5590, fall 2020
software defined networking

anduo wang, Temple University

T 17:30-20:00

Fabric: a retrospective on evolving SDN

end-to-end arguments in system design

MPLS

Fabric: end-to-end arguments + MPLS

End-To-End Arguments in System Design

http://web.mit.edu/Saltzer/www/publications/endtoend/endtoend.pdf

End-To-End arguments

design principle

- the placement of functions among the modules of a distributed system

End-To-End arguments

design principle

- the placement of functions among the modules of a distributed system
- functions placed at lower level
  - redundant
  - of little value

moving a function upward

placing a function in a layered system closer to the application that uses the function

- one class of function placement
- sharpened by the emergence of data communication network

data communication network

for a distributed system that includes communication

- draw a modular boundary around the communication subsystem (network) and a firm interface between it and the rest of the system
- a function can be placed at?

data communication network

for a distributed system that includes communication

- draw a modular boundary around the communication subsystem (network) and a firm interface between it and the rest of the system
- a function can be placed at
  - the network subsystem
  - the client (application that uses the function)
  - the joint nature
  - redundantly

data communication network

for a distributed system that includes communication

- draw a modular boundary around the communication subsystem (network) and a firm interface between it and the rest of the system
- a function can be placed at
  - the network subsystem
  - the client (application that uses the function)
  - the joint nature
  - redundantly

data communication network

for a distributed system that includes communication

- draw a modular boundary around the communication subsystem (network) and a firm interface between it and the rest of the system
- a f
  - t
  - t
  - t
  - r

End-To-End argument

- the function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the endpoints of the communication subsystem
- providing that questioned function as a feature of the communication subsystem is impossible

example function — reliable data transfer (rdt)

from host A to host B, failures can occur at various points

- A passes (app) data to the rdt program
- A rdt program askes the network subsystem to transmit
- the network subsystem moves packets from A to B
- B communication program removes packets from the network protocol to the rdt app
- rdt app writes the received data on the disc

reliable data transfer (rdt) — 1st attempt

brute force countermeasure

- reinforce each of the steps along the way
  - using duplicates, time-out, retry, redundancy, error checking
- reduce the probability of each individual threat

rdt — alternate approach

end-to-end check and retry

- if something wrong, retry from the beginning
- when failure rare:
  - normally work on a first try, occasionally a 2nd/3rd tries

brute force countermeasure VS. end-to-end check and retry

Q: whether or not this attempt to be helpful on the part of the network is useful to the rdt app

- brute force
  - even the threat of one step (e.g., step 4) is eliminated, the rdt app must still counter the remaining threats
  - only reduce the frequency of retries
  - no effect on the inevitability of correctness of the outcome

brute force countermeasure VS. end-to-end check and retry

Q: whether or not this attempt to be helpful on the part of the network is useful to the rdt app

- brute force
  - even the threat of one step (e.g., step 4) is eliminated, the rdt app must still counter the remaining threats
  - only reduce the frequency of retries
  - no effect on the inevitability of correctness of the outcome
  - for the network to go out of its way to be extraordinarily reliable does not reduce the burden on the app …

brute force countermeasure VS. end-to-end check and retry

Q: amount of effort to put into reliable measures

- an engineering trade-off based on performance, rather than a requirement for correctness, frequently the trade-off is complex
- brute force
  - more efficient (hop-by-hop), but some app may find the cost of the enhancement not worth the result
- end to end check and retry
  - within app, simplifies the network but increases overall cost

other functions

delivery guarantees
secure transmission
duplicate message suppression
in order message delivery

end to end argument and "Occam's Razor"

Occam's Razor

- do not make more assumptions than the minimum needed

end-to-end argument is a kind of "Occam's Razor"

- when it comes to choosing the functions to be provided within a subsystem
  - the subsystem frequently specified before app that uses the subsystem are known
  - a rational principle for organizing the subsystem

MPLS, the 2.5 layer

Tag Switching Architecture Overview

https://ieeexplore.ieee.org/document/650179/

tag switching =

a label swapping forwarding paradigm + network layer routing

tag switching =

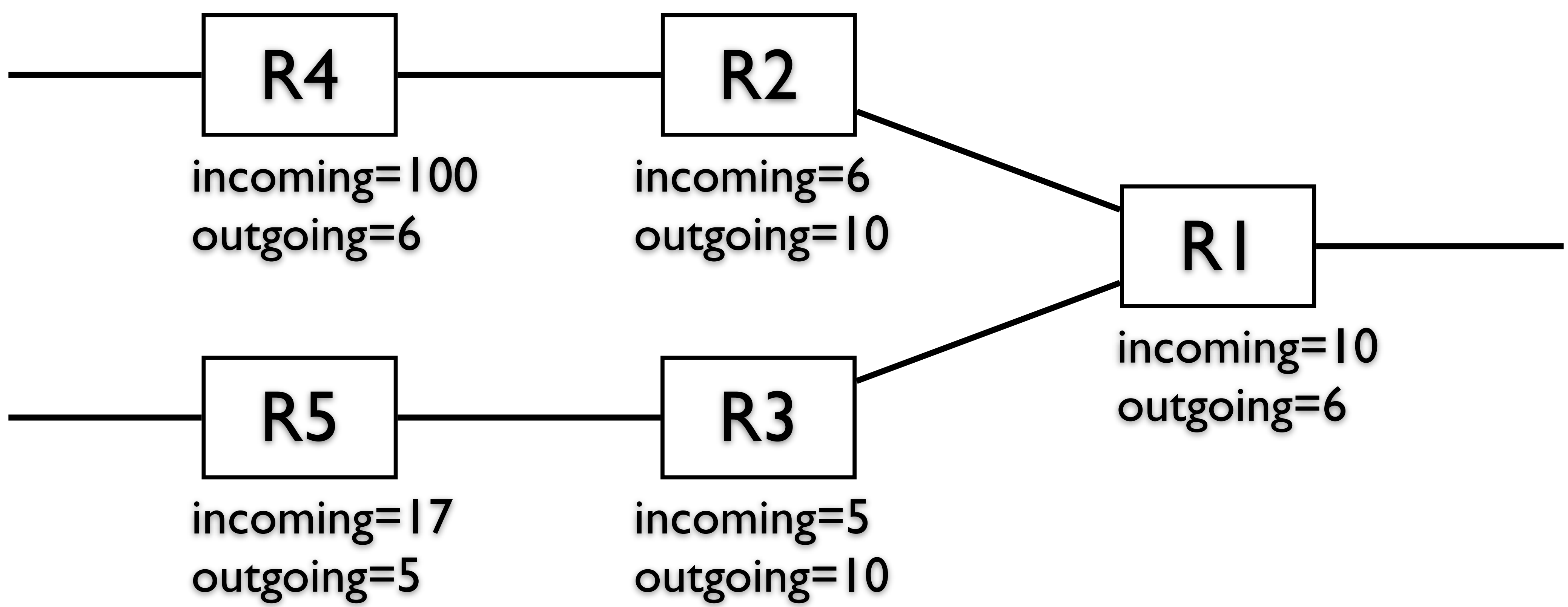
a label swapping forwarding paradigm + network layer routing

(forwarding component) (control component)

forwarding — label swapping

a tag switch uses the *tag as an index* in its TFIB

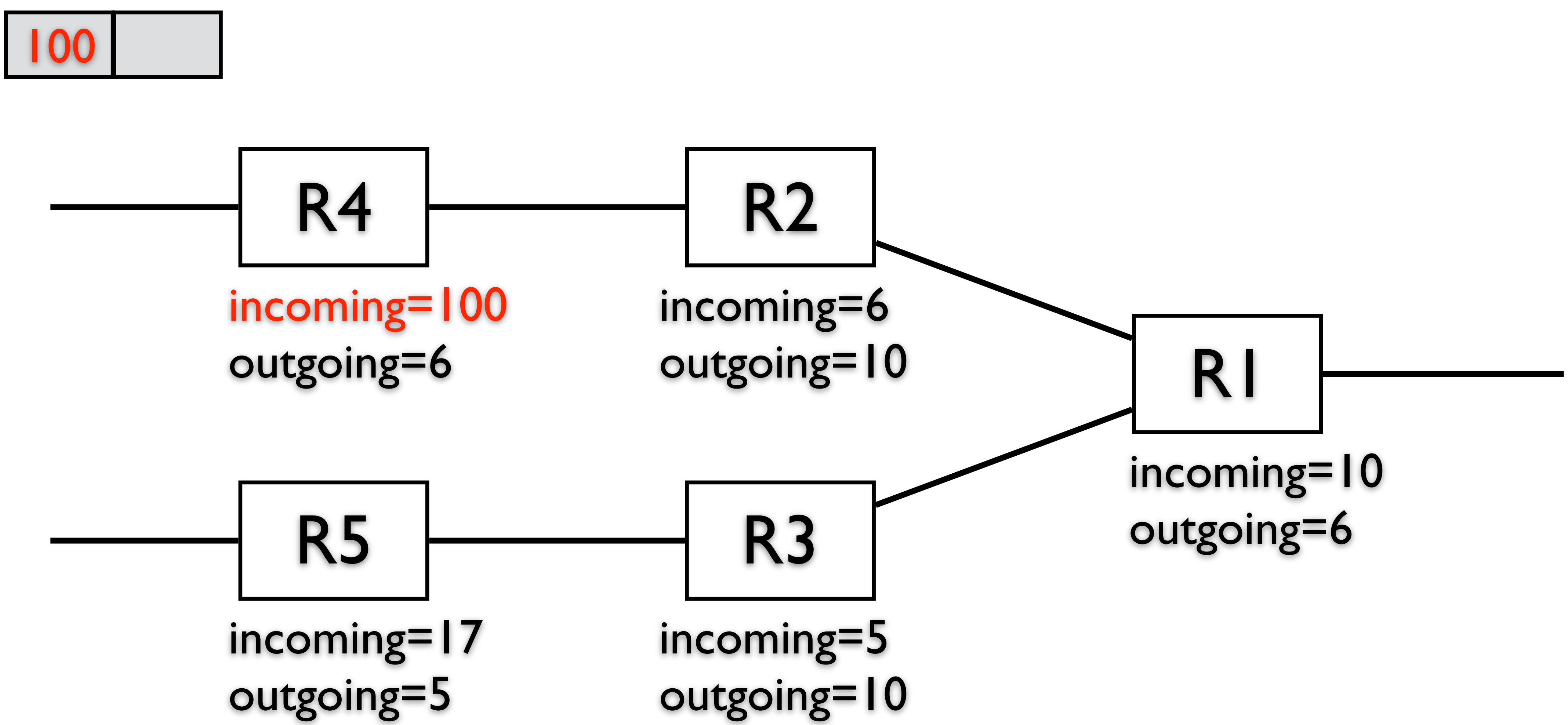
- <incoming tag, outgoing tag, outgoing interface …>



forwarding — label swapping

a tag switch uses the *tag as an index* in its TFIB

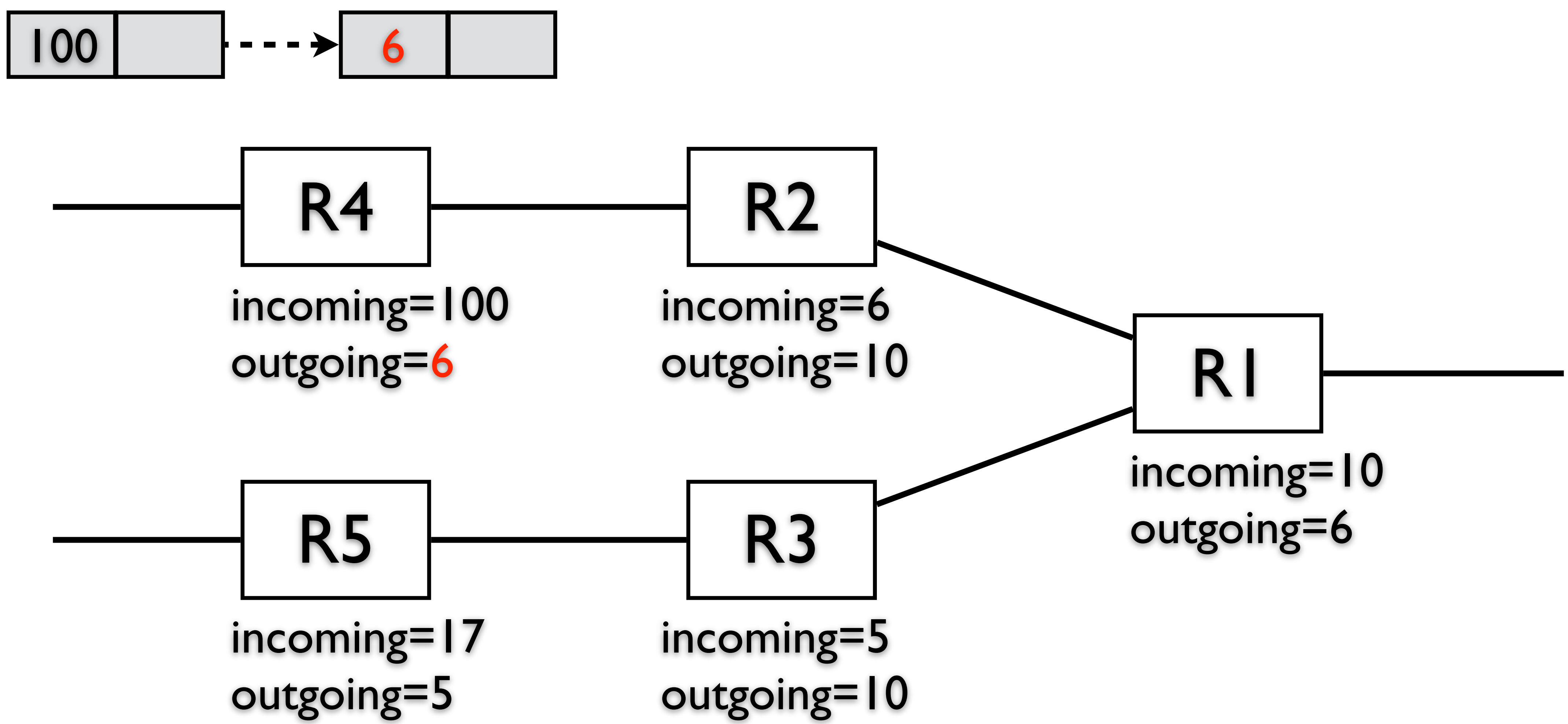
- <incoming tag, outgoing tag, outgoing interface …>



forwarding — label swapping

replaces the tag with the outgoing tag

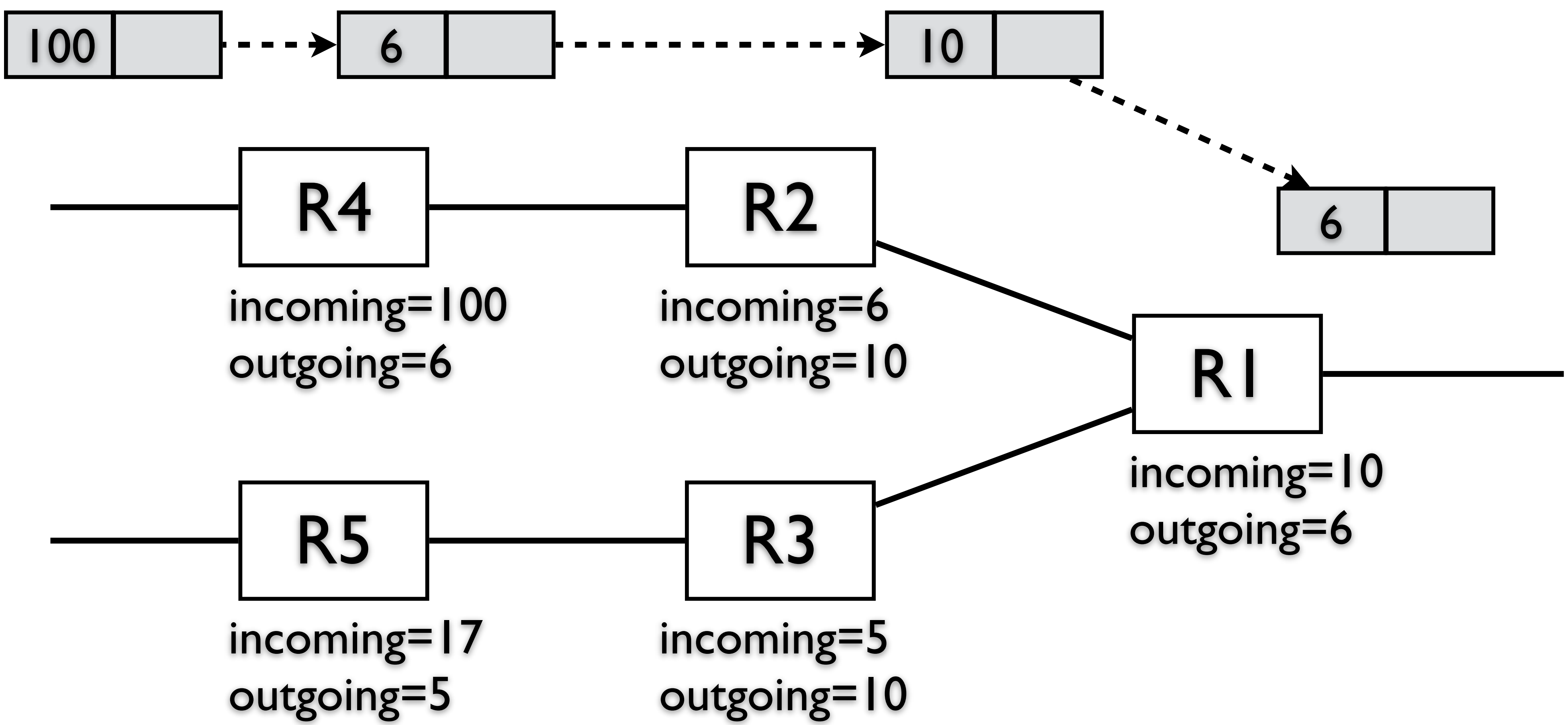
- <<u>incoming tag</u>, outgoing tag, outgoing interface …>



forwarding — label swapping

replaces the tag with the outgoing tag

- <incoming tag, outgoing tag, outgoing interface …>



high forwarding performance

label swapping enables high performance

- exact match algorithm using fixed length (20 bit)
- fairly short tag as an index

high forwarding performance

label swapping enables high performance

- exact match algorithm using fixed length (20 bit)
- fairly short tag as an index

compare: longest prefix match

high forwarding performance

label swapping enables high performance

- exact match algorithm using fixed length (20 bit)
- fairly short tag as an index

} compare: longest prefix match

simple enough to allow straightforward hardware implementation

control — tag binding

binding between a tag and network-layer route

- create a tag binding
  - allocating a tag, binding it to a route
- distribute the tag binding information among tag switches

tag binding examples

different tag binding scheme realizes different control functionalities

- destination-based routing
- flexible route (explicit routes)
- hierarchy of routing knowledge (BGP)

destination-based routing

a switch allocates tags and binds them to address prefixes in its FIB

- downstream allocation
  - the tag carried in a packet is generated and bound to a prefix by the switch at the downstream end of a link

destination-based routing

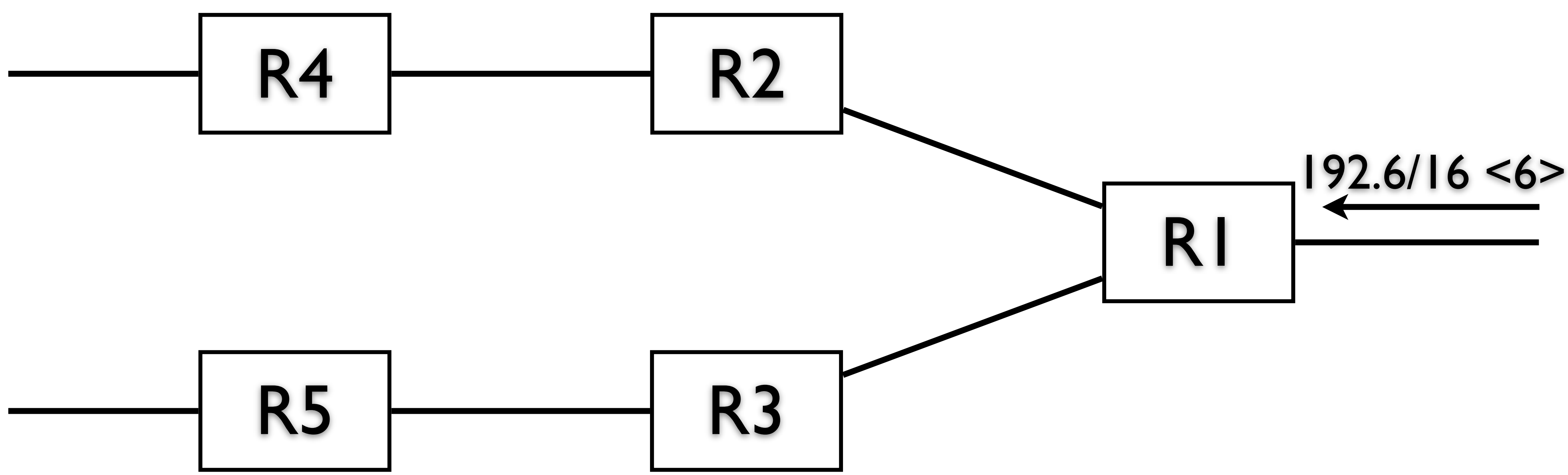
downstream allocation

- the tag carried in a packet is generated and bound to a prefix by the switch at the downstream end of a link
- for each route in the (downstream) switch's FIB
  - allocates a (incoming) tag
  - creates an entry in its TFIB
  - advertises the binding between the (incoming) tag and the route to the (upstream) other adjacent switches

destination-based routing

downstream allocation

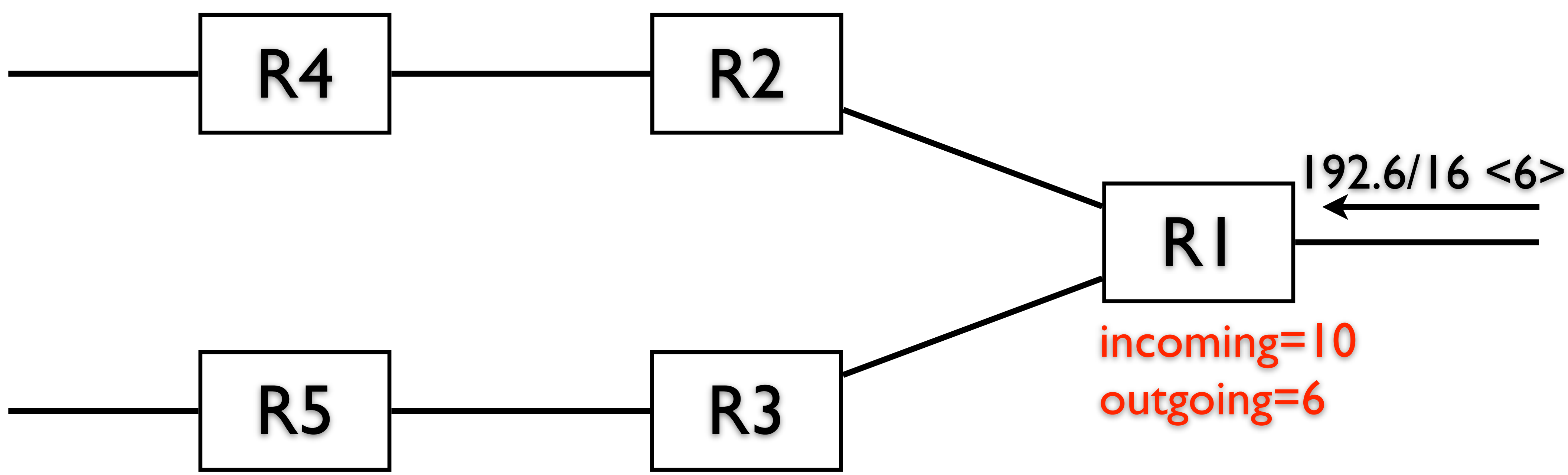
- R1 receives 192.6/16 bound to tag <6>



destination-based routing

R1 receives 192.6/16 with tag <6>

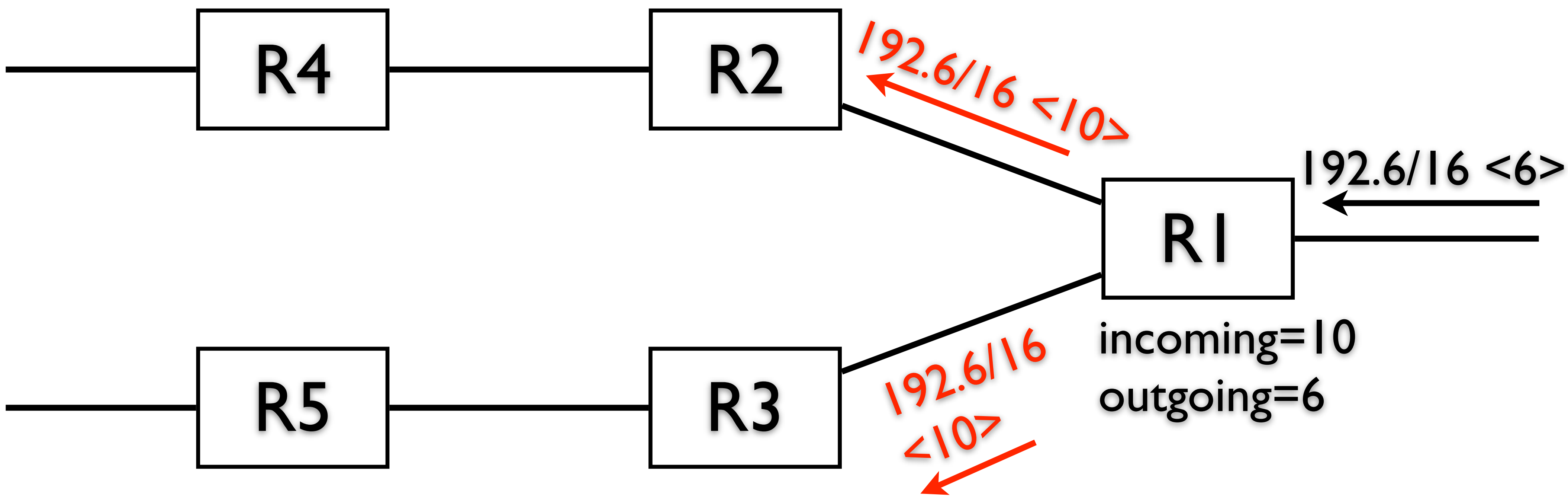
- creates an entry in TFIB, sets outgoing tag to <6>
- generates a local tag <10>, sets incoming tag to <10>



destination-based routing

R1 receives 192.6/16 with tag <6>

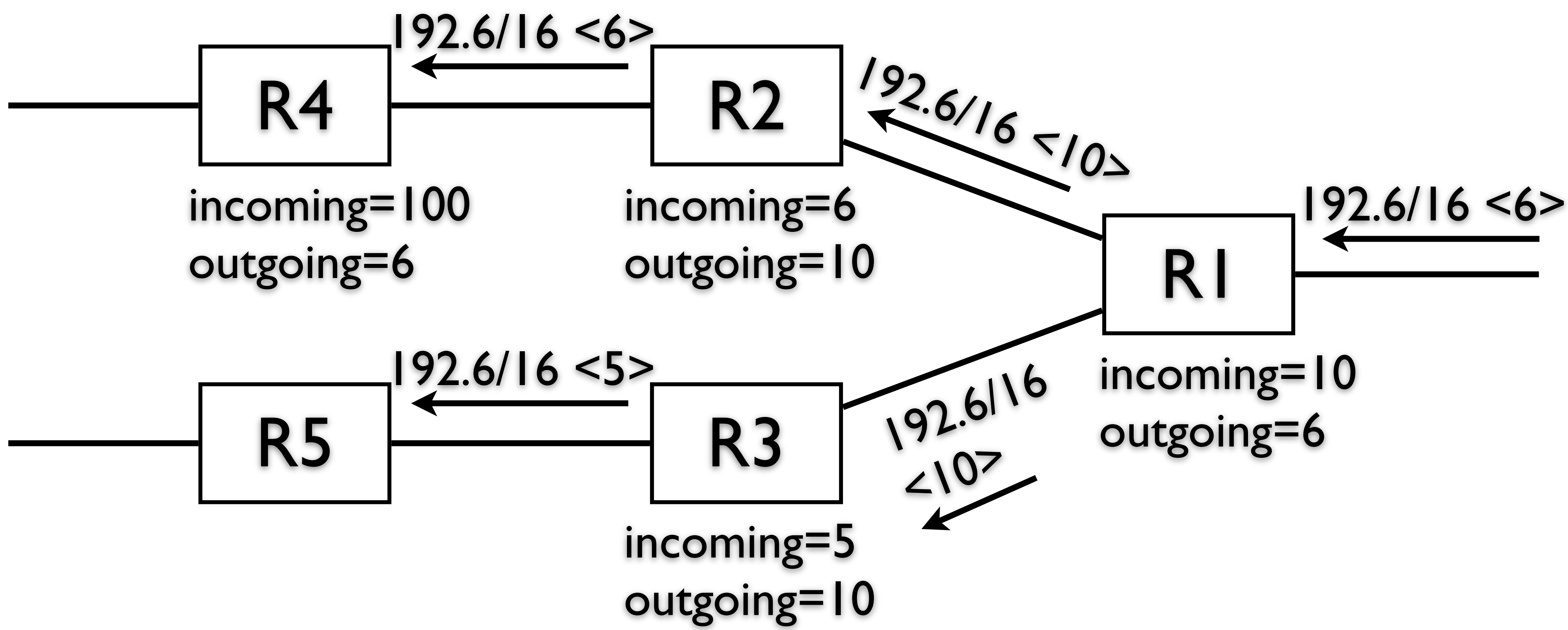
- set outgoing tag to <6>, set incoming tag to <10>
- advertises 192.6/16 with <10> to others



destination-based routing

similarly, R2, R3, R4

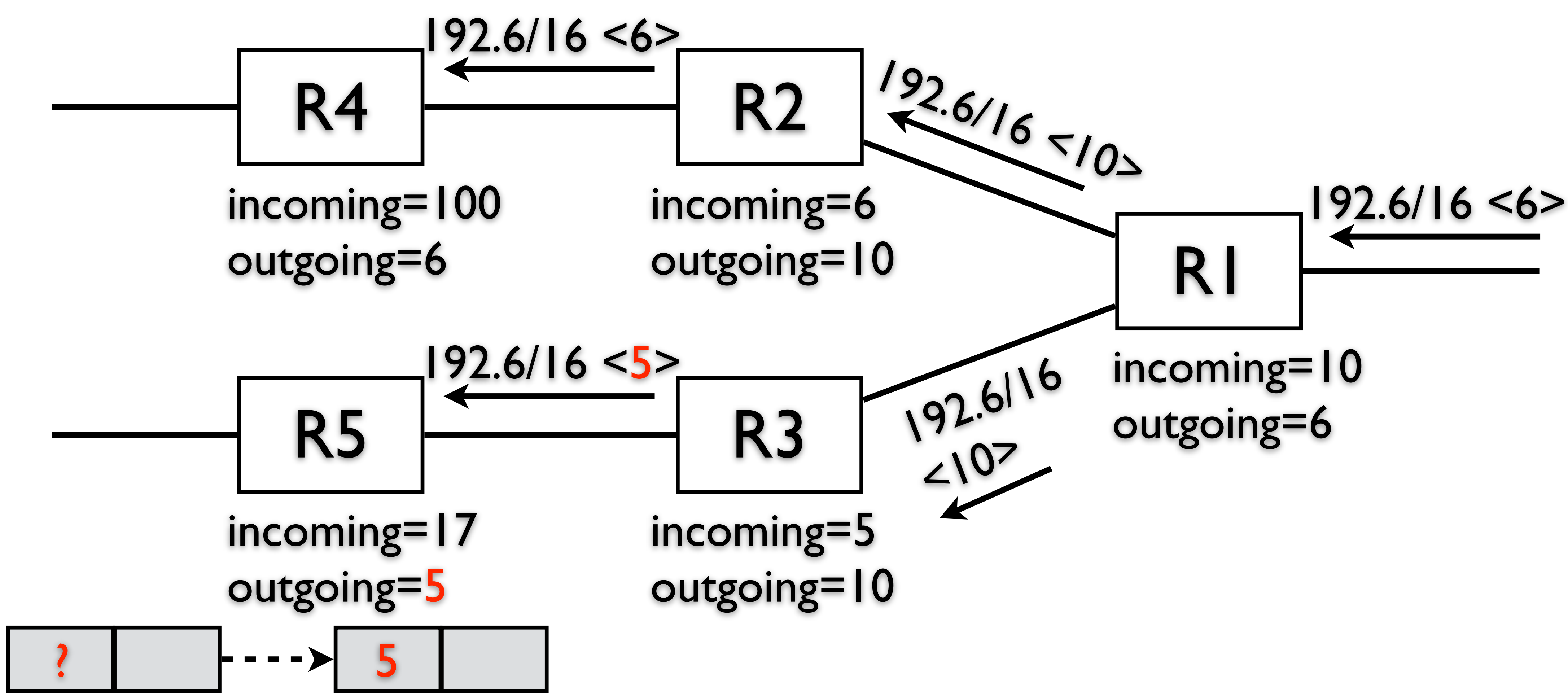
- receive tag binding, create TFIB entries, re-advertise



destination-based routing

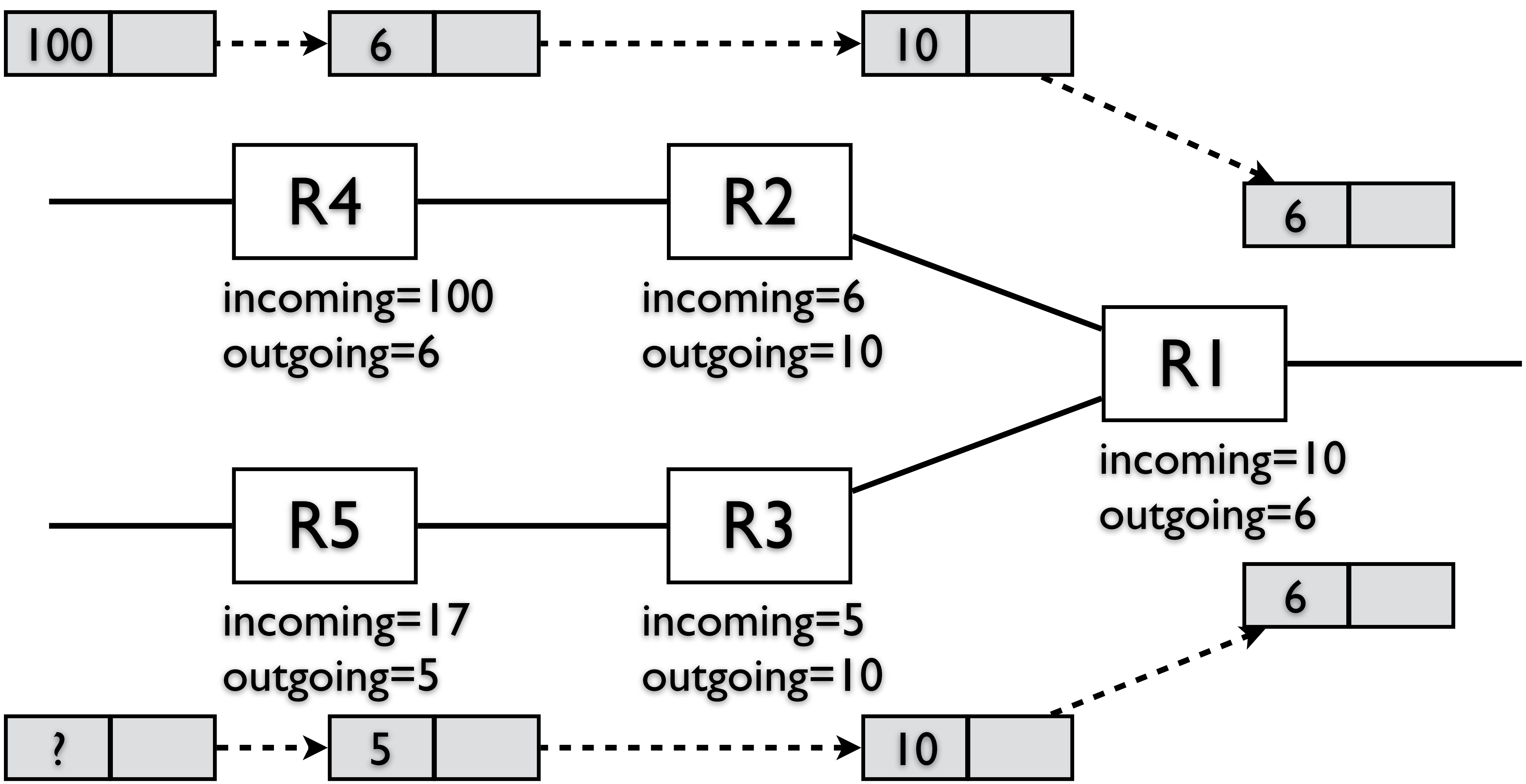
R5, router left to which is not a tag switch

- R5 also augments its FIB with outgoing tag <5>



destination-based routing

a switch allocates tags and binds them to address prefixes in its FIB



observation — routes aggregation

tag allocation is topology-driven

- if a tag switch forwards multiple packets to the same next-hop neighbor
  - only a single (incoming) tag is needed
- if a tag switch receives a set of routes associated with a single tag
  - only a single (incoming) tag is needed

scaling properties

tag switching used for destination-based routing

of tags a switch maintains

# of routes in the FIB

scaling properties

tag switching used for destination-based routing

of tags a switch maintains << # of routes in the FIB

scaling properties

tag switching used for destination-based routing

of tags a switch maintains << # of routes in the FIB

tag associated with routes, rather than flows

- much less state required
- no need to perform flow classicification

scaling properties

tag switching used for destination-based routing

of tags a switch maintains << # of routes in the FIB

tag associated with routes, rather than flows

- much less state required
- no need to perform flow classicification

more robust & stable destination-based routing in the presence of traffic pattern change

flexible routing (explicit routes)

provides forwarding along the paths different from the path determined by destination-based routing

- install tag binding in tag switches that do not correspond to the destination based routing paths

hierarchical routing (BGP)

Internet routing (BGP)

- 2-tier routing scheme, collection of routing domains

tag switching

- decouples interior (intra-) and exterior (inter-) routing
- significantly reduces load on non-border switches
- only border maintains routing information for both interior/exterior routing

hierarchical routing (BGP)

tag stack

- a set of tags carried by a packet organized as a stack

operations

- label swapping as before: swap tag at the top

hierarchical routing (BGP)

tag stack

- a set of tags carried by a packet organized as a stack

operations

- label swapping as before: swap tag at the top
- pop the stack
- push one more tag into the stack

hierarchical routing (BGP)

when a packet is forwarded between two border tag switches in different domains

- the tag stack only has one tag, associated with the AS-level route

hierarchical routing (BGP)

when a packet is forwarded between two border tag switches in different domains

- the tag stack only has one tag, associated with the AS-level route

when a packet is forwarded within a domain

- ingress router: 2nd tag associated with an interior route to the egress border is pushed
- internal switches: only operate on the 2nd top tag
- egress border: pop the top (2nd) tag, uses the original tag for tag switching to routers in another domain

Fabric: A Retrospective on Evolving SDN

http://yuba.stanford.edu/~casado/fabric.pdf

Fabric:
end to end arguments + MPLS

many proposals towards a better network

MPLS

- simplifies hardware + improves control flexibility

SDN attempts to make further progress but suffers certain shortcomings

- can we overcome those shortcomings by adopting the insights underlying MPLS?

an ideal network

hardware

- simple (inexpensive)
- vendor-neutral
- future proof: accommodate future innovation as much as possible

control

- flexible: meet future requirements as they arise

review

original Internet, MPLS, SDN along two dimensions

- requirements
- interfaces

requirements

two sources

- hosts
- operators

hosts

- want their packets to travel to a particular destination with some QoS requirement about the nature of the services these packets receive en-route to the destination

operators

- TE, tunneling, virtualization, isolation, …

interfaces

places where control information pass between network entities

- host-network
  - *how hosts inform the network of their requirements*
  - e.g., packet header (destination address), …

interfaces

places where control information pass between network entities

- host-network
  - *how hosts inform the network of their requirements*
  - e.g., packet header (destination address), …
- operator-network
  - how operator informs the network of their requirements
  - e.g., per-box configuration command

interfaces

places where control information pass between network entities

- host-network
  - *how hosts inform the network of their requirements*
  - e.g., packet header (destination address), …
- operator-network
  - how operator informs the network of their requirements
  - e.g., per-box configuration command
- packet-switch
  - how a packet identifies itself to a switch
  - e.g., packet header as an index into the forwarding table

original Internet VS. *MPLS* VS. *SDN*

	host-network interface	operator-network interface	packet-switch interface
original Internet	destination address	none	destination address
MPLS	packet header (inspected by edge tag switch)	none	label (used by internal tag switch)
SDN	packet header (Openflow)	fully programmatic interface (network abstractions)	packet header (Openflow)

original Internet VS. *MPLS* VS. *SDN*

	host-network interface	operator-network interface	packet-switch interface
original Internet	destination address	none	destination address
MPLS	packet header (inspected by edge tag switch)	none	label (used by internal tag switch)
SDN	packet header (Openflow)	fully programmatic interface (network abstractions)	packet header (Openflow)

shortcomings of SDN

not fulfill the promise of simple hardware

- Openflow far more complex than the tens of bits MPLS

host requirements generality expected to increase

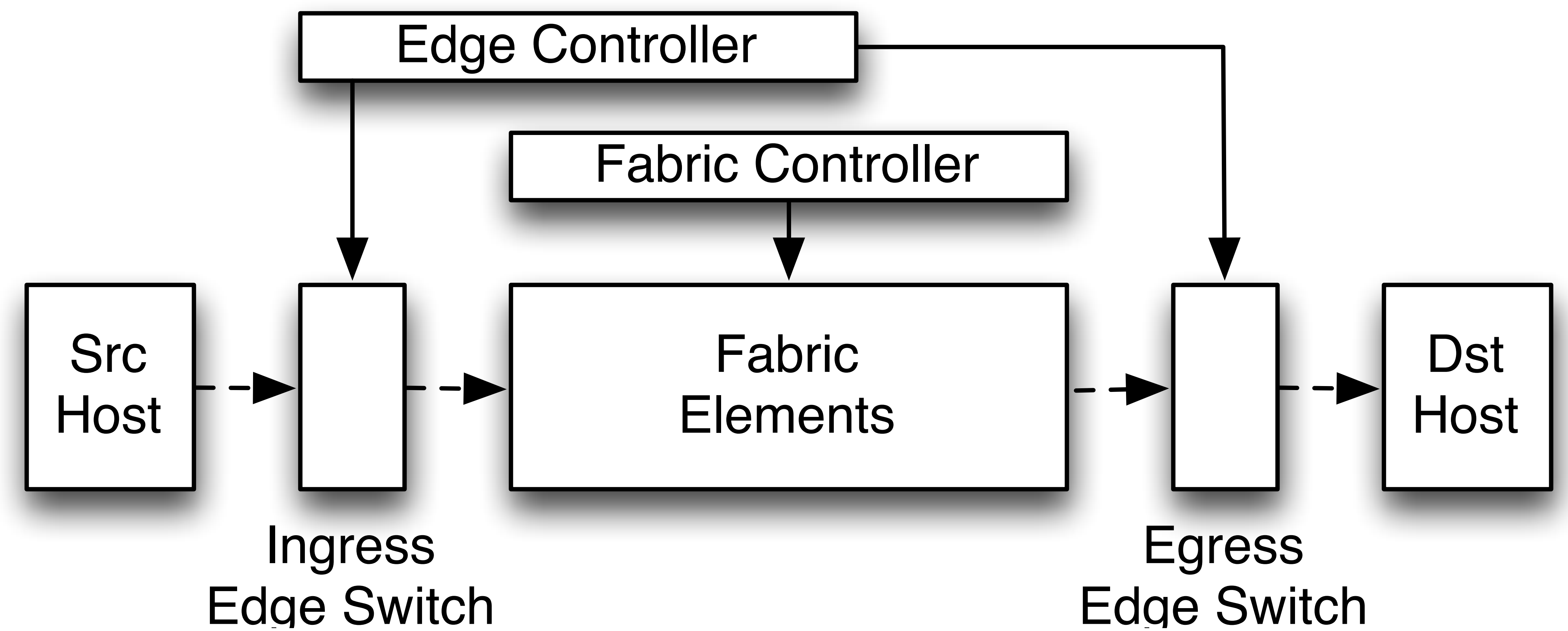
- in turn means the generality of the host-network interface will increase, but the increased generality must also be present to every switch

unnecessary coupling the host requirements to the network core behavior

extending SDN with MPLS inspiration

SDN architecture should incorporate "fabric"

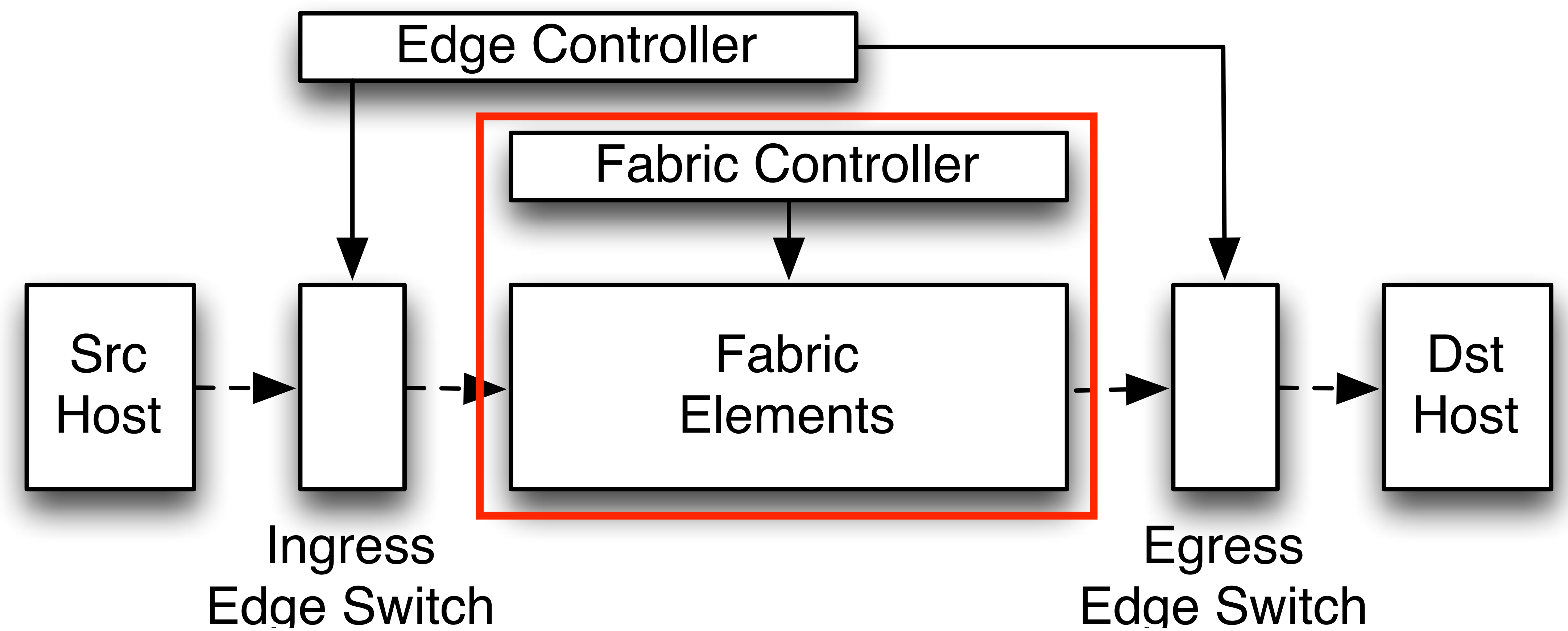
- fabric is a transport element



extending SDN with MPLS inspiration

SDN architecture should incorporate “fabric”

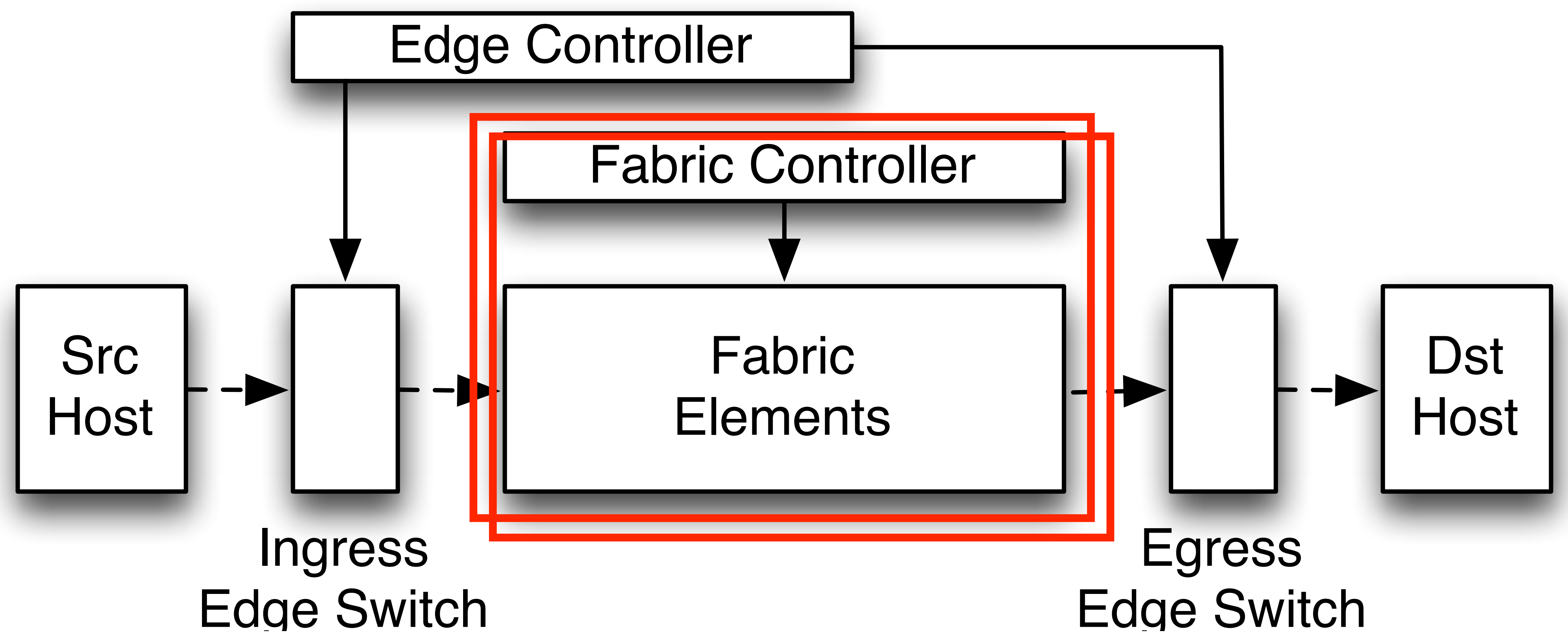
- fabric is a transport element



extending SDN with MPLS inspiration

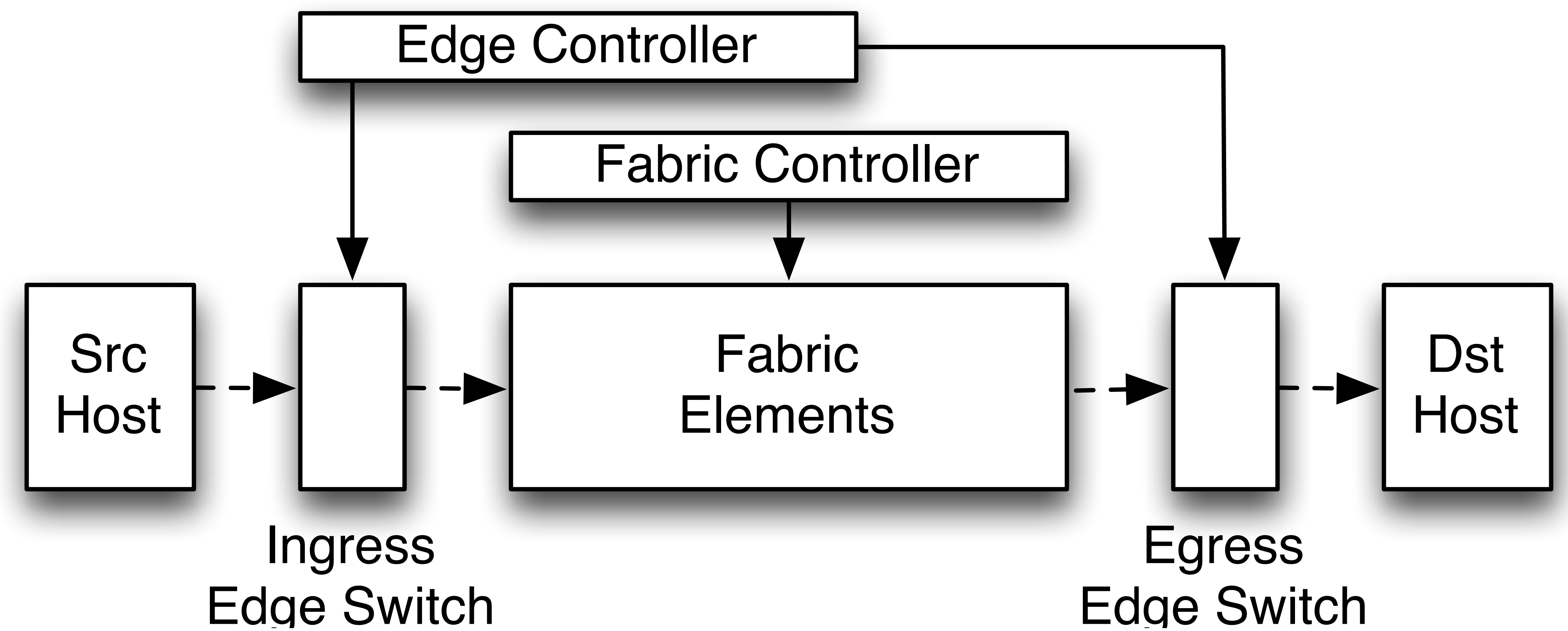
SDN architecture should incorporate “fabric”

- fabric is a transport element



extending SDN with MPLS inspiration

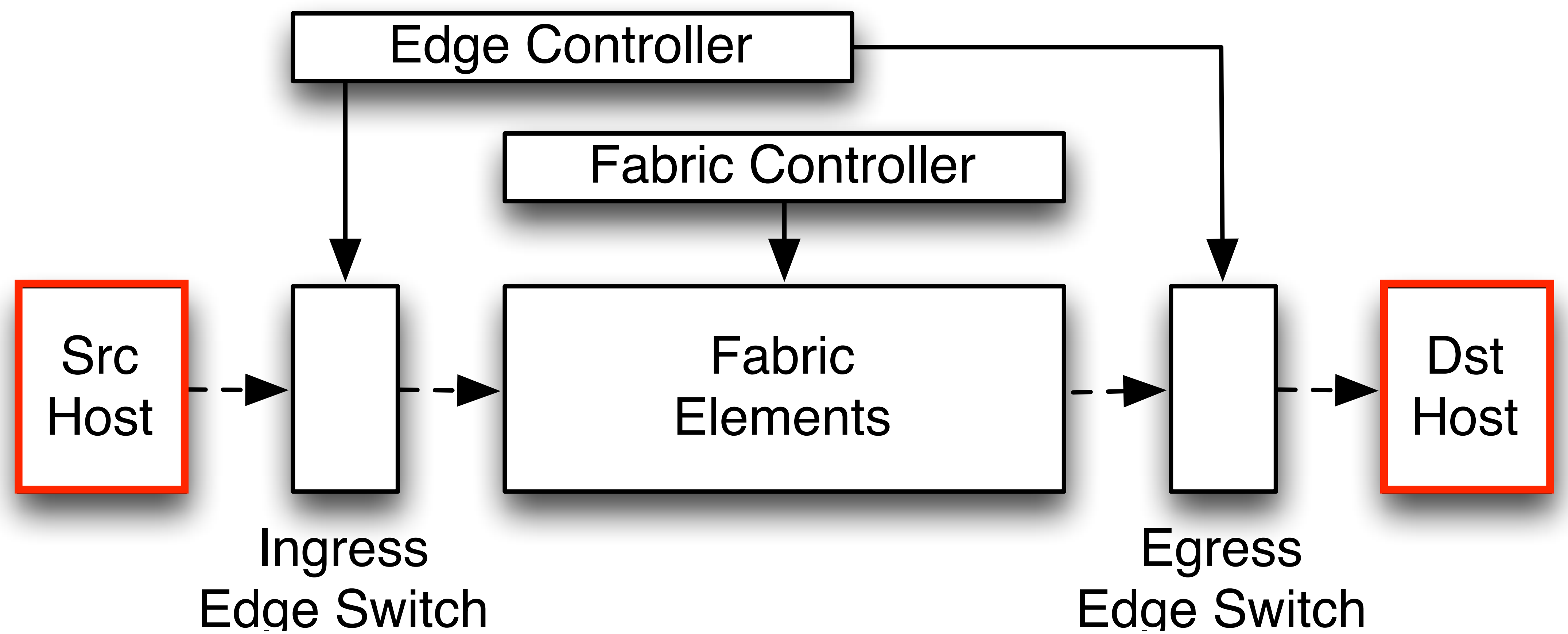
three components: hosts, edge (ingress, egress), fabric (core)



extending SDN with MPLS inspiration

host

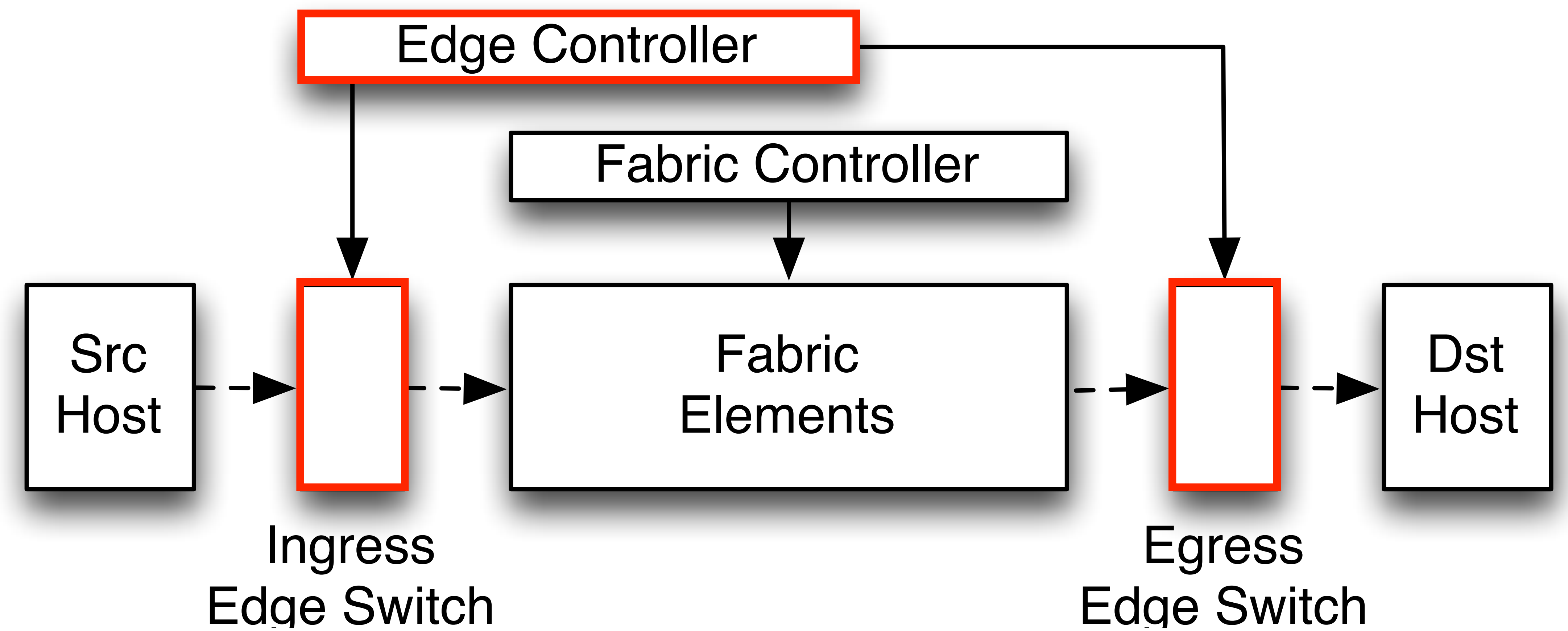
- generator and destination of traffic



extending SDN with MPLS inspiration

edge

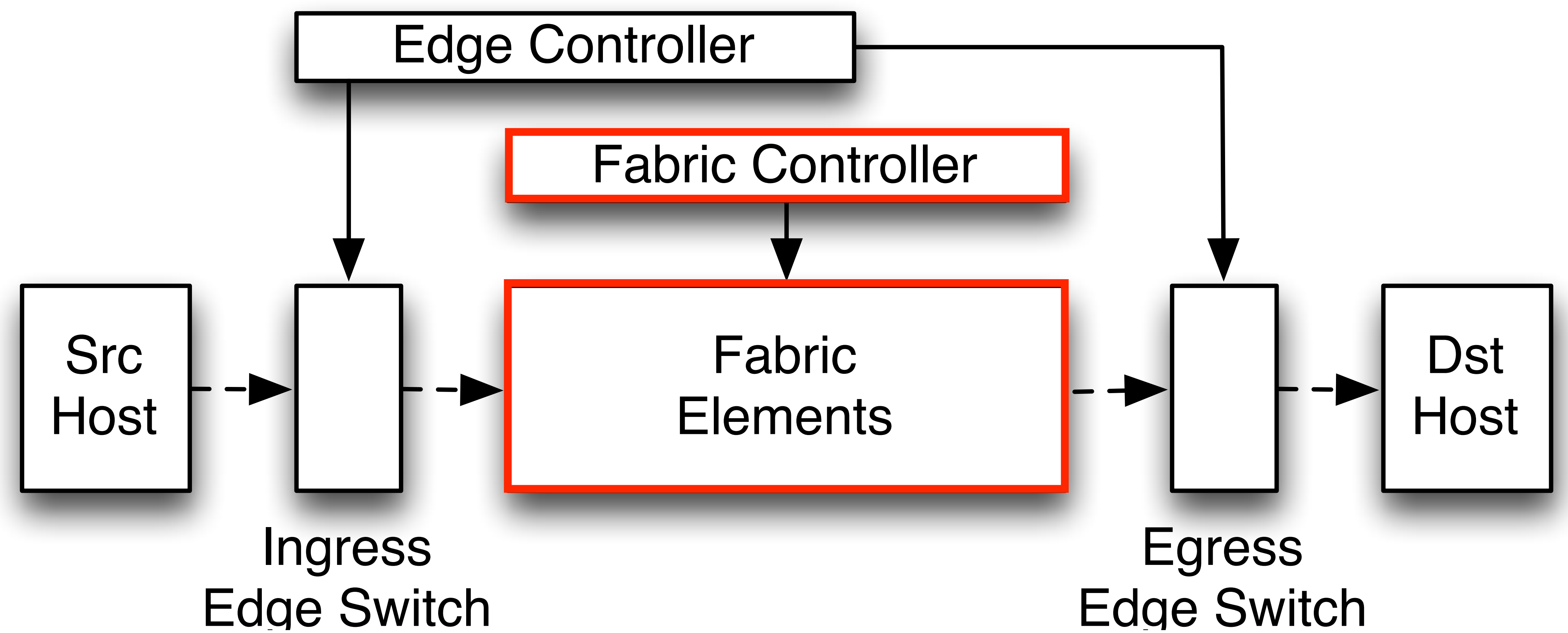
- (ingress + edge controller) provide the host-network interface
- edge controller provides operator-network interface



extending SDN with MPLS inspiration

fabric

- packet-switch interface (packet transfer alone)



extending SDN with MPLS inspiration

simplifies hardware + improves control flexibility