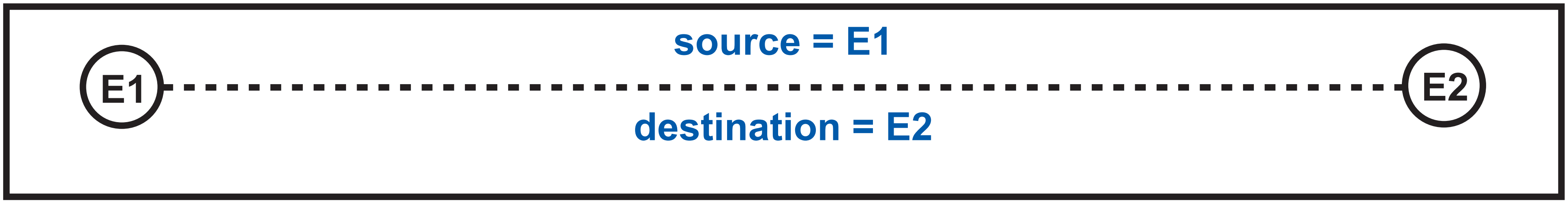
PATTERNS IN NETWORK ARCHITECTURE:

MIDDLEBOXES

MORE ABOUT BRIDGING

PROBLEM: YOU HAVE A HIGH-LEVEL, SPECIALIZED NETWORK TO IMPLEMENT



The network has no persistent links—need to create a dynamic link between endpoints.

There is no single network on which to implement this link.

However, there is a pair of bridged networks that might together implement it.

What are the problems and how can they be overcome?

To maximize the potential of bridging, we allow bridged networks to be as independent as possible.

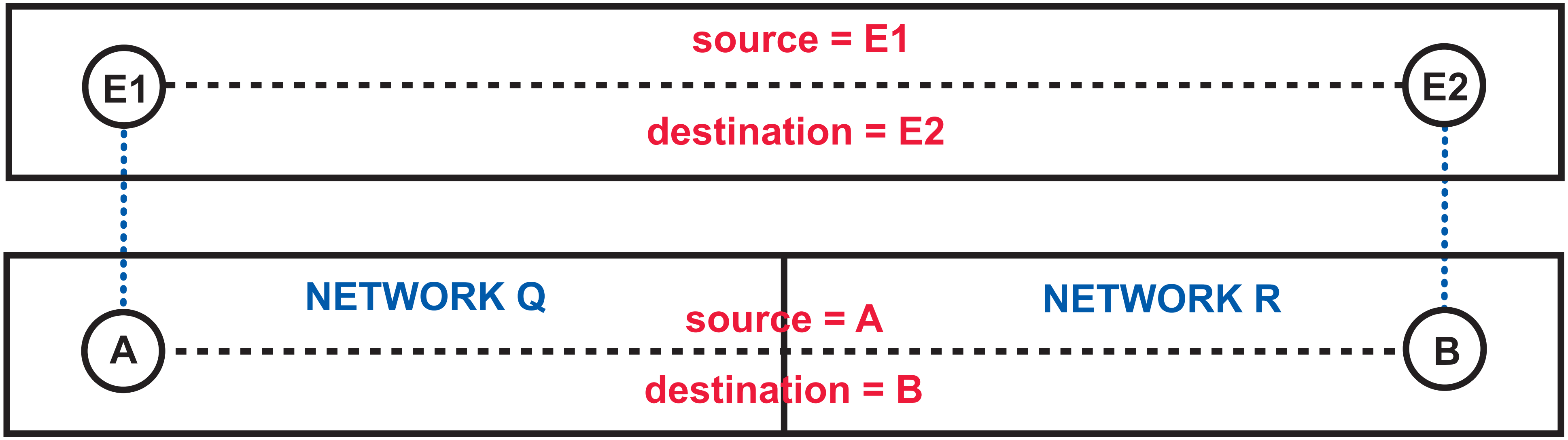
The two must share:

- one or more session protocols
- one or more nodes

The two may share:

- routing
- forwarding at shared nodes

MORE ABOUT BRIDGING 2



in normal layering, E1 requests implementation of a link to E2

A looks up E2 in the Q directory, finds it is located at B, and creates a session from A to B

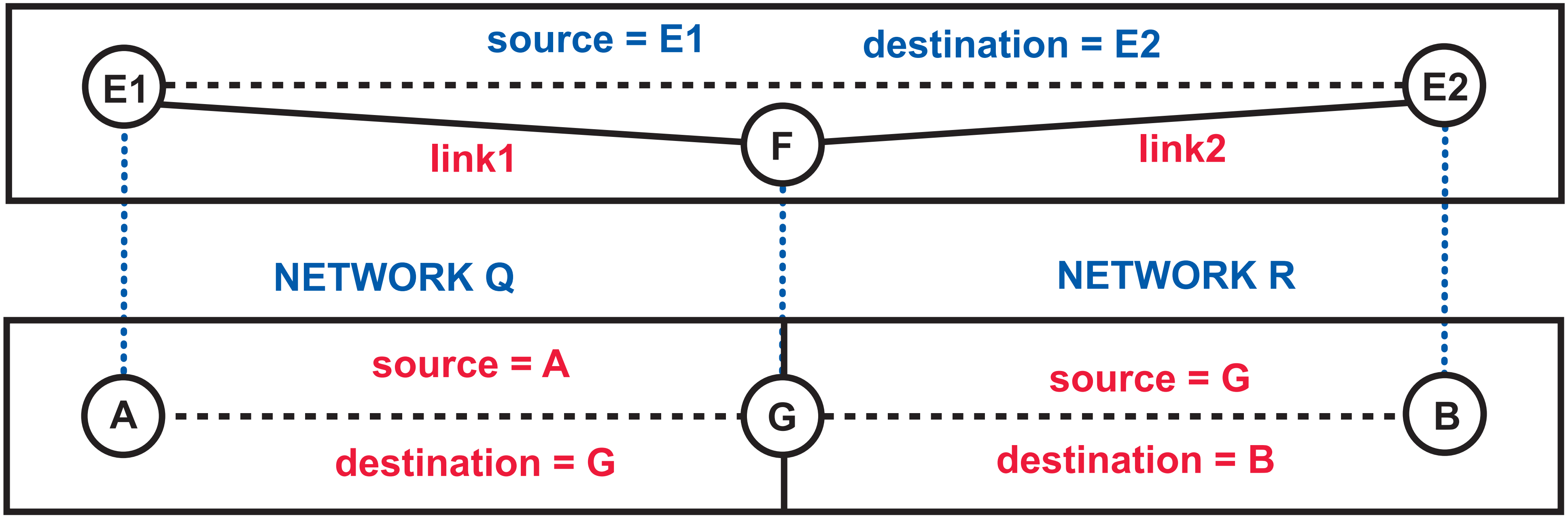
this may still work!

Problem 1: B is not routable in Q, meaning that forwarding tables have no entries for it—Q and R may even have incompatible namespaces.

Problem 2: B may have a different meaning in the context of Q—it is not globally unique across the bridged networks.

Lookup in Q's directory MUST result in a name that is routable in Q.

BRIDGING 3: ONE SOLUTION TO THE PROBLEMS



In Q, the directory locates E2 at the bridging gateway G, which is the location of F in the overlay network.

of course, G is routable in network Q

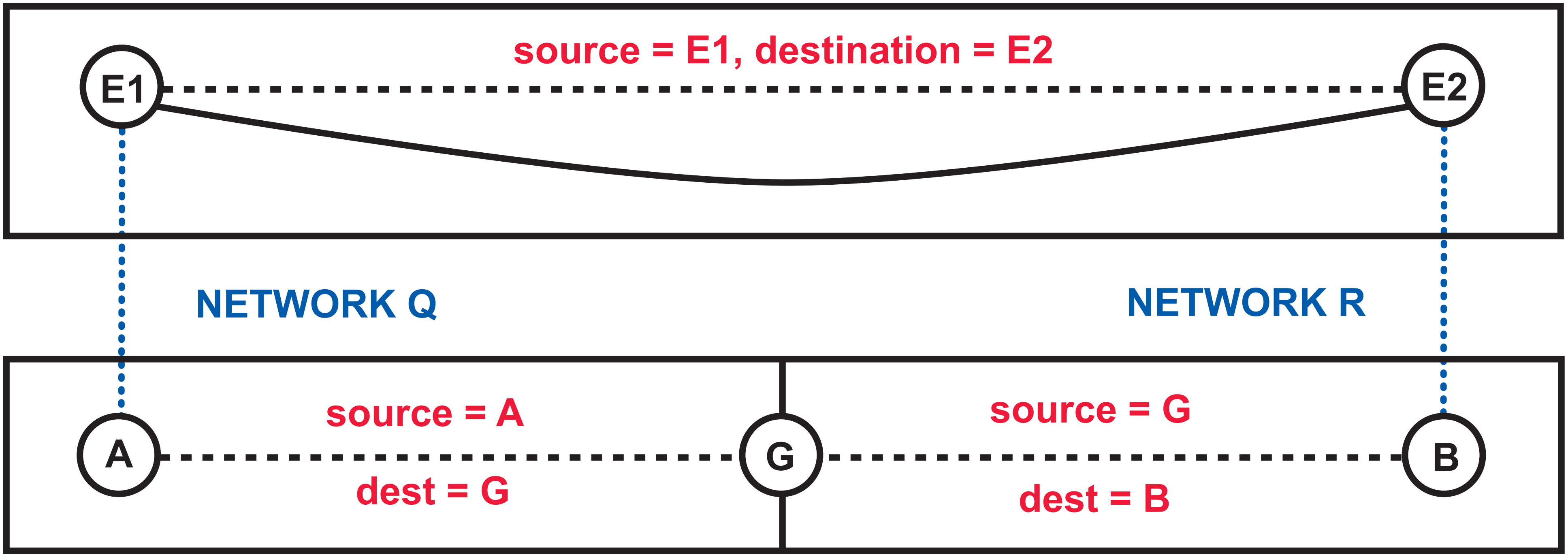
The session in Q ends at G.

Dynamic link1 in the overlay ends at router F.

F is a router. When F receives the initial packet for the session on link1, it sends it to new dynamic link2.

Link2 is set up in R, where B is unique and routable.

BRIDGING 4: ANOTHER SOLUTION TO THE PROBLEMS



in Q, directory lookup of E2 returns the list [G, B], and only the first name G needs to be unique and routable

when the session-initiation packet gets to G, G forms a *compound session*

the internal state of G maintains the mapping between *simple sessions* A to G and G to B

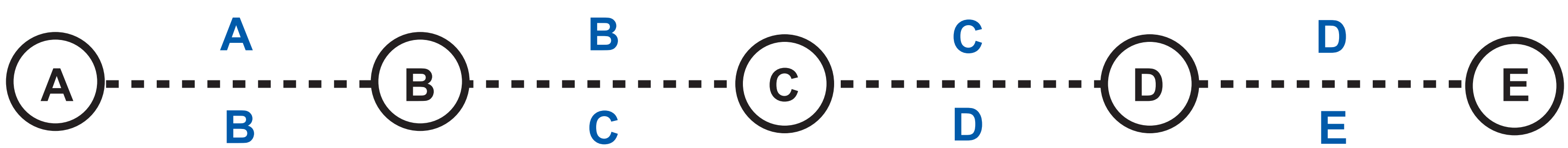
if G is a NAT, it forms a compound session because A is not unique and routable in R!

compound sessions can also be used to insert middleboxes in a session

MORE ABOUT BRIDGING 5: COMPOUND SESSIONS

COMPOUND SESSIONS ARE VERY POWERFUL

- session-initiation packet can carry a list of nodes to be visited, e.g., [A, B, C, D]
- any node in the session can add nodes to the unvisited part of the list, e.g., at C, the list becomes [A, B, C, D, E] or [A, B, C, F, D]
- any node in the session can hide visited parts of the list, e.g., at C the list becomes [C, D]



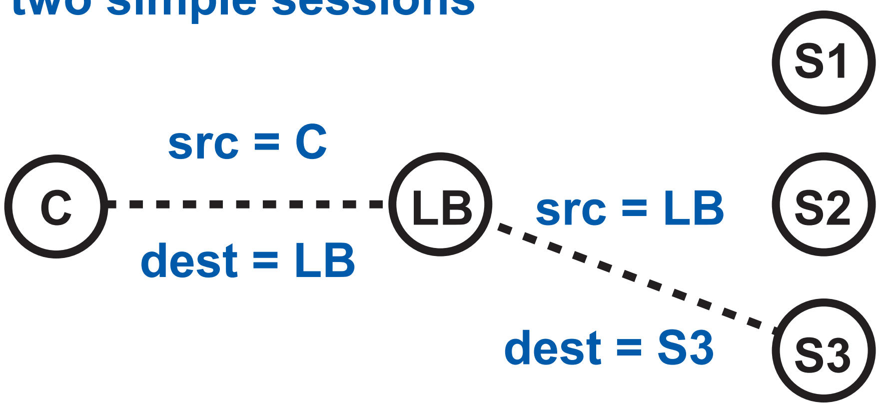
COMPOUND SESSION PROTOCOLS

IS THE SIGNALING . . .

. . . END-TO-END?

“Layer 4 Load Balancer”

LB rewrites TCP/IP headers in both directions, using its internal mapping between the two simple sessions



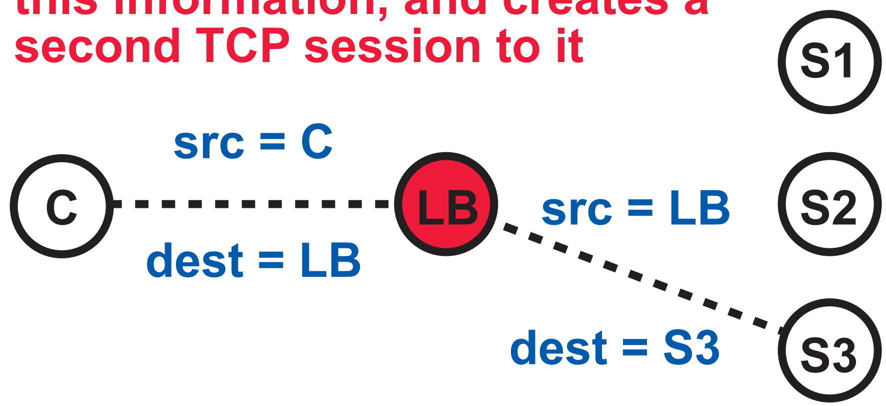
except for this interference, the TCP handshake takes place end-to-end

. . . OR PIECEWISE?

“Layer 7 Load Balancer”

LB completes the TCP handshake, so that it can get some data from C, including the HTTP request

LB then chooses a server based on this information, and creates a second TCP session to it



middleboxes that are signaling endpoints are very important in voice-over-IP

they can do big things, like make a conference with a third party

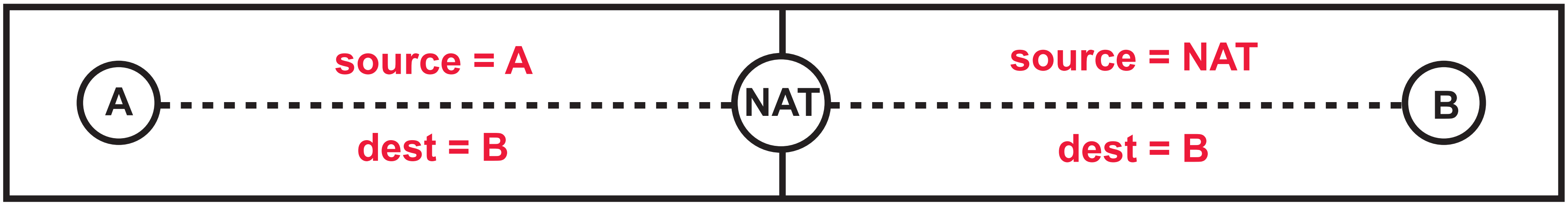
WHAT NATs BREAK

THE INTERNET WAS DESIGNED TO EMPOWER USERS AND ENCOURAGE INNOVATION (a philosophy reflected in the “end-to-end principle”) . . .

. . . FROM THIS PERSPECTIVE, IT IS VERY DIFFICULT TO IMAGINE A WORSE ADDITION THAN PRIVATE ADDRESS SPACES AND NAT

PRIVATE NETWORK

PUBLIC INTERNET



- only allows client/server communication, not peer-to-peer—no public node can find or initiate communication to a private node
- applications cannot talk about private nodes, either, e.g., set up communication state for them

this is a huge problem in VoIP

- NATs are a single point of failure, drop compound sessions prematurely, can run out of resources

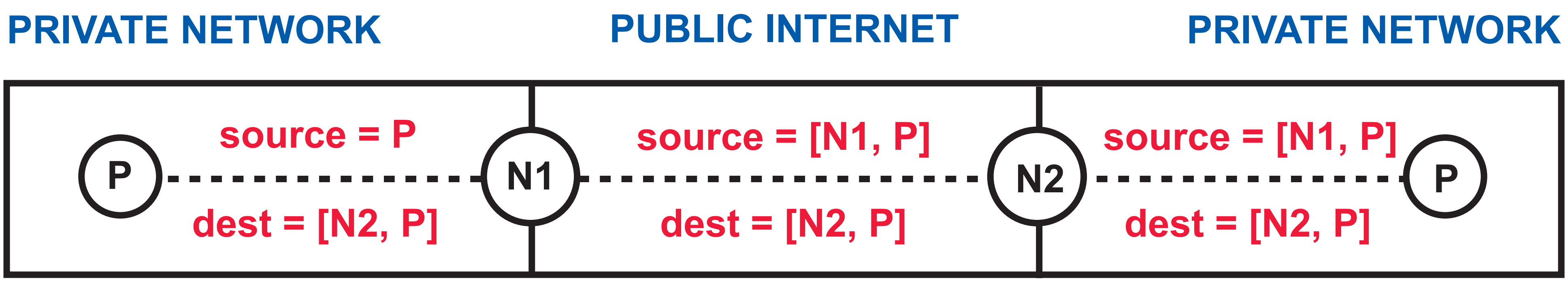
globally-unique addresses would also be extremely useful for . . .

- keys in data stores
- logging and debugging
- programming reliable (redundant) systems

NATs are “the idea that launched a thousand hacks”

COMPOUND SESSIONS ARE SO POWERFUL . . .

. . . THAT A VERY GENERAL IMPLEMENTATION OF THEM COULD SOLVE THE PROBLEMS INTRODUCED BY PRIVATE ADDRESS SPACES AND NAT . . .



. . . BUT WE HAVEN'T ANALYZED ALL THE CONSEQUENCES

NOTE: There is a next-generation Internet proposal in which the data structure is not a list but a DAG! In the graph, transitions out of the current routing state are next hops from here, and the transitions are ordered for first-choice vs. backup

DISCUSSION OF

“SIMPLE-fying Middlebox Enforcement Policy Using SDN”

THREE PROBLEMS, THREE SOLUTIONS

so far I understand (or believe) 1.5 of them

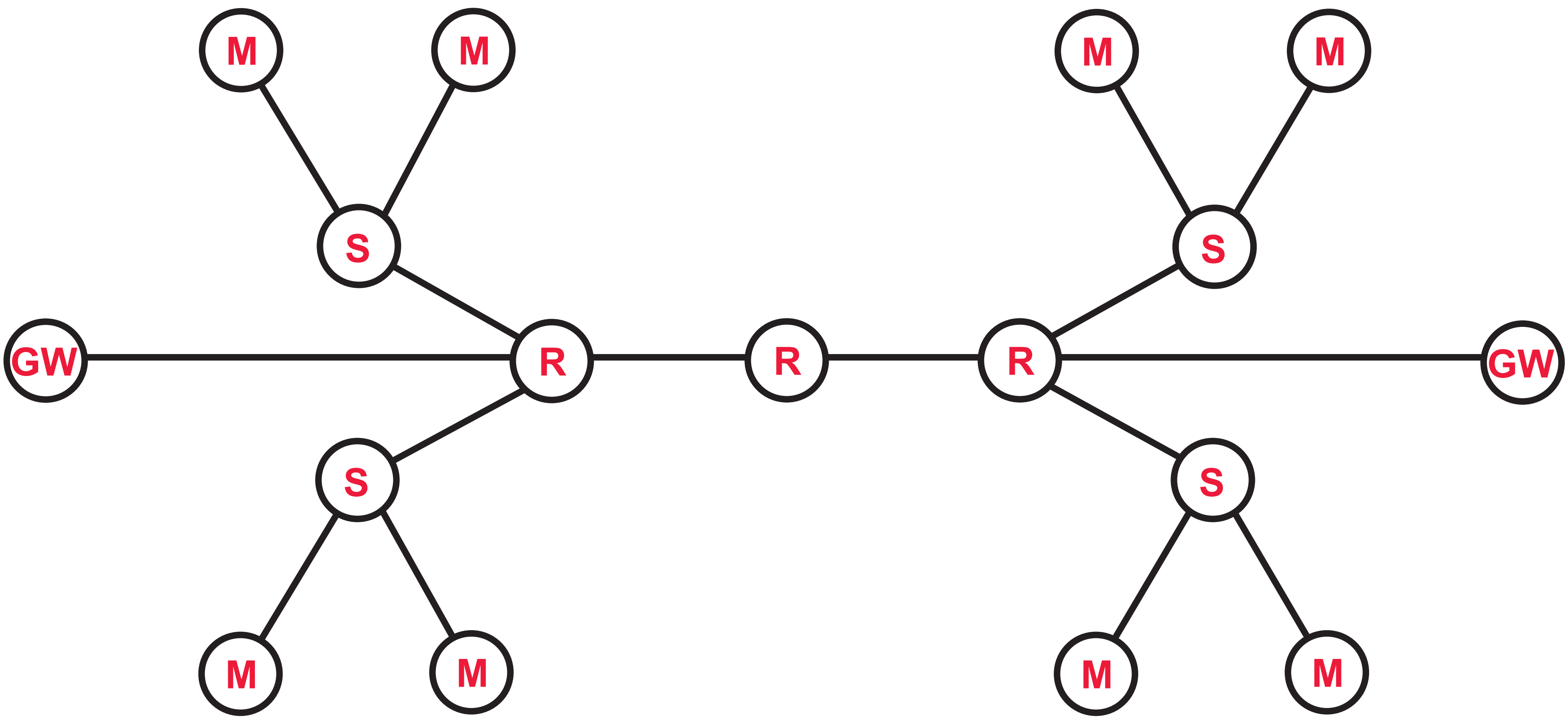
THE OTHER SIDE OF COMPOSITION IS DECOMPOSITION

POLICY: < F, M1, M2, M3 >

F is a header pattern that identifies a flow

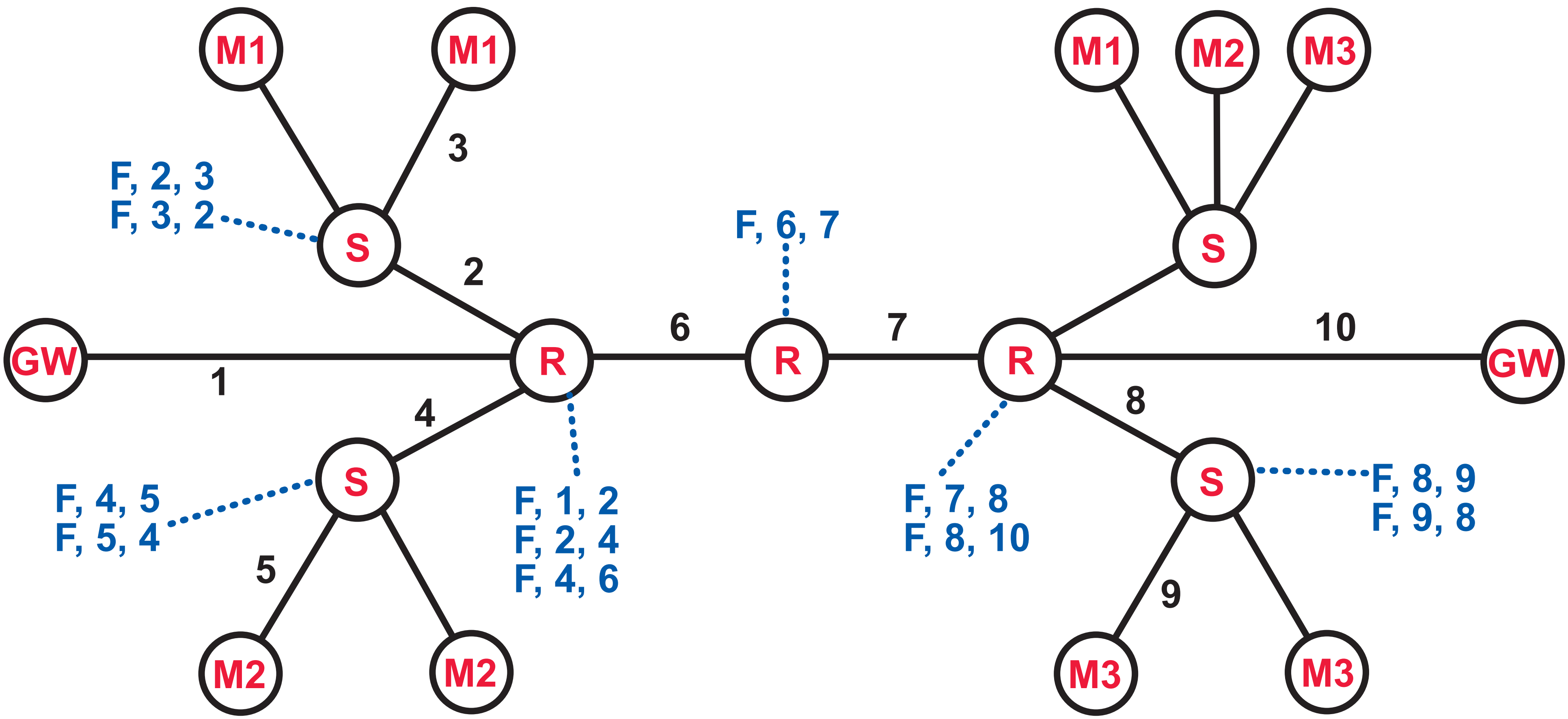
packets of flow should go through middleboxes of types M1, M2, M3, in that order

POLICY MUST BE ENFORCED BY A DATA CENTER NETWORK



THE OTHER SIDE OF COMPOSITION IS DECOMPOSITION 2

POLICY: < F, M1, M2, M3 >



THE ROUTERS HAVE
A LOT OF RULES!

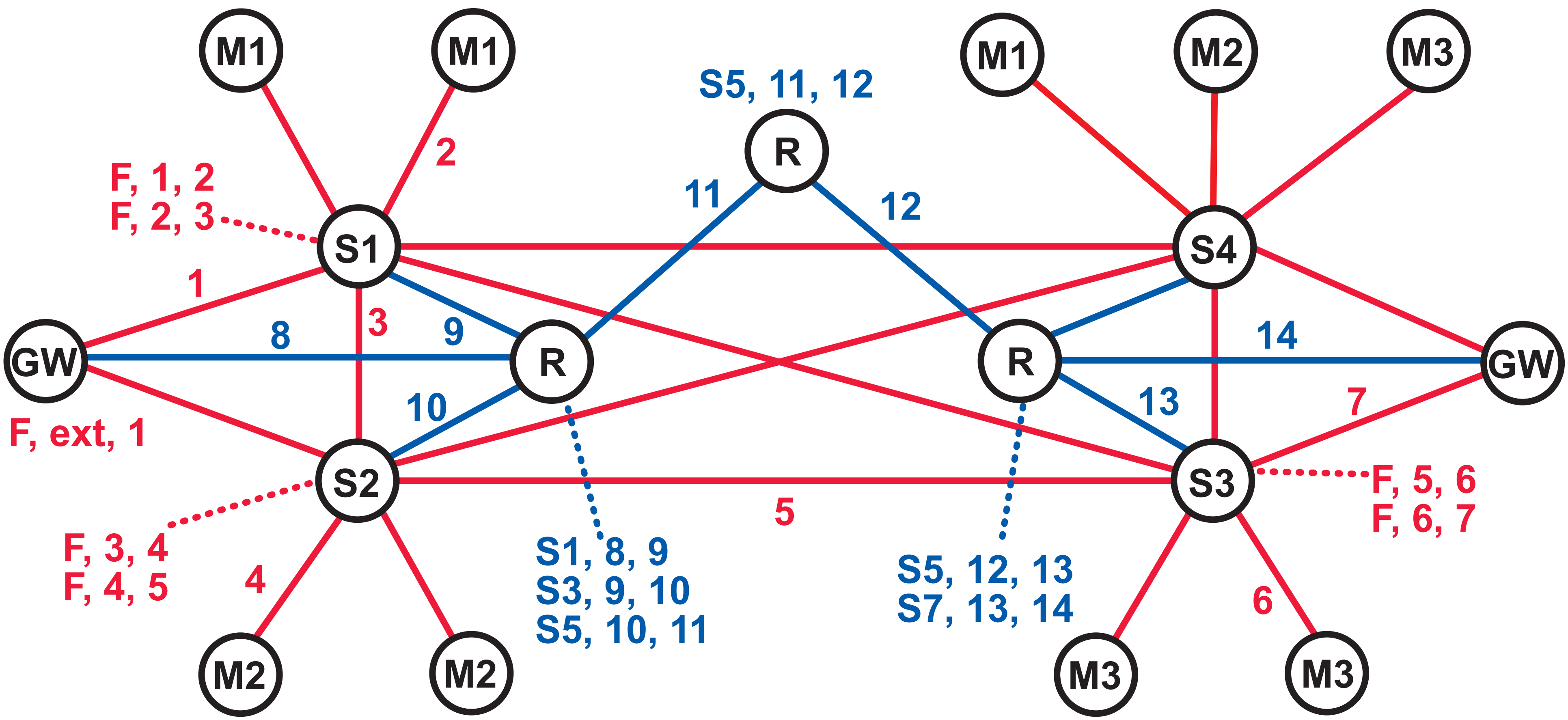
SOURCES OF CHANGE

- physical topology
- node and link failures
- policy changes
- fluctuation in load

THE OTHER SIDE OF COMPOSITION IS DECOMPOSITION 3

THE RED NETWORK (OVERLAY) IS FOR SERVICE CHAINING

CAUSES FOR CHANGE: policy changes, fluctuations in load, switch or middlebox failures



THE BLUE NETWORK (UNDERLAY) IS FOR REACHABILITY (NO PER-FLOW STATE)

CAUSES FOR CHANGE: topology changes, physical link or router failures

Dynamic Service Chaining

with

DYSCO

Pamela Zave

AT&T Advanced Technology

Ronaldo A. Ferreira, X. Kelvin Zou, Masaharu Morimoto, and Jennifer Rexford

Princeton University

SERVICE CHAINING IS A BIG CHALLENGE EVEN WITH SDN

steering traffic through middleboxes or network functions (NFs)

FINE-GRAINED FORWARDING RULES

- need switch-level state that grows with the . . .

  . . . diversity of policies
  . . . difficulty of classifying traffic
  . . . length of service chains
  . . . number of instances per middlebox type

- need real-time response from the central controller to handle frequent events

  e.g., new middlebox instances, link failures

- are insufficient to handle . . .

  . . . session affinity
  . . . service chaining across administrative boundaries
  . . . middleboxes that modify the 5-tuple used to identify packets
  . . . middleboxes that classify packets

ENCAPSULATION FORMATS

e.g., Contrail service chaining, Network Services Header

a step in the right direction!

traffic forwarded by destination address alone,

. . . so service chaining is independent of routing,

. . . but there are still many limitations

DYSCO HANDLES ALL OF THESE CASES, PLUS DYNAMIC SERVICE CHAINING

reconfigure an ongoing session

INSERT . . . a packet scrubber when intrusion detection raises an alarm

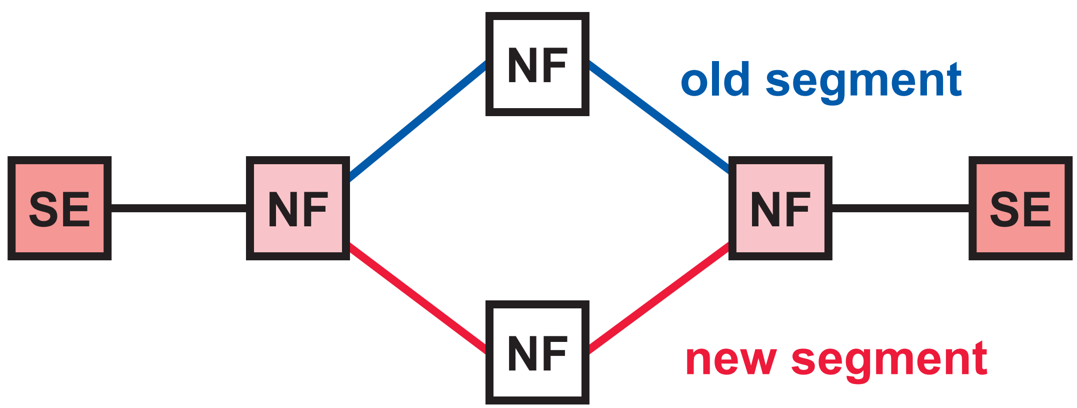
. . . a video transcoder during periods of network congestion

DELETE . . . a load balancer after the server has been chosen

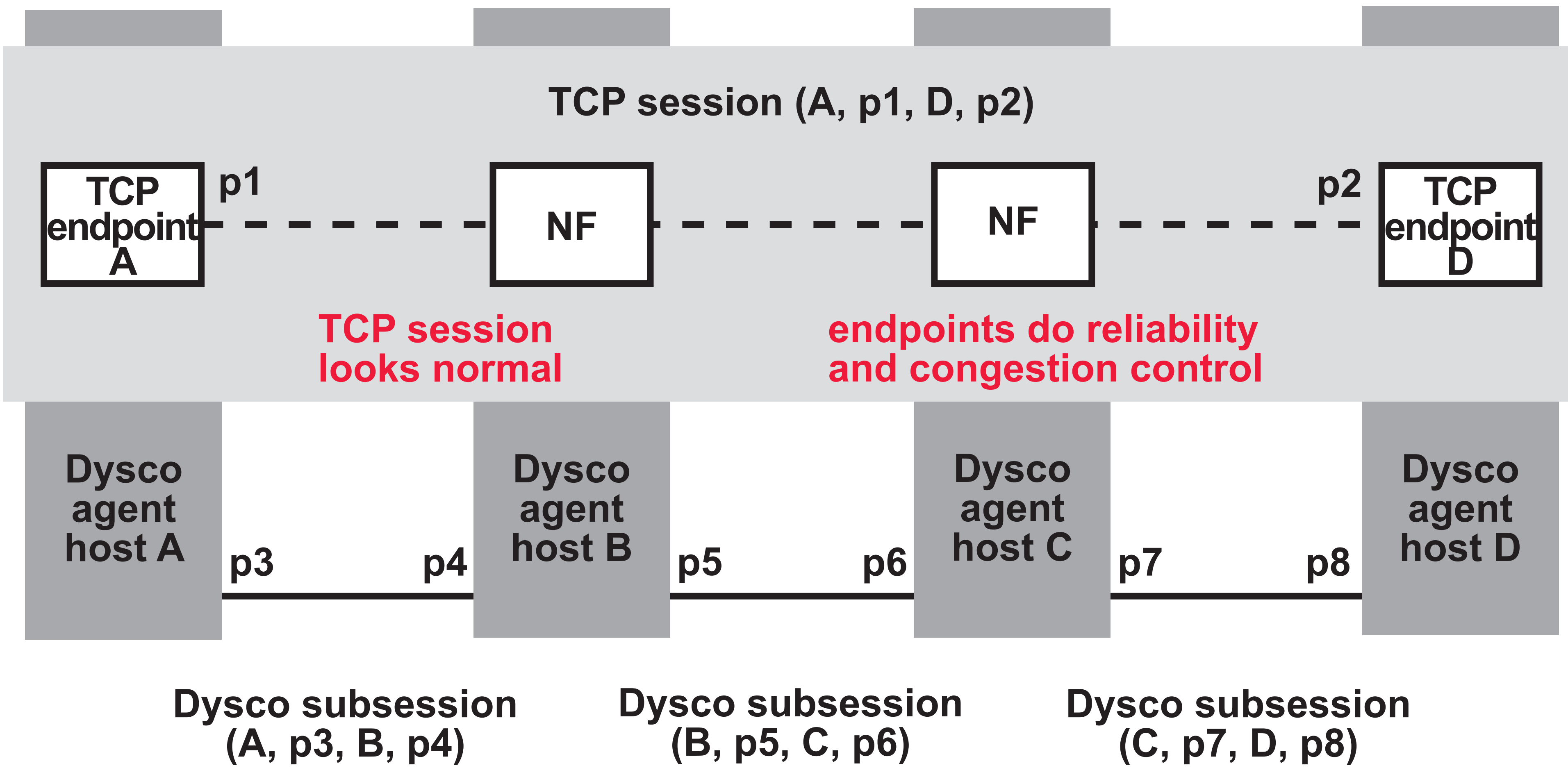
. . . a caching proxy if the content is non-cacheable

REPLACE . . . a middlebox that needs maintenance

. . . a middlebox that has become a hairpin after endpoint mobility



DYSCO IS A SESSION PROTOCOL FOR SERVICE CHAINING

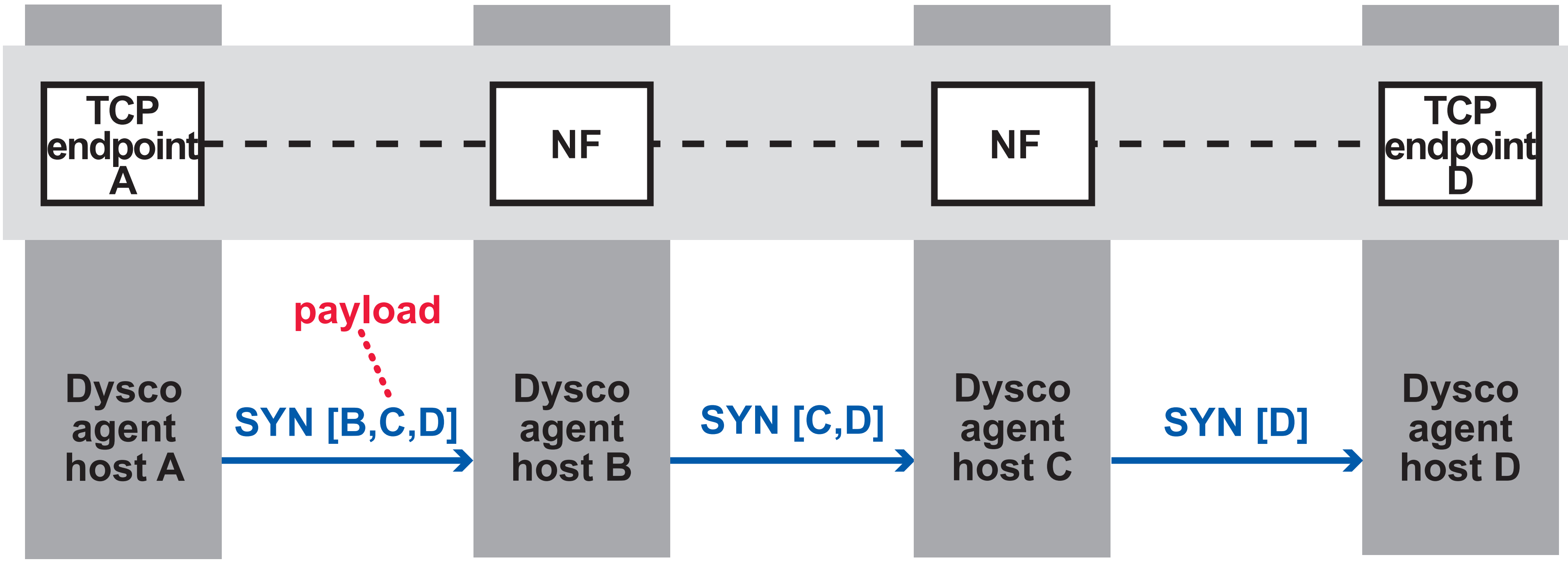


session and its subsessions are set up at the same time

instead of using encapsulation, Dysco agents rewrite packet headers so packet length stays the same

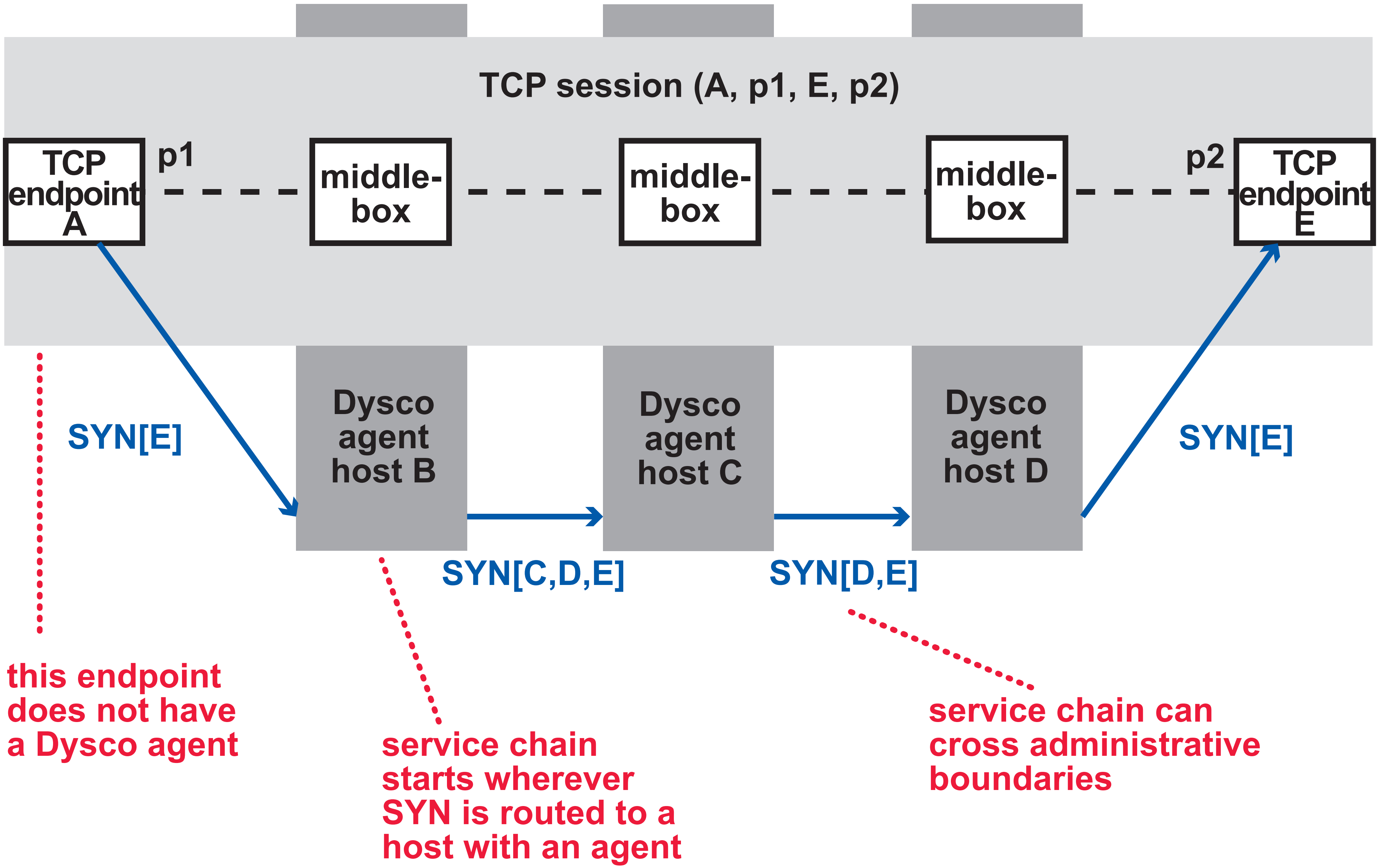
assuming that the NFs are stateful, there is very little extra state

AUTONOMOUS AGENTS, NO NEED FOR CONTROLLER



- any agent can cache policies (abstract or concrete service chains)
- session affinity comes for free
- most NFs run unmodified—Dysco is transparent to them
- local tags associate SYN packets going into and coming out of NFs that modify the TCP 5-tuple
- with an API, a NF can classify SYN packets and tell the Dysco agent where to send them next

INCREMENTAL OR SECURE DEPLOYMENT



DYNAMIC RECONFIGURATION

TRIGGERED BY MIDDLEBOX, AGENT, OR CONTROLLER

EXECUTED BY AGENT

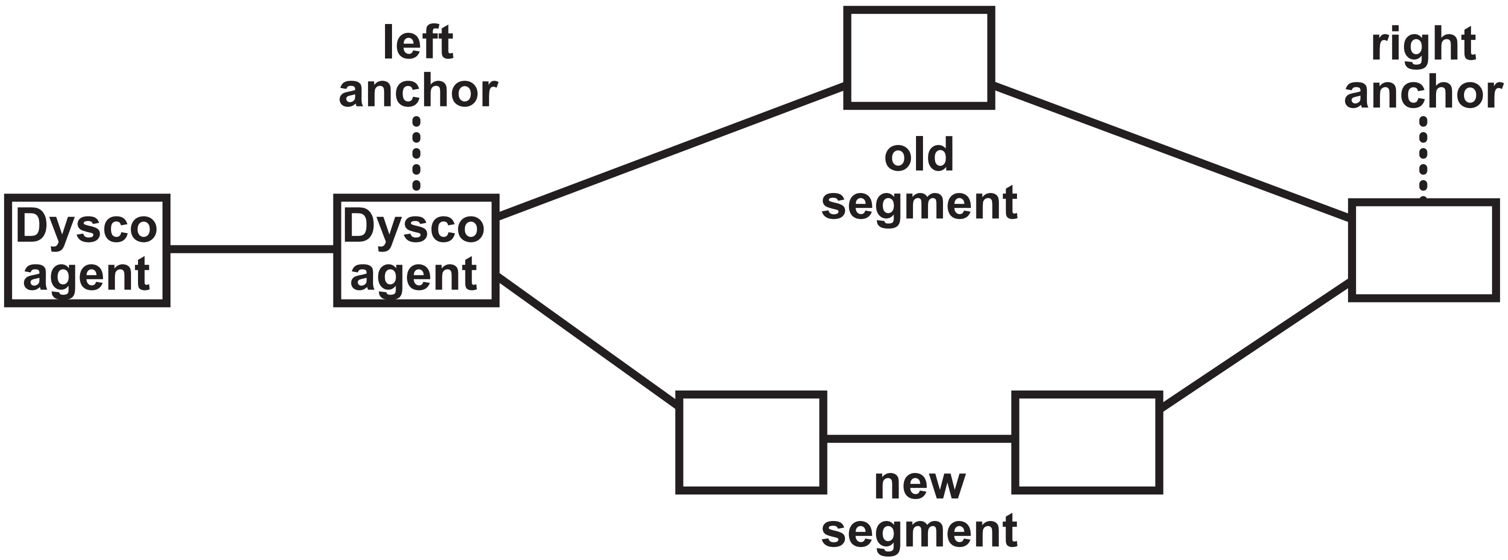
1 Left and right anchors lock the old segment.

control messages on old path

requestLock →

← *ackLock*

if there is contention to lock overlapping segments, protocol will resolve it



2 Left and right anchors set up new path.
If new path cannot be set up, abort reconfiguration.

control messages on new path

SYN →

← *SYN ACK*

ACK →

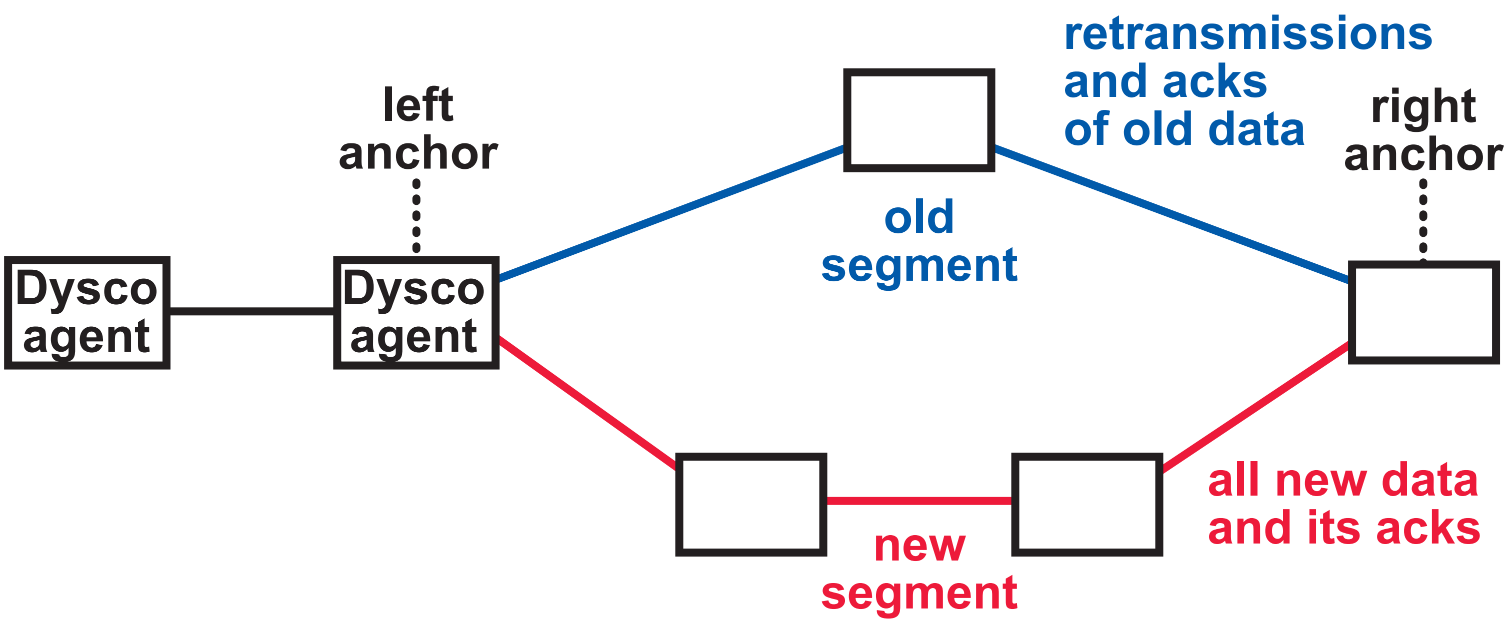
data is still being transmitted on the old path, so there are no delays

DYNAMIC RECONFIGURATION, CONTINUED

Anchors transmit all new data on new path.

3 On the old path, they send retransmissions of old data and acks of old data.

If NFs on old path altered sequence numbers, anchors compensate for this on the new path.



4 When all data on the old path has been acknowledged, the anchors go back to normal operation.

THE PROTOCOL CAN BE TRUSTED BECAUSE IT HAS BEEN VERIFIED WITH MODEL-CHECKING

DYSCO DEGRADES PERFORMANCE VERY LITTLE

SESSION INITIATION

- session initiation with 4 middleboxes
- worst case: checksum computation not offloaded to NIC
- average Dysco delay .094 ms

DYSCO agent is a Linux kernel module

could also use DPDK

TCP GOODPUT

- 1000 sessions going through the same middlebox (link is saturated)
- worst-case Dysco penalty is 1.5%

SERVER REQUESTS PER SECOND

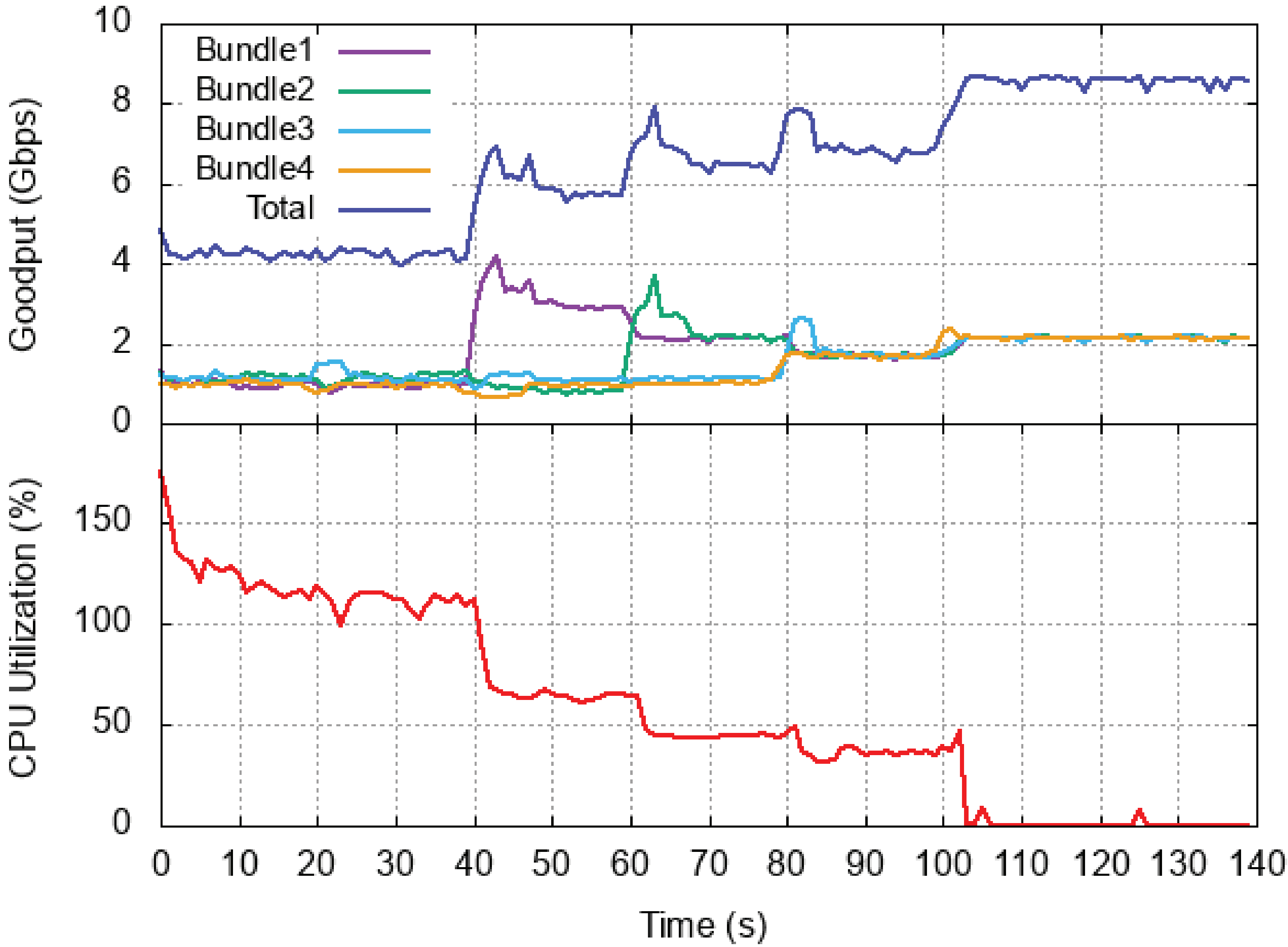
- we use NGINX HTTP server
- load is approximately 300,000 requests per second
- 4 middleboxes between the client and server
- worst-case Dysco penalty is 1.8%

RECONFIGURATION IMPROVES PERFORMANCE

600 TCP sessions, each going through a proxy

at intervals, we trigger removal of the proxy from 1/4 of the sessions

80% of reconfigura-tions take less than 2 ms, none more than 4 ms



after all removals, CPU utilization at the proxy drops to zero, GOODPUT DOUBLES