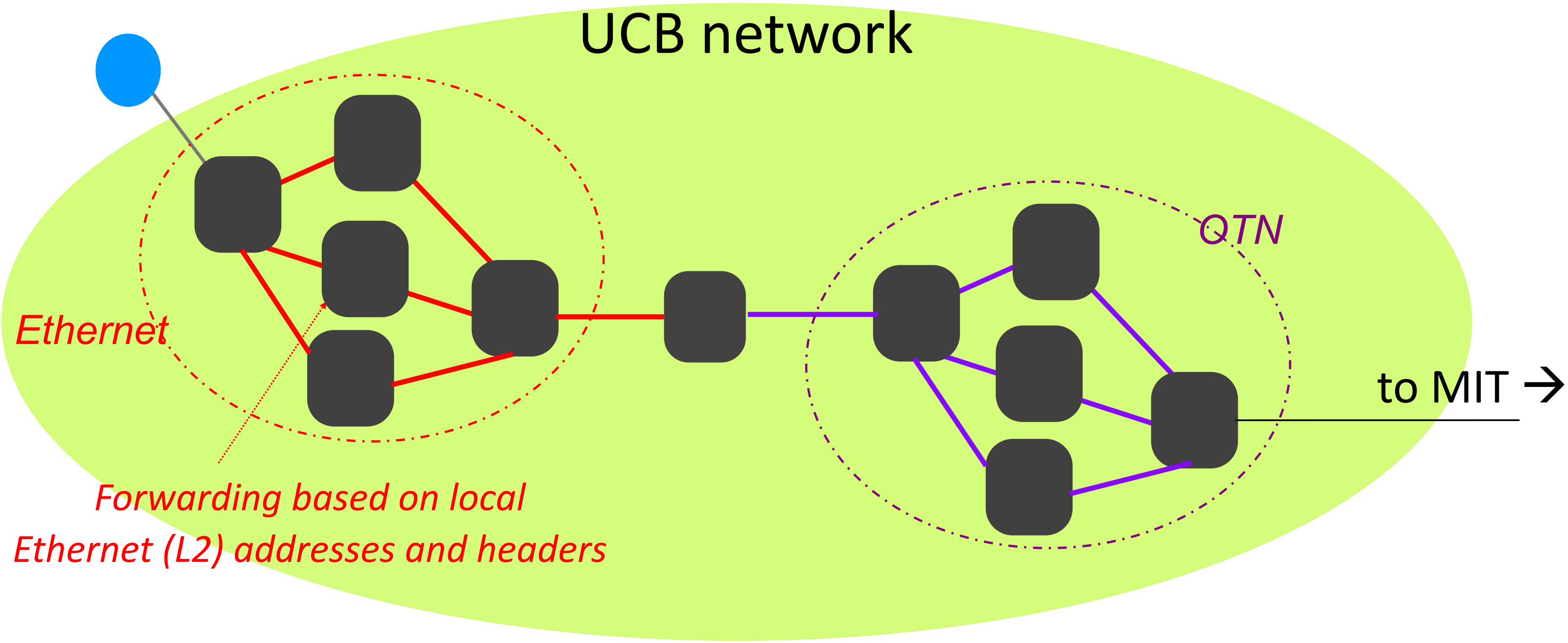
#1 Using L3 to interconnect different L2 technologies



#1 Using L3 to interconnect different L2 technologies

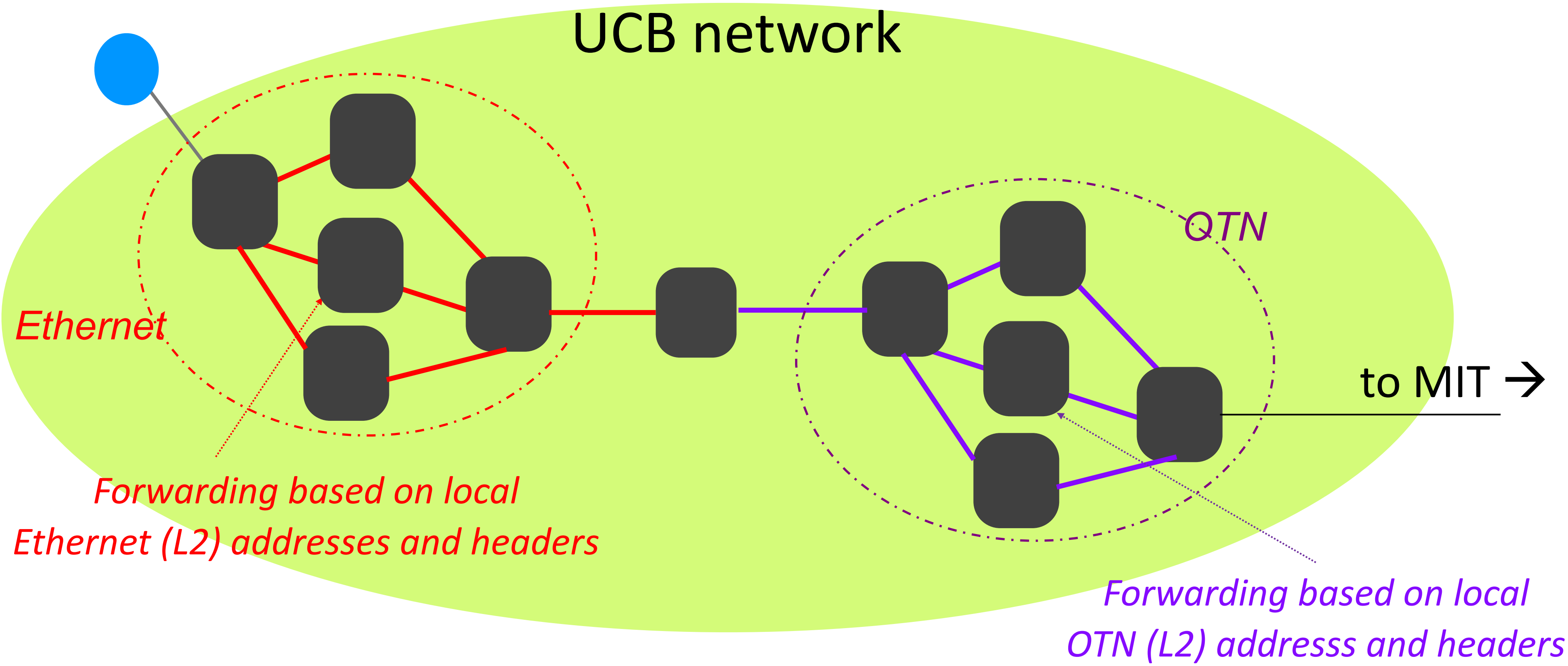


#1 Using L3 to interconnect different L2 technologies



Forwarding based on local Ethernet (L2) addresses and headers

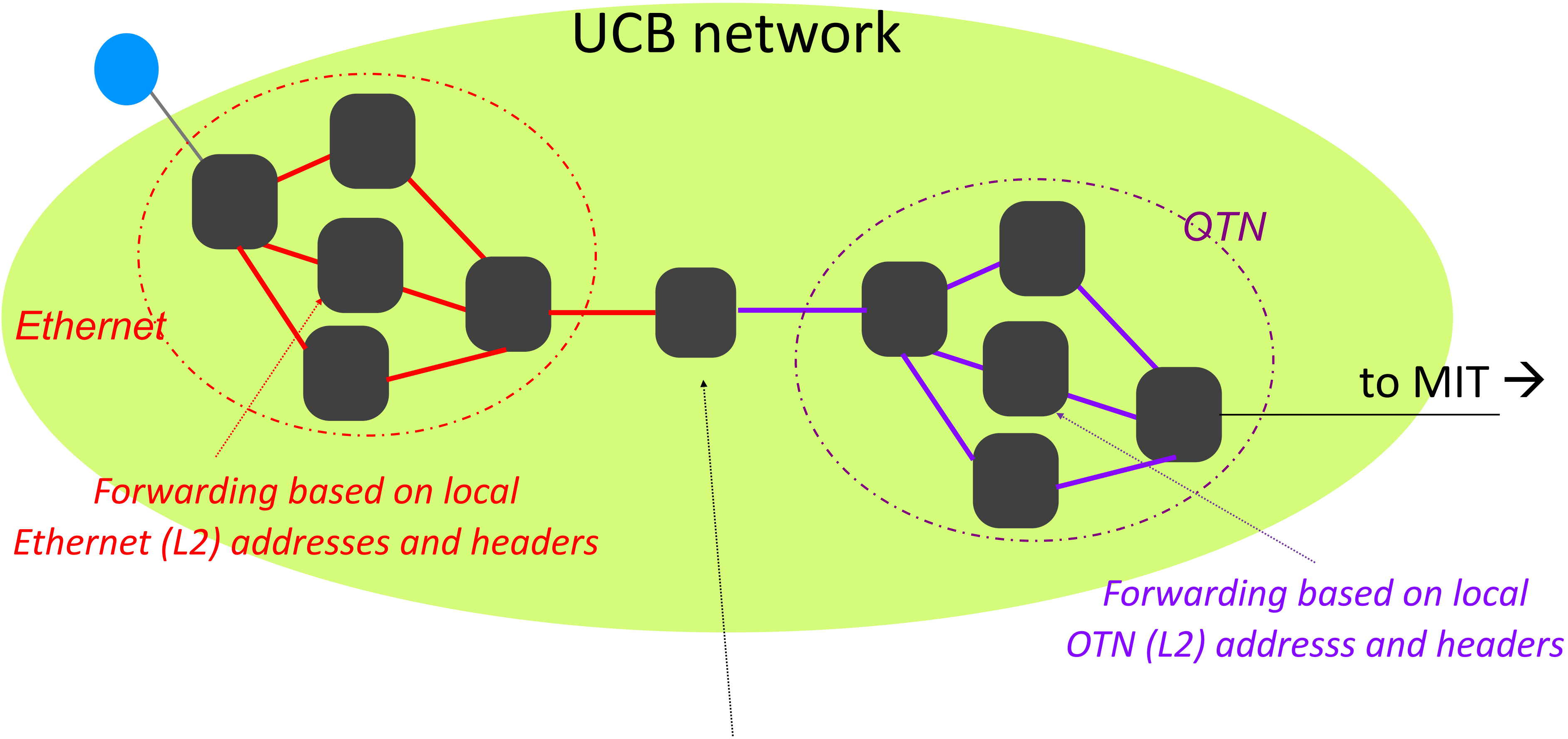
#1 Using L3 to interconnect different L2 technologies



Forwarding based on local Ethernet (L2) addresses and headers

Forwarding based on local OTN (L2) addresss and headers

#1 Using L3 to interconnect different L2 technologies

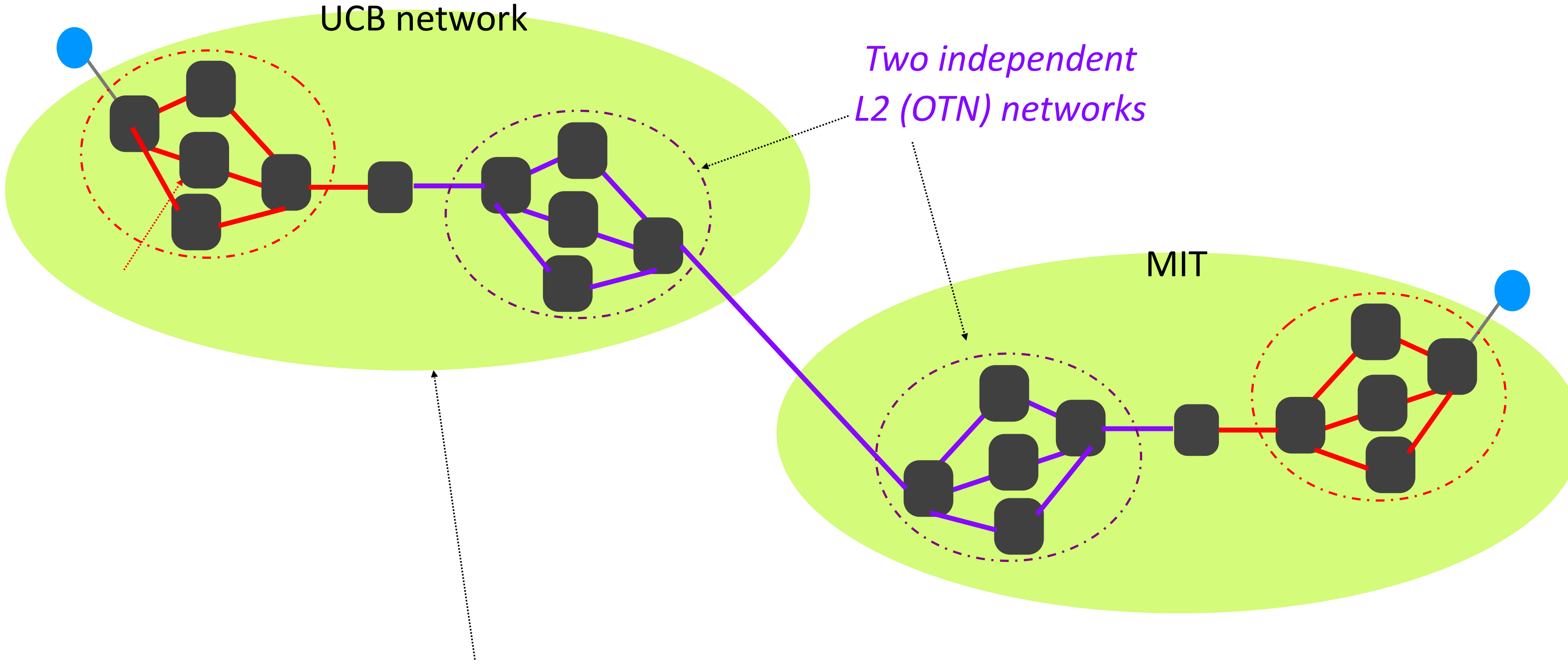


Forwarding based on local Ethernet (L2) addresses and headers

Forwarding based on local OTN (L2) addresss and headers

Forwarding based on global IP (L3) addresses and headers

#2) Using L3 to interconnect L2 networks in different administrative/policy domains



Packet forwarded based on global IP (L3) address and header

Local vs. Global networking

- Recall:
  - Local → relying on L2 headers (and local addresses)
  - Global → relying on L3 headers (and global addresses)

- So, when do we need L3 to interconnect L2 networks?
  1. When the L2 networks are based on different technologies
  2. When the L2 networks are operated independently (admin, policy)

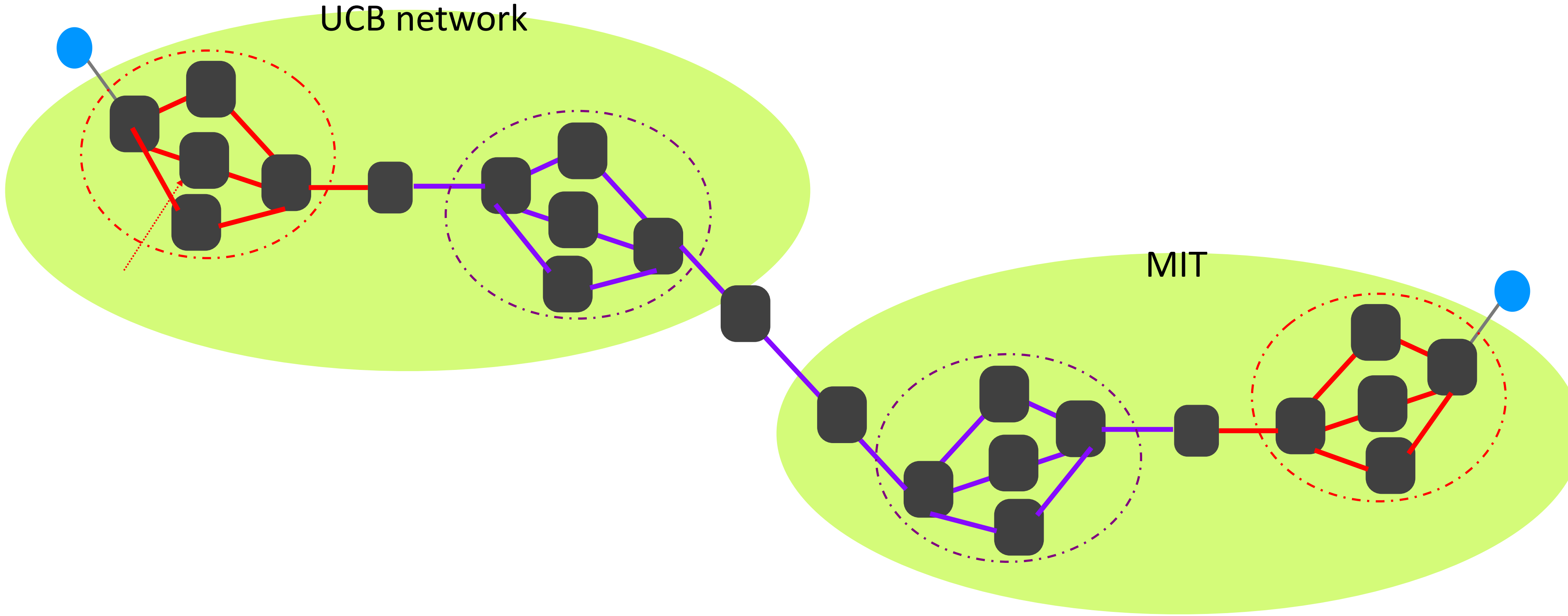
Local vs. Global networking

- Recall:
  - Local → relying on L2 headers (and local addresses)
  - Global → relying on L3 headers (and global addresses)

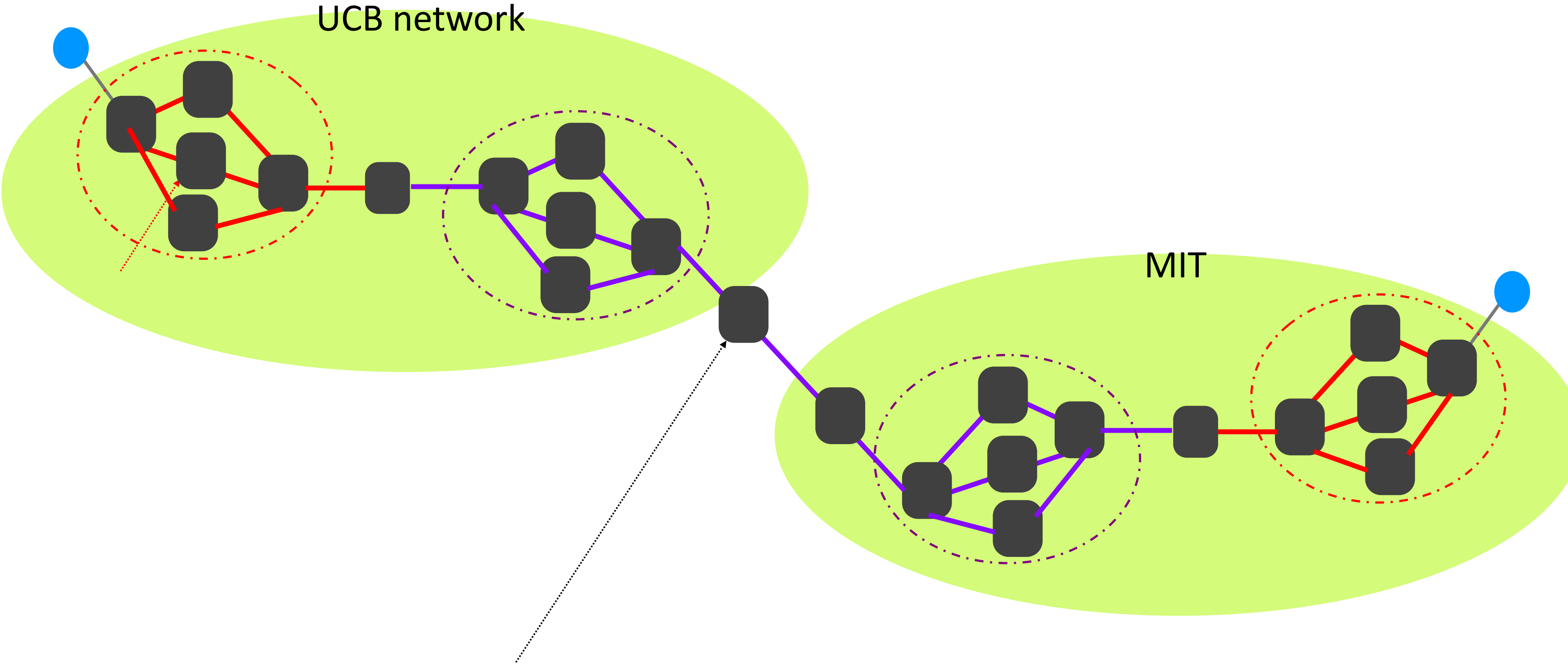
- So, when do we need L3 to interconnect L2 networks?
  1. When the L2 networks are based on different technologies
  2. When the L2 networks are operated independently (admin, policy)

- Can we just interconnect L3 routers directly?
  - Yes! Just a degenerate case of interconnecting L2 networks

#2a) Using L3 to interconnect L2 networks in different administrative/policy domains

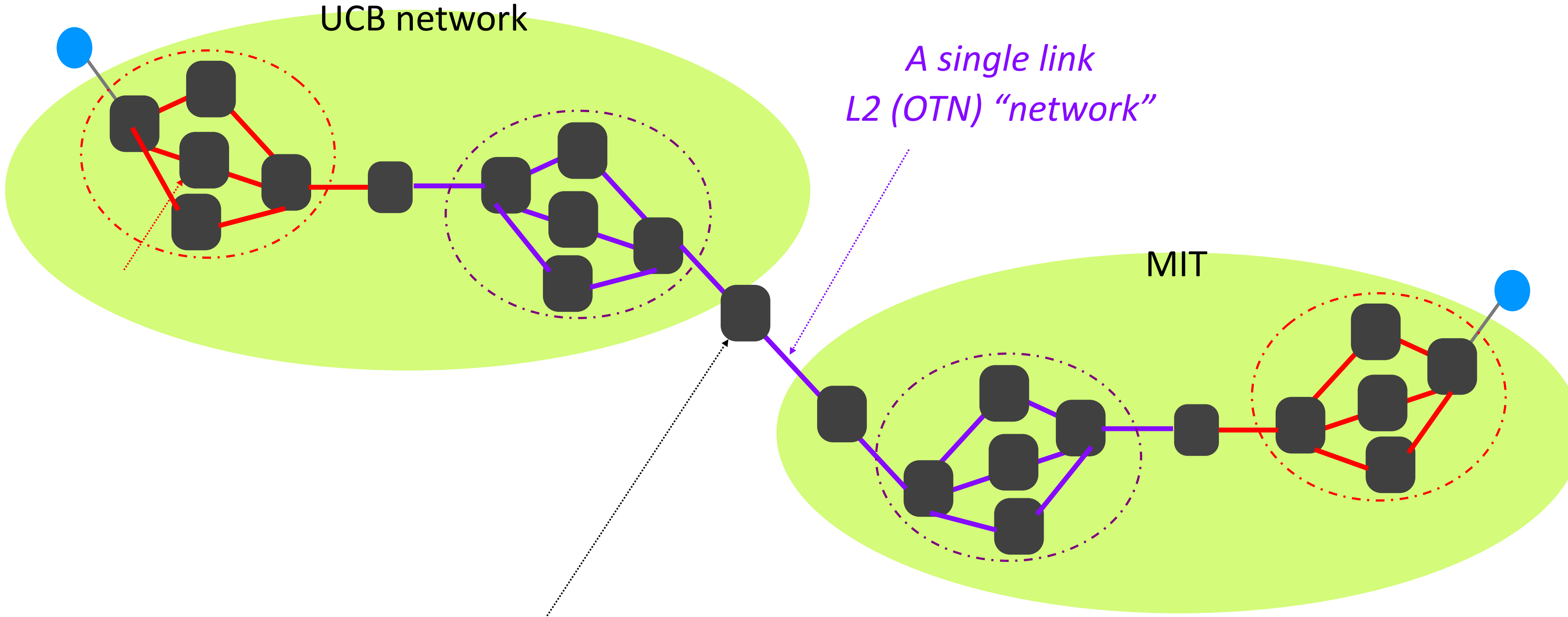


#2a) Using L3 to interconnect L2 networks in different administrative/policy domains



Packet forwarded based on global IP (L3) address and header

#2a) Using L3 to interconnect L2 networks in different administrative/policy domains

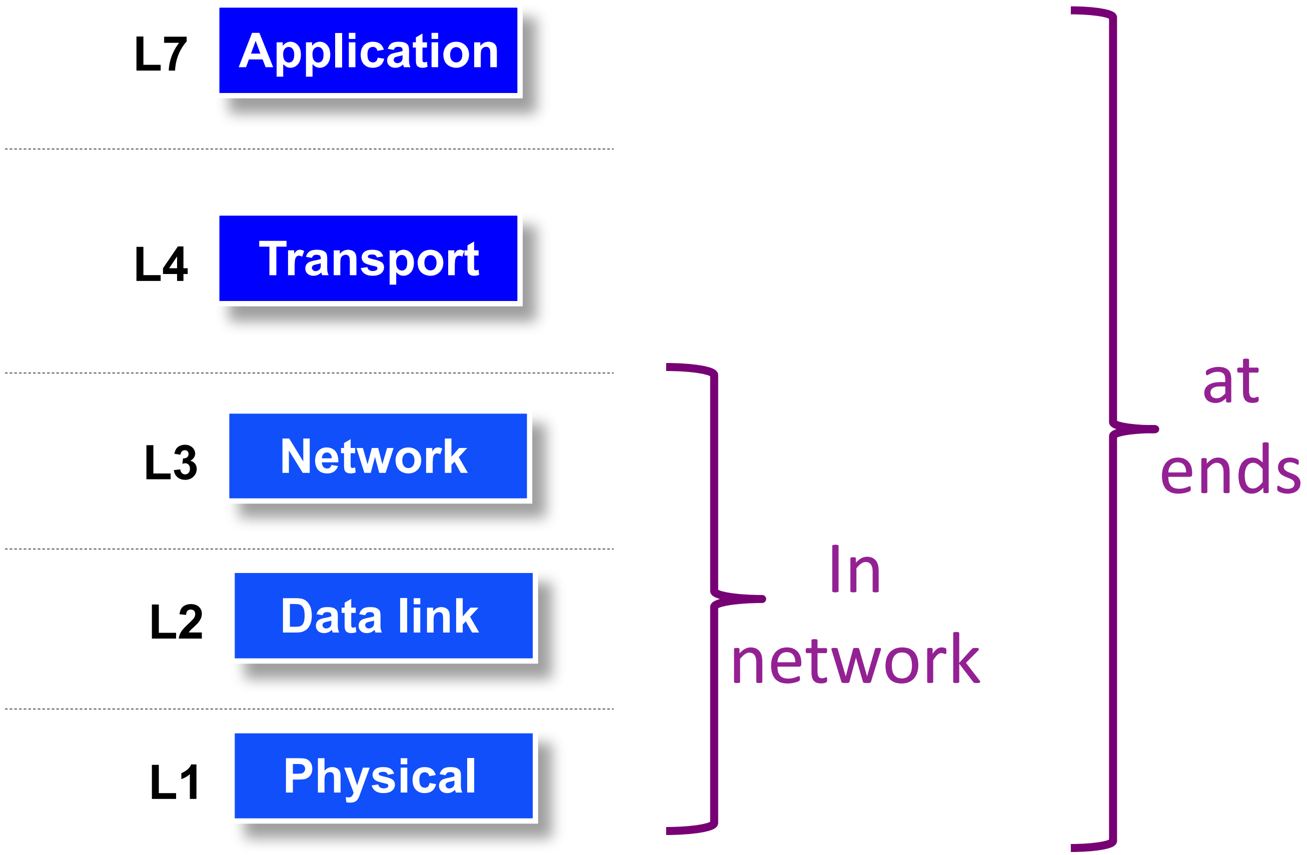


A single link L2 (OTN) "network"

Packet forwarded based on global IP (L3) address and header

Questions?

Rest of this lecture



Why is *this* assignment of tasks good?

Architectural Wisdom

- David D. Clark
  - Chief protocol architect for the Internet in the 80s

Architectural Wisdom

- David D. Clark
  - Chief protocol architect for the Internet in the 80s
- Co-authored two classics
  - “End-to-End Arguments in System Design” (1981)
  - “The Design Philosophy of the DARPA Internet Protocols” (1988)
- Articulates the rationale underlying the Internet’s arch.

The End-to-End Principle

The End-to-End Principle

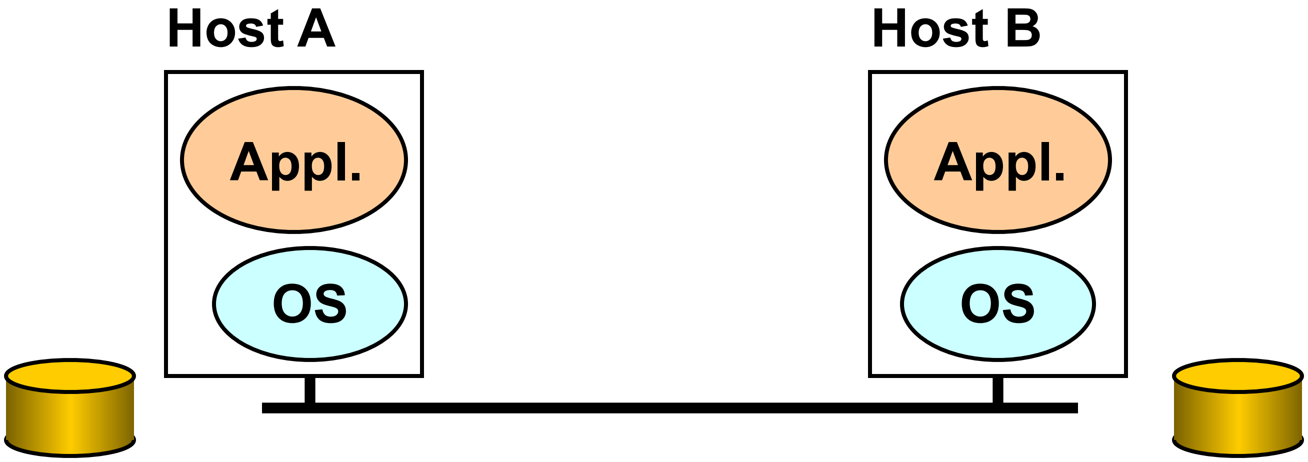
- Guides the debate about what functionality the network does or doesn’t implement

The End-to-End Principle

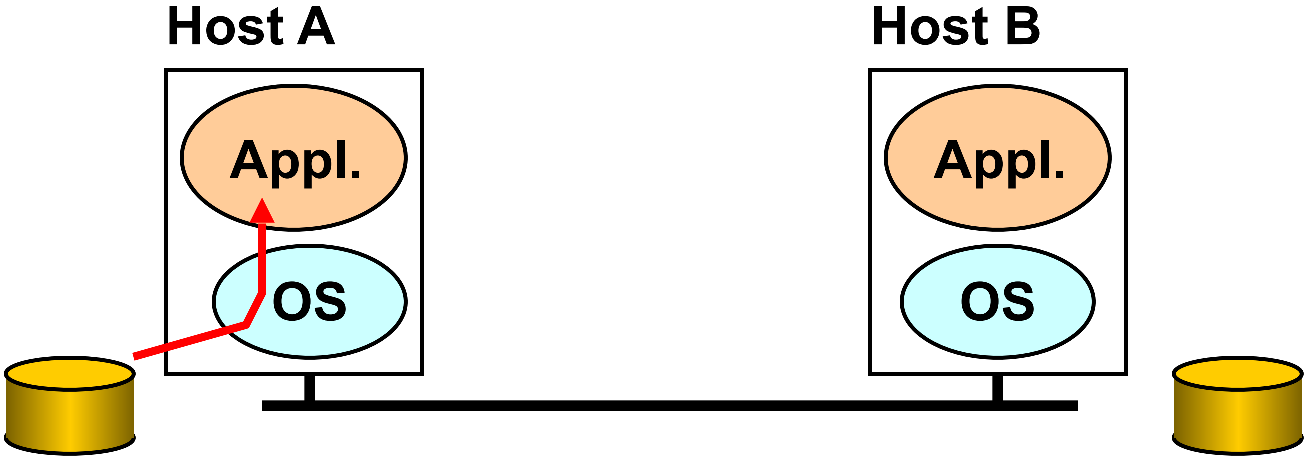
- Guides the debate about what functionality the network does or doesn’t implement

- Everyone believes it, but no one agrees on what it means …

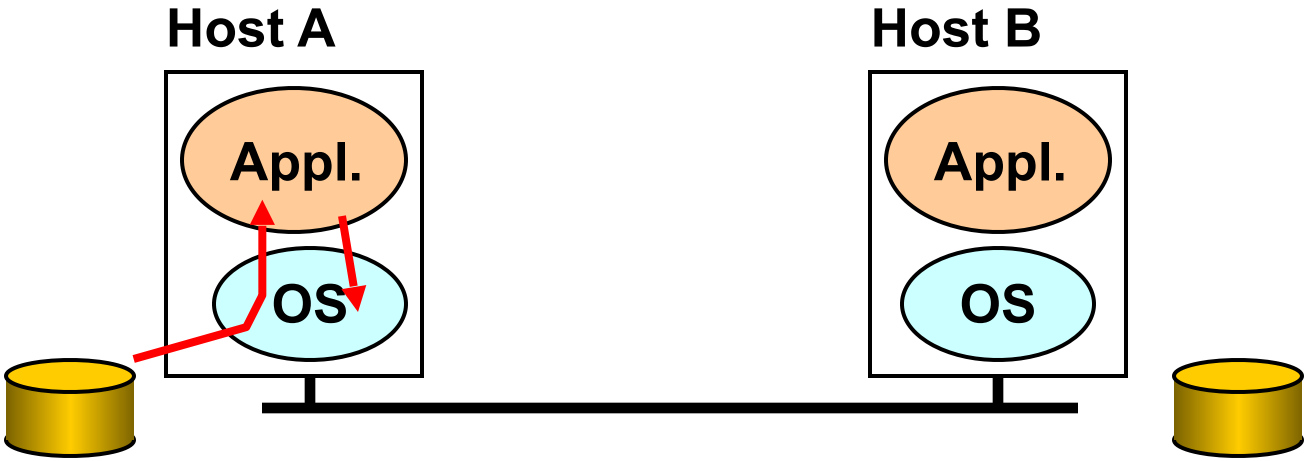
Example: Reliable File Transfer



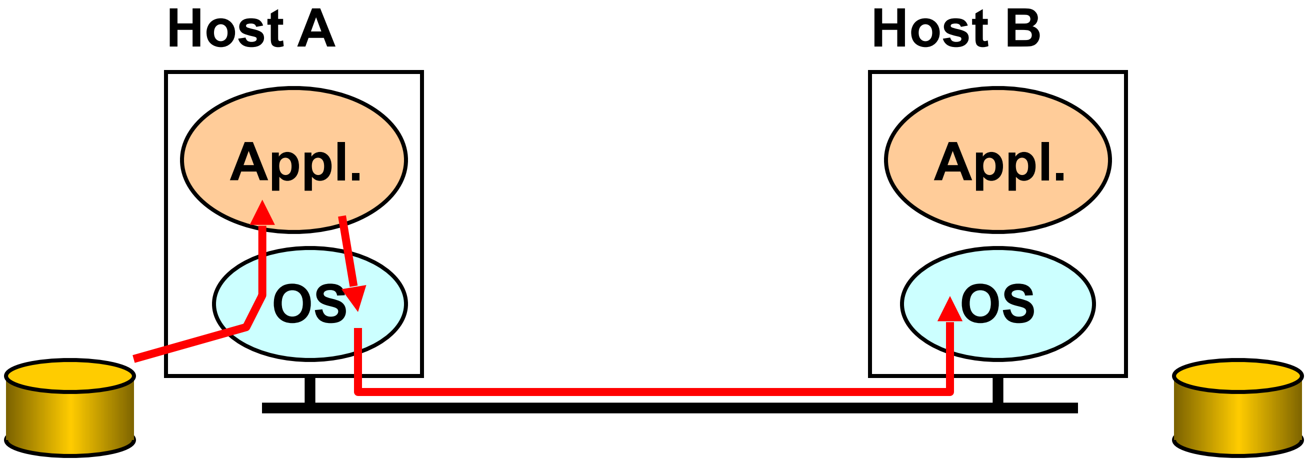
Example: Reliable File Transfer



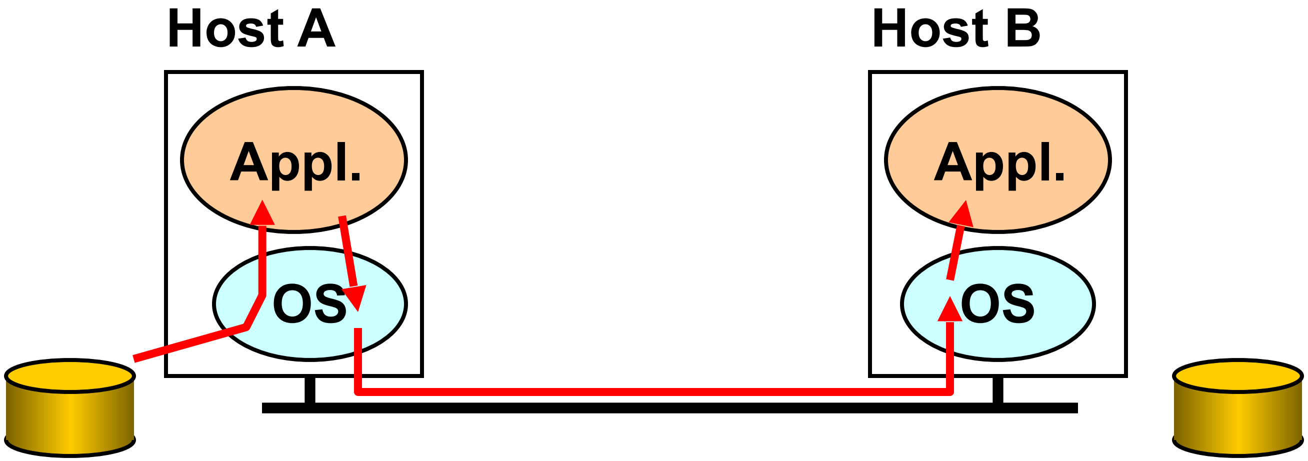
Example: Reliable File Transfer



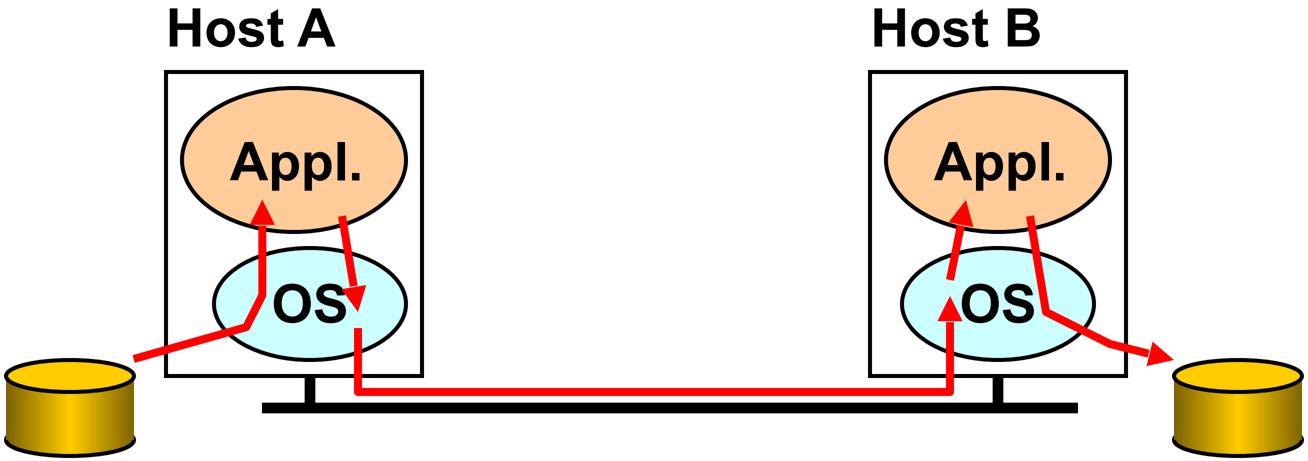
Example: Reliable File Transfer



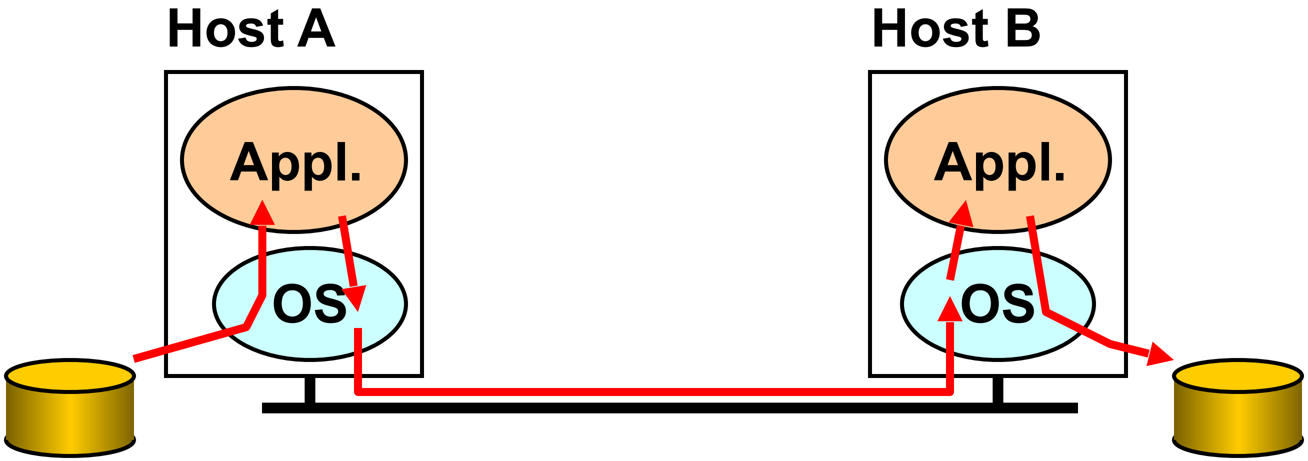
Example: Reliable File Transfer



Example: Reliable File Transfer

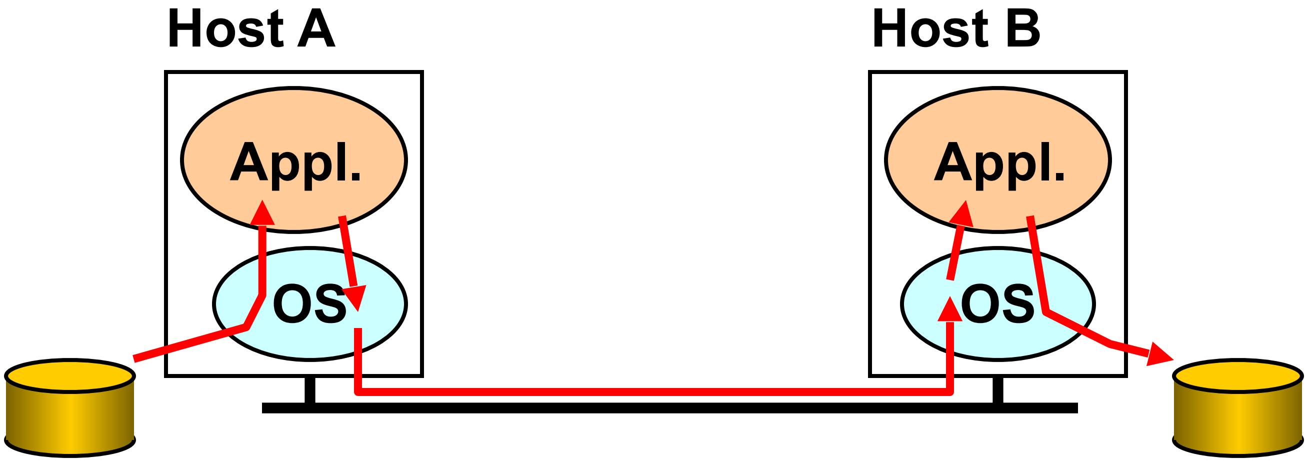


Example: Reliable File Transfer



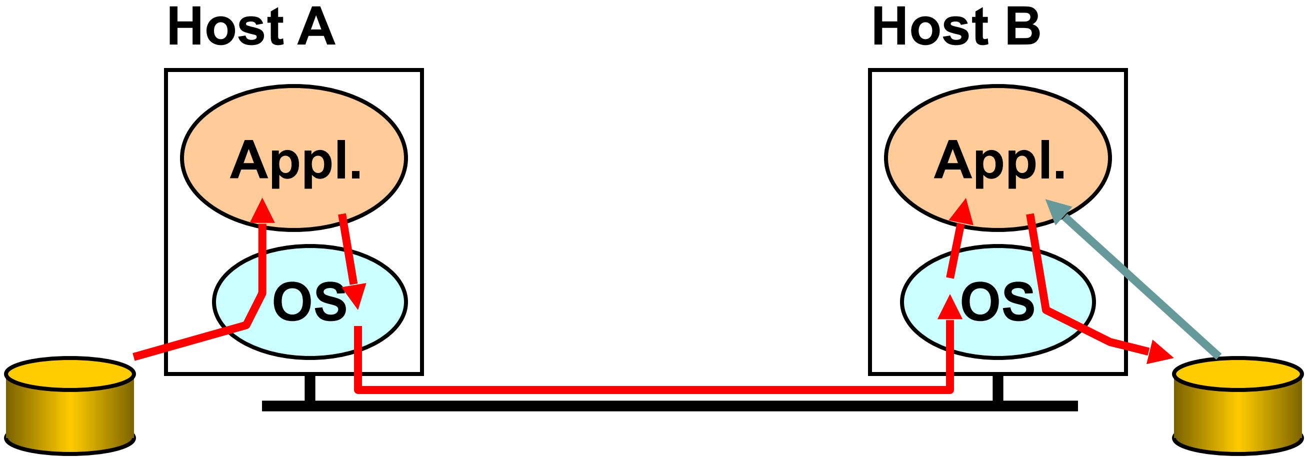
- Solution 1: make each step reliable
  (requires network to handle reliability)

Example: Reliable File Transfer



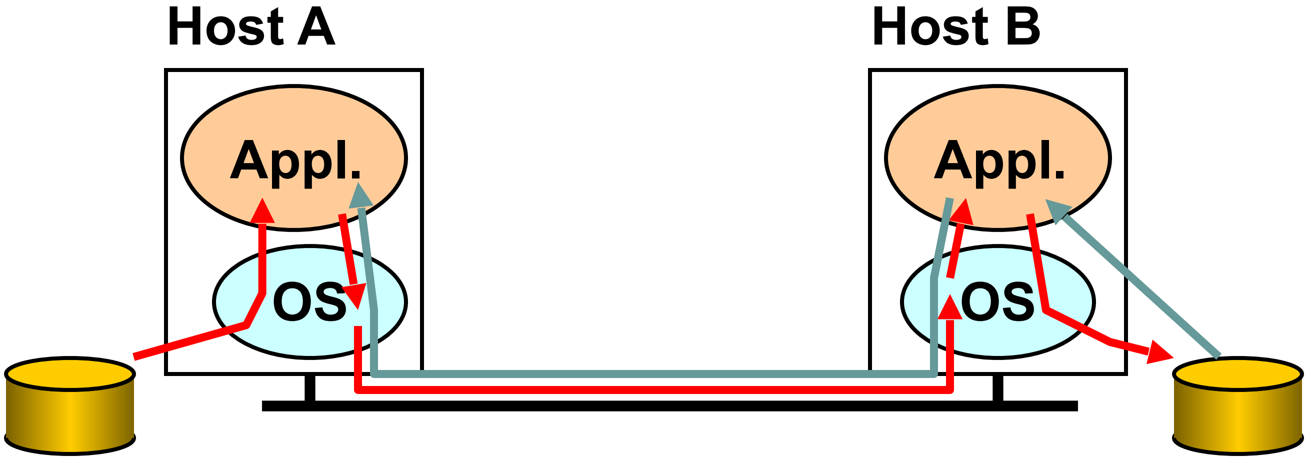
- Solution 1: make each step reliable
  (requires network to handle reliability)
- Solution 2: allow steps to be unreliable, but do end-to-end **check** and try again if necessary
  (do not assume network is reliable)

Example: Reliable File Transfer



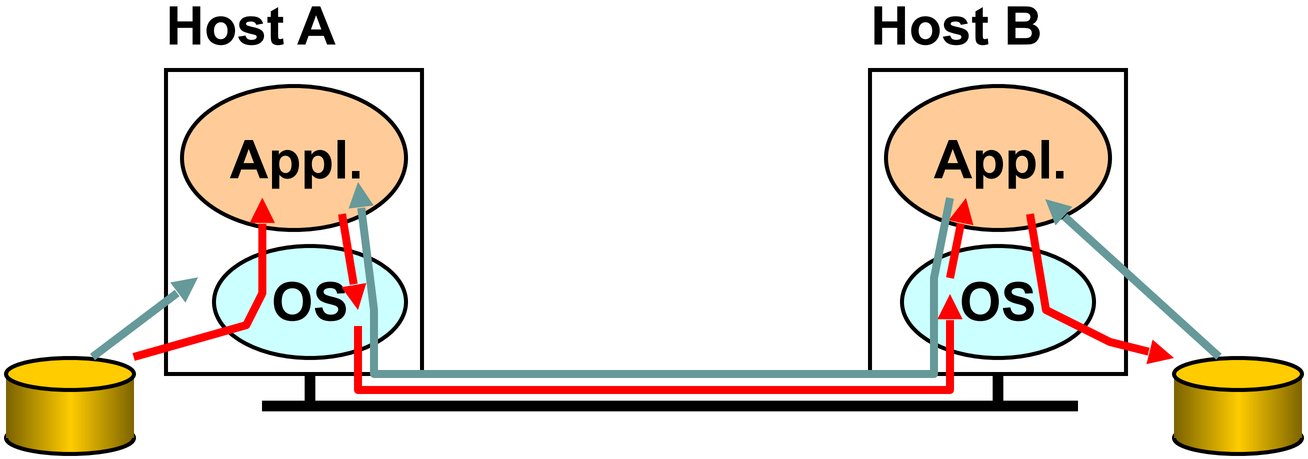
- Solution 1: make each step reliable
  (requires network to handle reliability)
- Solution 2: allow steps to be unreliable, but do end-to-end **check** and try again if necessary
  (do not assume network is reliable)

Example: Reliable File Transfer



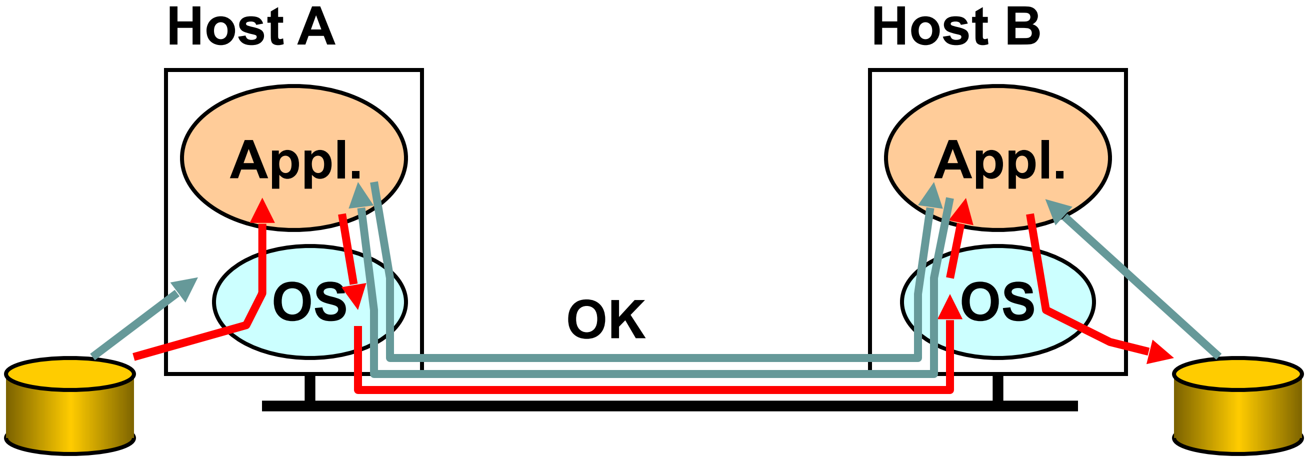
- Solution 1: make each step reliable
  (requires network to handle reliability)
- Solution 2: allow steps to be unreliable, but do end-to-end **check** and try again if necessary
  (do not assume network is reliable)

Example: Reliable File Transfer



- Solution 1: make each step reliable
  (requires network to handle reliability)
- Solution 2: allow steps to be unreliable, but do end-to-end **check** and try again if necessary
  (do not assume network is reliable)

Example: Reliable File Transfer



- Solution 1: make each step reliable
  (requires network to handle reliability)
- Solution 2: allow steps to be unreliable, but do end-to-end **check** and try again if necessary
  (do not assume network is reliable)

Discussion

Discussion

- Solution 1 cannot be made perfectly reliable
  - What happens if a component fails between steps?
  - Receiver has to do the check anyway!

Discussion

- Solution 1 cannot be made perfectly reliable
  - What happens if a component fails between steps?
  - Receiver has to do the check anyway!

- Solution 2 can also fail, but only if the endhost itself fails (i.e., doesn’t follow its own protocol)

Discussion

- Solution 1 cannot be made perfectly reliable
  - What happens if a component fails between steps?
  - Receiver has to do the check anyway!
- Solution 2 can also fail, but only if the endhost itself fails (i.e., doesn’t follow its own protocol)
- Solution 2 only relies on what it can control
  - The endpoint behavior

Discussion

- Solution 1 cannot be made perfectly reliable
  - What happens if a component fails between steps?
  - Receiver has to do the check anyway!
- Solution 2 can also fail, but only if the endhost itself fails (i.e., doesn’t follow its own protocol)
- Solution 2 only relies on what it can control
  - The endpoint behavior
- Solution 1 requires endpoints trust other elements

Recap

Recap

Making the network reliable:

Recap

Making the network reliable:

- Doesn’t reduce host implementation complexity

Recap

Making the network reliable:

- Doesn’t reduce host implementation complexity
- Does increase network complexity

Recap

Making the network reliable:

- Doesn’t reduce host implementation complexity
- Does increase network complexity
- Can impose overhead on all applications, *even if they don’t need reliability*

Two Notions of Reliability (Clark)

Two Notions of Reliability (Clark)

- The network recovers from failures quickly so that, as long as *some* path exists, two endpoints should be able to communicate.

- Network failures should not interfere with endpoint *semantics*

Two Notions of Reliability (Clark)

- The network recovers from failures quickly so that, as long as *some* path exists, two endpoints should be able to communicate.

- Network failures should not interfere with endpoint *semantics*

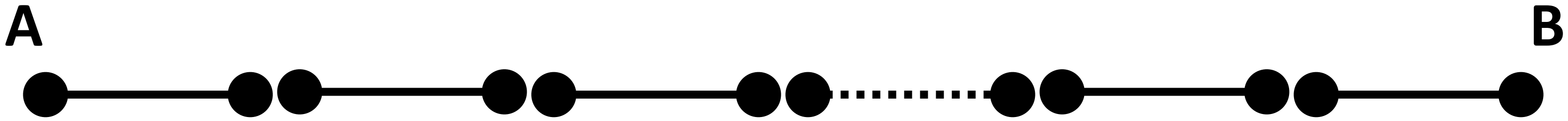
Two Notions of Reliability (Clark)

- The network recovers from failures quickly so that, as long as *some* path exists, two endpoints should be able to communicate.
- Network failures should not interfere with endpoint *semantics*
- The second requirement implies that we must adopt solution 2 (cannot depend on network).

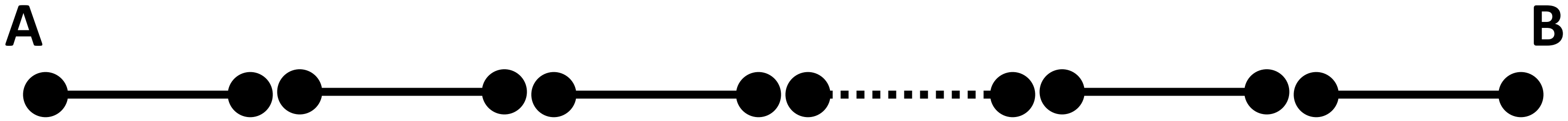
So...

- Should you ever implement reliability in network?
    - I.e., in addition to doing so in the hosts

Performance

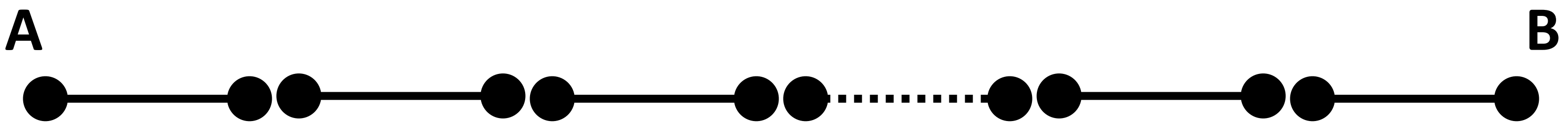


Performance



- If each link drops packets 10% of the time, and we have 10 links, then E2E failure rate is ~65%

Performance



- If each link drops packets 10% of the time, and we have 10 links, then E2E failure rate is ~65%

- What if the link implemented two retransmissions?
  - Per-link drop rate reduced to 0.1%, E2E error rate is ~1%

Performance

- Should you ever implement reliability in network?
  - I.e., in addition to doing so in the hosts

Performance

- Should you ever implement reliability in network?
  - I.e., in addition to doing so in the hosts
- Perhaps, if needed for reasonable performance

Performance

- Should you ever implement reliability in network?
  - I.e., in addition to doing so in the hosts

- Perhaps, if needed for reasonable performance
  - Don’t aim for perfect reliability, but ok to reduce error rate

The end-to-end argument in Clark’s words

“The function in question can completely and correctly be implemented only with the knowledge and help of the application at the end points. Therefore, providing that function as a feature of the communication system itself is not possible. (Sometimes an incomplete version of the function provided by the communication system may be useful as a performance enhancement.)”

The End-to-End Principle

- Everyone believes it, but no one knows what it means…..

The End-to-End Principle

- Everyone believes it, but no one knows what it means…..

- Pretty convincing in the context of reliability but not as clear in other cases

The End-to-End Principle

- Everyone believes it, but no one knows what it means…..
- Pretty convincing in the context of reliability but not as clear in other cases
- In general, three interpretations of the end-to-end principle

“Only-if-Sufficient” Interpretation

- Don’t implement a function at the lower levels of the system unless it can be completely implemented at this level

- *Don’t bother unless you can eliminate the burden from hosts*

“Only-if-Necessary” Interpretation

- Don’t implement *anything* in the network that can be implemented correctly by the hosts

- Make network layer absolutely minimal
  - This E2E interpretation trumps performance issues
  - Increases flexibility, since lower layers stay **simple**

“Only-if-Useful” Interpretation

- If hosts can implement functionality correctly, implement it in a lower layer only as a performance enhancement

- But do so only if it does not impose overhead on apps that do not require that functionality

- This criterion typically weighs performance heavily in deciding where to place functionality

What Does This Mean In Practice?

Interpretation	Reliability	QoS (Priority forwarding)	Routing
Sufficient	No	?	?
Necessary	No	?	?
Useful	Sometimes	-	?

- Reliable Transport: at ends (sometimes network)

What Does This Mean In Practice?

Interpretation	Reliability	QoS (Priority forwarding)	Routing
Sufficient	No	?	?
Necessary	No	?	?
Useful	Sometimes	-	?

- Reliable Transport: at ends (sometimes network)
- Priorities: ?

What Does This Mean In Practice?

Interpretation	Reliability	QoS (Priority forwarding)	Routing
Sufficient	No	?	?
Necessary	No	?	?
Useful	Sometimes	-	?

- Reliable Transport: at ends (sometimes network)
- Priorities: ?
- Routing: ?

What Does This Mean In Practice?

Interpretation	Reliability	QoS (Priority forwarding)	Routing
Sufficient	No	Yes	Yes
Necessary	No	Yes	No
Useful	Sometimes	-	Yes

- Reliable Transport: at ends (sometimes network)

What Does This Mean In Practice?

Interpretation	Reliability	QoS (Priority forwarding)	Routing
Sufficient	No	Yes	Yes
Necessary	No	Yes	No
Useful	Sometimes	-	Yes

- Reliable Transport: at ends (sometimes network)
- Priorities: In network

What Does This Mean In Practice?

Interpretation	Reliability	QoS (Priority forwarding)	Routing
Sufficient	No	Yes	Yes
Necessary	No	Yes	No
Useful	Sometimes	-	Yes

- Reliable Transport: at ends (sometimes network)
- Priorities: In network
- Routing: In network (in almost all cases)

Summary

Summary

- **Where** to implement functionality is non-trivial
  - E2E principle shaped how we reason about tradeoffs!

Summary

- **Where** to implement functionality is non-trivial
  - E2E principle shaped how we reason about tradeoffs!

- Important: remember it's an argument, not a rule
  - Though everyone agrees that reliability should be primarily implemented in the hosts

Characteristics often attributed to the E2E principle

Characteristics often attributed to the E2E principle

- “Dumb” network and “smart” end systems

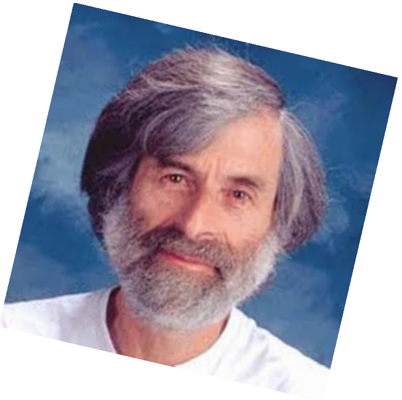
Characteristics often attributed to the E2E principle

- “Dumb” network and “smart” end systems
- “Fate sharing”

A Cynical View of Distributed Systems

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable”

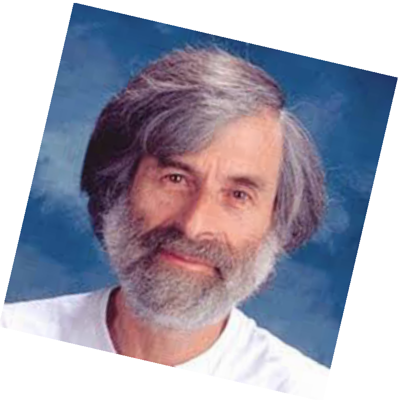
-- Leslie Lamport



A Cynical View of Distributed Systems

“A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable”

-- Leslie Lamport



Fate Sharing tries to prevent this!

General Principle: *Fate-Sharing*

General Principle: *Fate-Sharing*

- When storing state in a distributed system, co-locate it with entities that rely on that state

- Only way failure can cause loss of the critical state is if the entity that cares about it also fails ...
  - … in which case it doesn't matter

General Principle: *Fate-Sharing*

- When storing state in a distributed system, co-locate it with entities that rely on that state

- Only way failure can cause loss of the critical state is if the entity that cares about it also fails ...
  - … in which case it doesn’t matter

- Often argues for keeping *flow state* at end hosts rather than inside routers
  - E.g., packet-switching rather than circuit-switching

What Does E2E Principle Ignore?

What Does E2E Principle Ignore?

- There are other stakeholders besides users
    - ISPs care about the operation/security of their network
    - Plus looking for new revenue-generating functions

What Does E2E Principle Ignore?

- There are other stakeholders besides users
  - ISPs care about the operation/security of their network
  - Plus looking for new revenue-generating functions
- These functions more easily done in the network.
  - Think of firewalls…..

What Does E2E Principle Ignore?

- There are other stakeholders besides users
    - ISPs care about the operation/security of their network
    - Plus looking for new revenue-generating functions

- These functions more easily done in the network.
    - Think of firewalls…..

- **Easier because this is what the ISPs control!**
    - They don't control hosts, so it doesn't matter if it *could* be implemented in the hosts; it *won't* be

What Does E2E Principle Ignore?

- There are other stakeholders besides users
  - ISPs care about the operation/security of their network
  - Plus looking for new revenue-generating functions
- These functions more easily done in the network.
  - Think of firewalls…..
- **Easier because this is what the ISPs control!**
  - They don’t control hosts, so it doesn’t matter if it *could* be implemented in the hosts; it *won’t* be
- Led to widespread deployments of “middleboxes”
  - Firewalls, proxies, NAT, etc. Will cover later in course…

Recap: architectural wisdom (the ”how")

Recap: architectural wisdom (the ”how")

- How to decompose system into modules?
  - Layering

Recap: architectural wisdom (the ”how")

- How to decompose system into modules?
  - Layering
- Where are layers implemented?
  - End hosts implement all layers (L1-L7)
  - Network implements only layers (L1-L3)

Recap: architectural wisdom (the ”how")

- How to decompose system into modules?
  - Layering
- Where are layers implemented?
  - End hosts implement all layers (L1-L7)
  - Network implements only layers (L1-L3)
- One unifying protocol at the network layer
  - Internet Protocol (IP)

Recap: architectural wisdom (the "why")

Recap: architectural wisdom (the "why")

- **Layering** provided a clean separation of concerns
  - And hence enabled innovation!

Recap: architectural wisdom (the "why")

- **Layering** provided a clean separation of concerns
  - And hence enabled innovation!

- **End-to-end principle** kept unnecessary state and functionality out of the network
  - And hence allowed the Internet to scale!