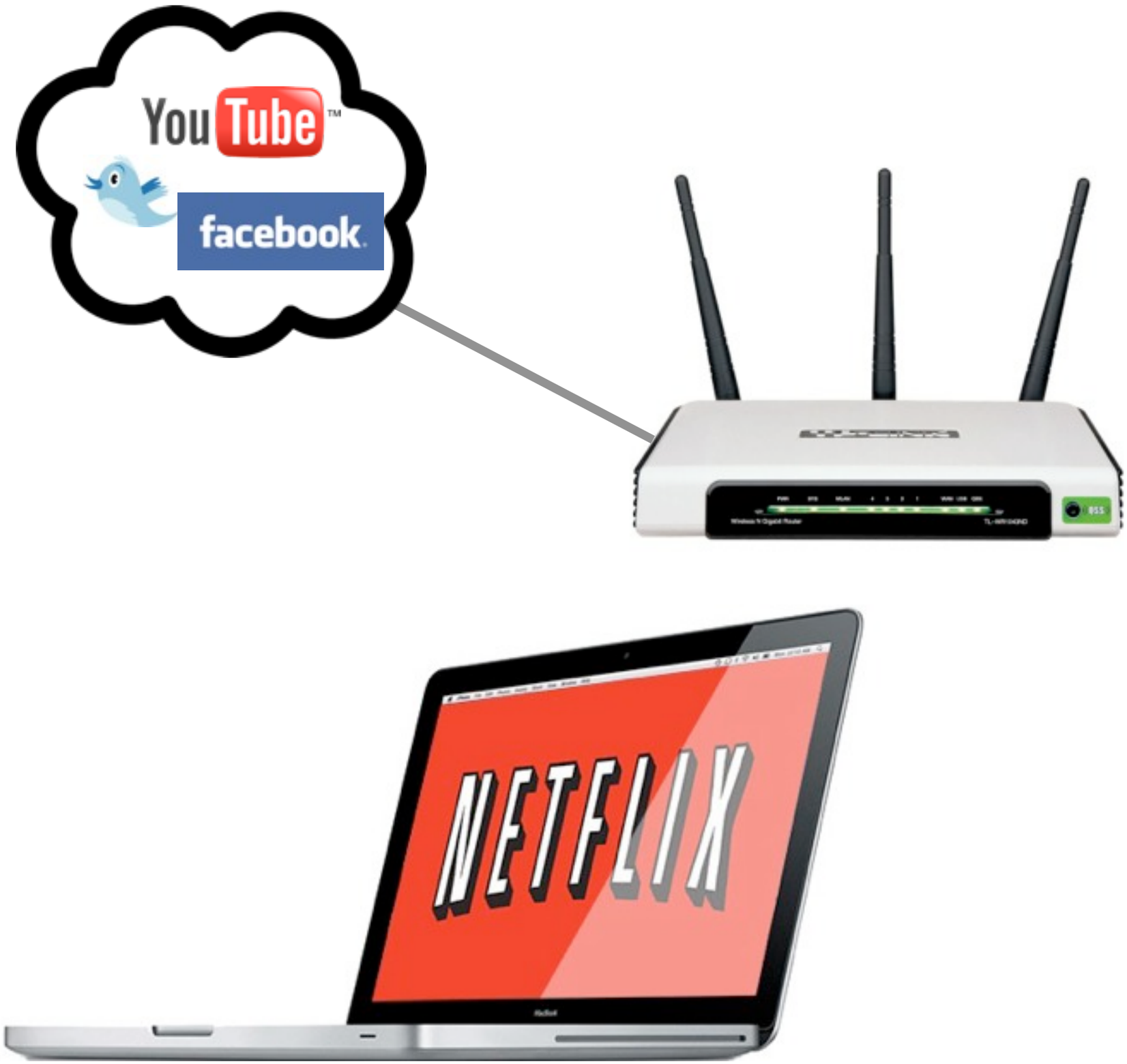
Hierarchical Policies for Software Defined Networks

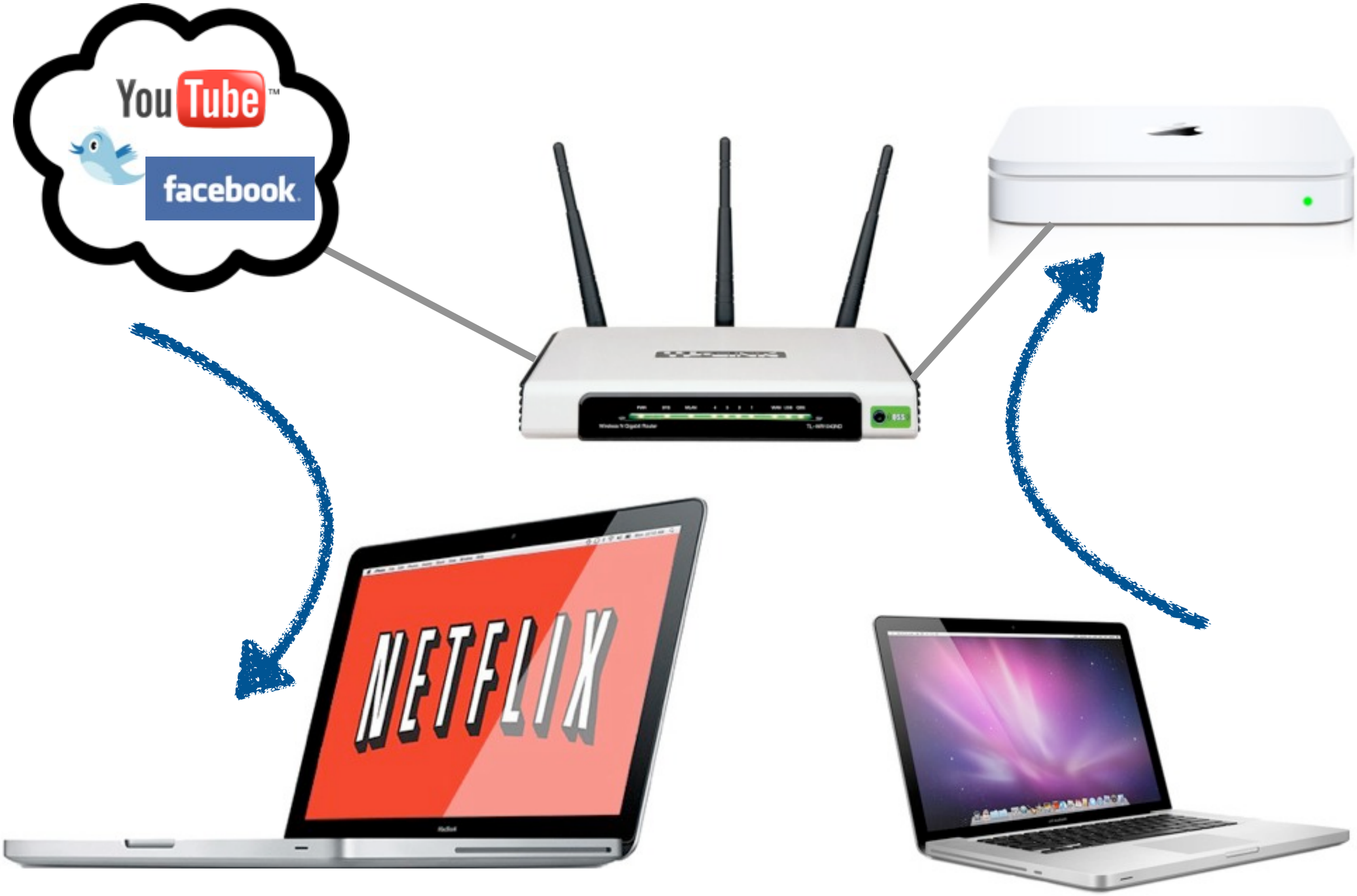
Andrew Ferguson, Arjun Guha, Chen Liang,
Rodrigo Fonseca, and Shriram Krishnamurthi

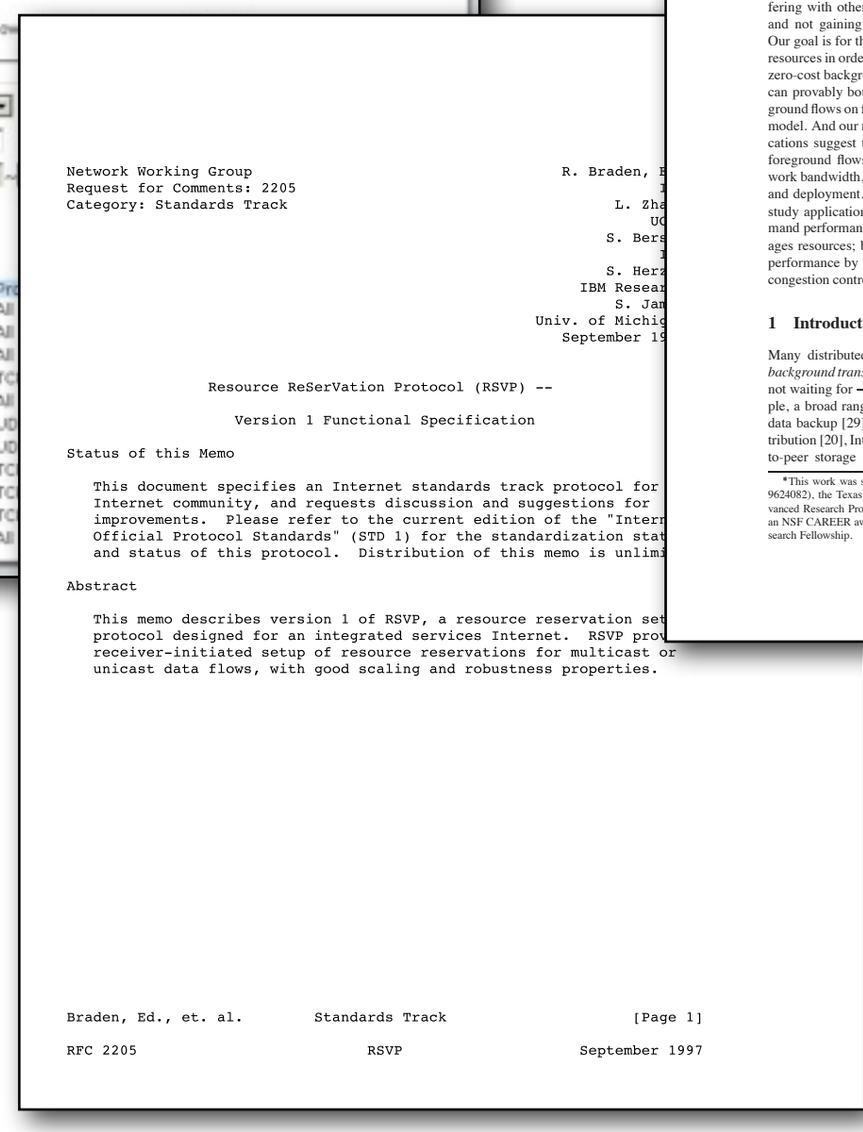
BROWN

Particpatory Networking


YouTube
facebook
NETFLIX


NETFLIX
89%
Buffering
Full Screen
More Episodes
Back to Browsing


YouTube
facebook
NETFLIX



```
Network Working Group                                    R. Braden,
Request for Comments: 2205                                       L
                                                            L. Zha
                                                                UC
                                                            S. Bers
                                                                 I
                                                            S. Herz
                                                       IBM Resear
                                                            S. Jam
                                                   Univ. of Michig
                                                      September 19

                 Resource ReSerVation Protocol (RSVP) --

                     Version 1 Functional Specification

Status of this Memo

   This document specifies an Internet standards track protocol for
   Internet community, and requests discussion and suggestions for
   improvements.  Please refer to the current edition of the "Intern
   Official Protocol Standards" (STD 1) for the standardization stat
   and status of this protocol.  Distribution of this memo is unlimi

Abstract

   This memo describes version 1 of RSVP, a resource reservation set
   protocol designed for an integrated services Internet.  RSVP prov
   receiver-initiated setup of resource reservations for multicast or
   unicast data flows, with good scaling and robustness properties.

Braden, Ed., et. al.        Standards Track                    [Page 1]

RFC 2205                          RSVP                    September 1997
```

TCP Nice: A Mechanism for Background Transfers

Arun Venkataramani Ravi Kokku Mike Dahlin *

Laboratory of Advanced Systems Research
Department of Computer Sciences
University of Texas at Austin, Austin, TX 78712
{arun, rkoku, dahlin}@cs.utexas.edu

Abstract

Many distributed applications can make use of large *background transfers* — transfers of data that humans are not waiting for — to improve availability, reliability, latency or consistency. However, given the rapid fluctuations of available network bandwidth and changing resource costs due to technology trends, hand tuning the aggressiveness of background transfers risks (1) complicating applications, (2) being too aggressive and interfering with other applications, and (3) being too timid and not gaining the benefits of background transfers. Our goal is for the operating system to manage network resources in order to provide a simple abstraction of near zero-cost background transfers. Our system, TCP Nice, can provably bound the interference inflicted by background flows on foreground flows in a restricted network model. And our microbenchmarks and case study applications suggest that in practice it interferes little with foreground flows, reaps a large fraction of spare network bandwidth, and simplifies application construction and deployment. For example, in our prefetching case study application, aggressive prefetching improves demand performance by a factor of three when Nice manages resources; but the same prefetching hurts demand performance by a factor of six under standard network congestion control.

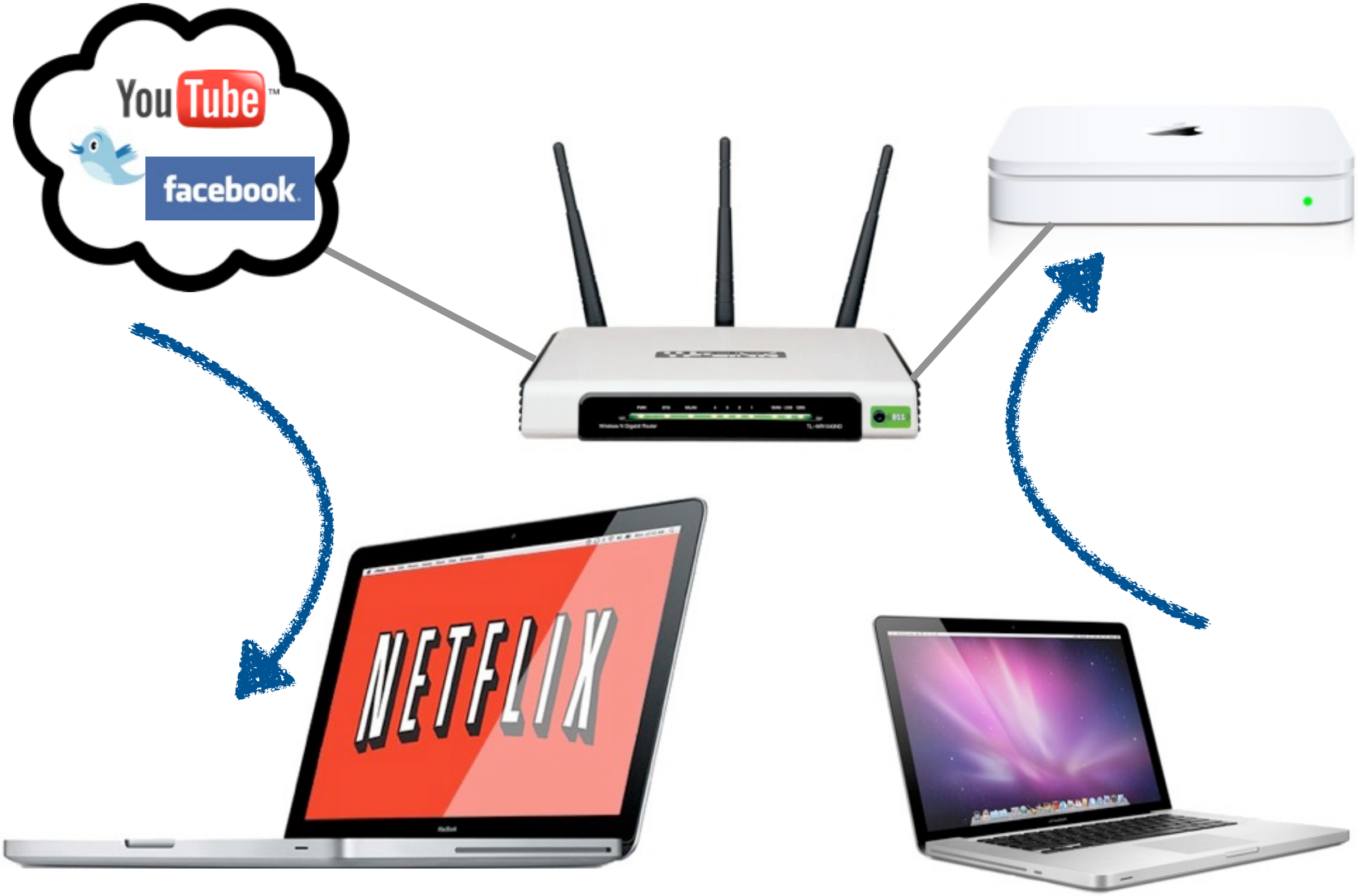
1 Introduction

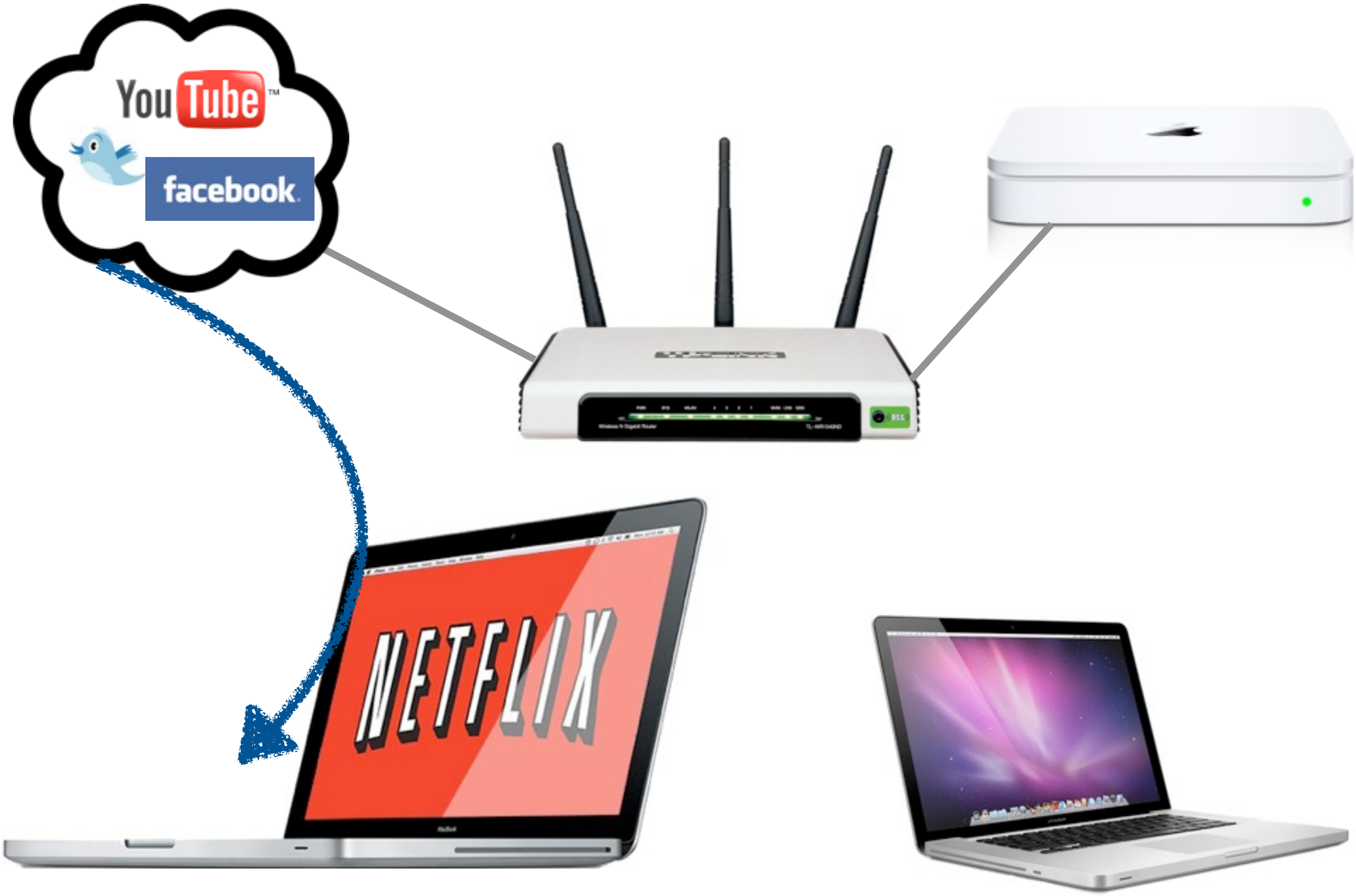
Many distributed applications can make use of large *background transfers* — transfers of data that humans are not waiting for — to improve service quality. For example, a broad range of applications and services such as data backup [29], prefetching [50], enterprise data distribution [20], Internet content distribution [2], and peer-to-peer storage [16, 43] can trade increased network bandwidth consumption and possibly disk space for improved service latency [15, 18, 26, 32, 38, 50], improved availability [11, 53], increased scalability [2], stronger consistency [53], or support for mobility [28, 41, 47]. Many of these services have potentially unlimited bandwidth demands where incrementally more bandwidth consumption provides incrementally better service. For example, a web prefetching system can improve its hit rate by fetching objects from a virtually unlimited collection of objects that have non-zero probability of access [8, 10] or by updating cached copies more frequently as data change [13, 50, 48]; Technology trends suggest that "wasting" bandwidth and storage to improve latency and availability will become increasingly attractive in the future: per-byte network transport costs and disk storage costs are low and have been improving at 80-100% per year [9, 17, 37]; conversely network availability [11, 40, 54] and network latencies improve slowly, and long latencies and failures waste human time.

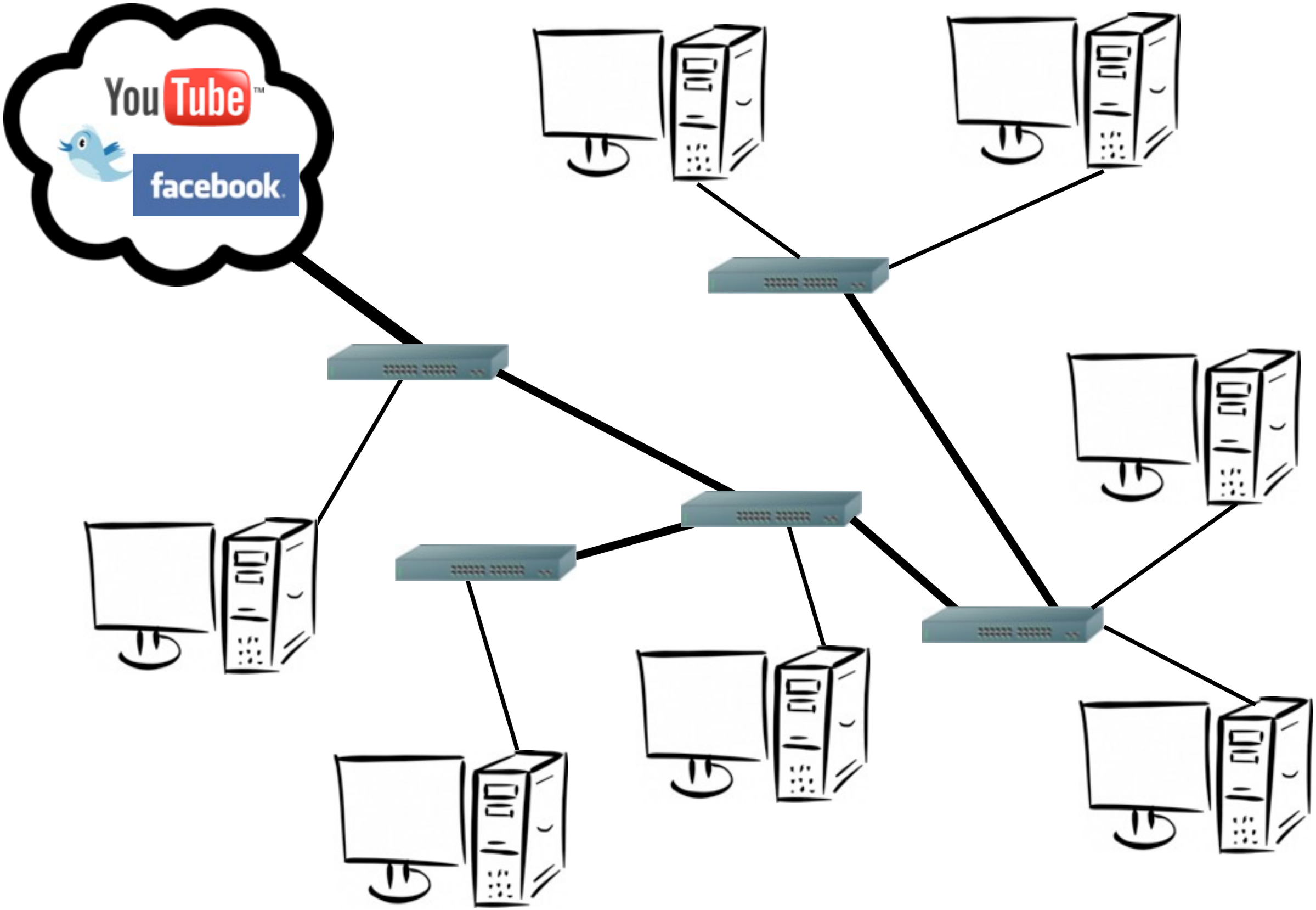
Current operating systems and networks do not provide good support for aggressive background transfers. In particular, because background transfers compete with foreground requests, they can hurt overall performance and availability by increasing network congestion. Applications must therefore carefully balance the benefits of background transfers against the risk of both *self-interference*, where applications hurt their own performance, and *cross-interference*, where applications hurt other applications' performance. Often, applications attempt to achieve this balance by setting "magic numbers" (e.g., the prefetch threshold in prefetching algorithms [18, 26]) that have little obvious relationship to system goals (e.g., availability or latency) or constraints (e.g., current spare network bandwidth).

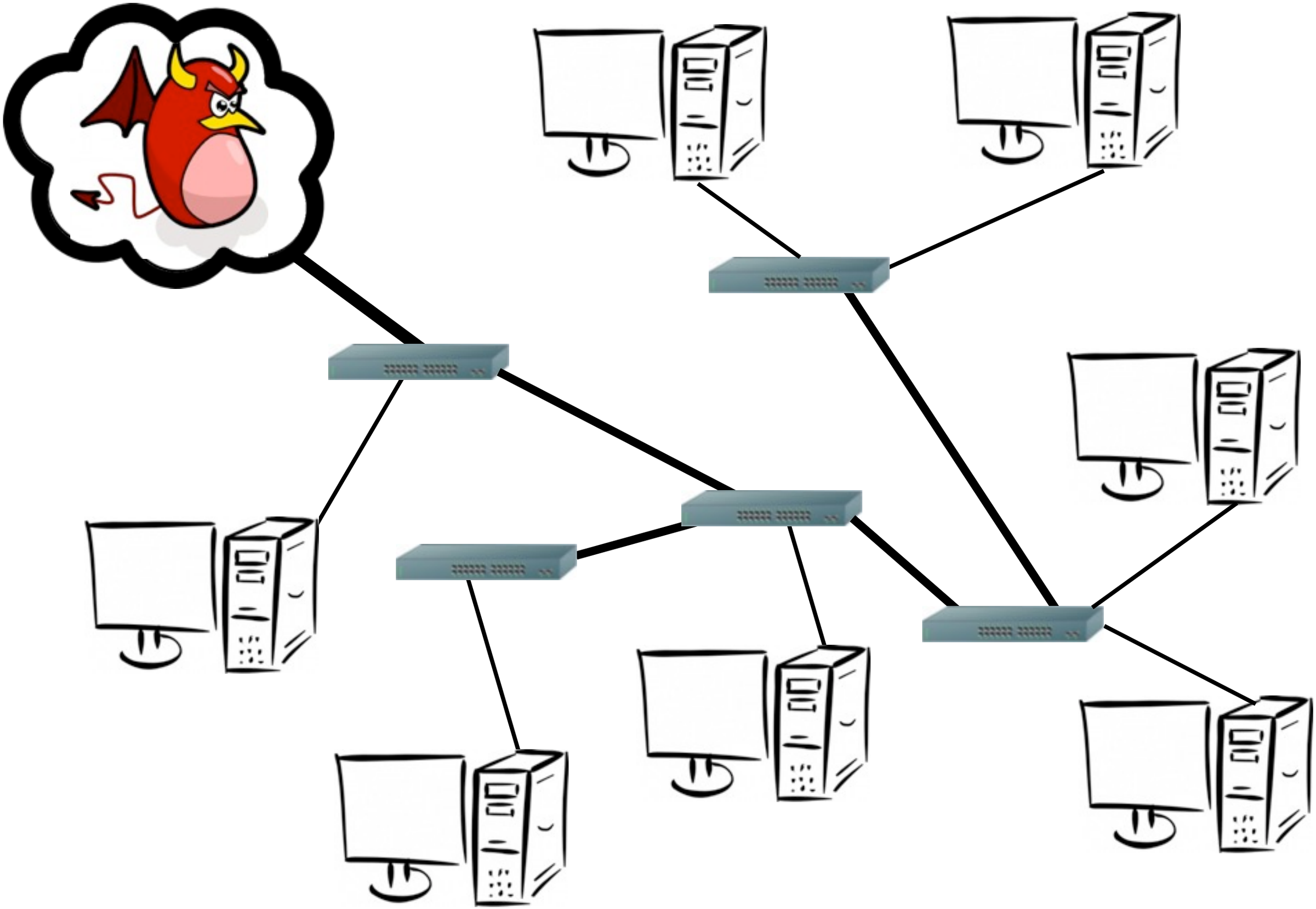
Our goal is for the operating system to manage network resources in order to provide a simple abstraction of zero-cost background transfers. A self-tuning background transport layer will enable new classes of applications by (1) simplifying applications, (2) reducing the risk of being too aggressive, and (3) making

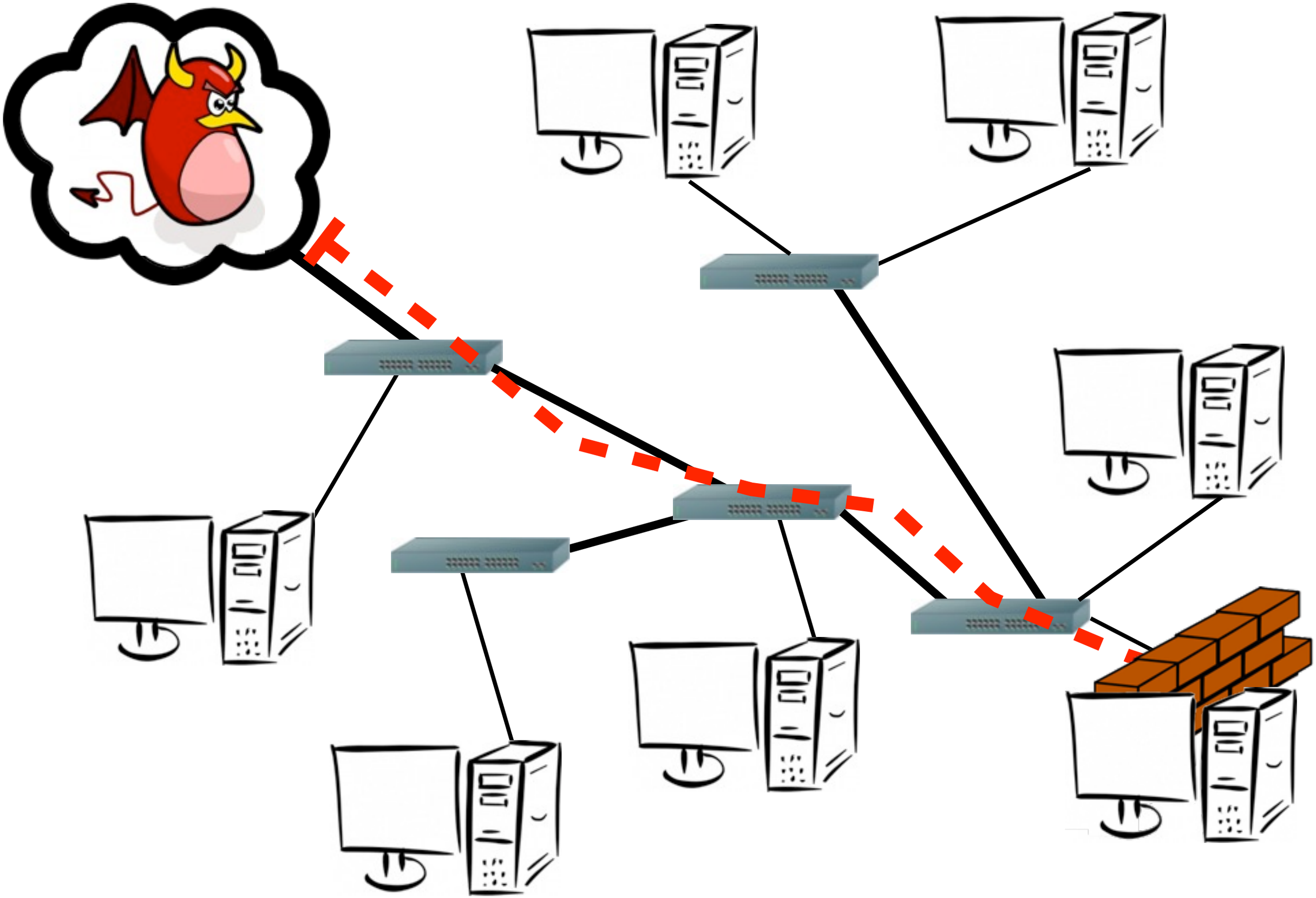
*This work was supported in part by an NSF CISE grant (CDA-9624082), the Texas Advanced Technology Program, the Texas Advanced Research Program, and Tivoli. Dahlin was also supported by an NSF CAREER award (CCR-9733842) and an Alfred P. Sloan Research Fellowship.

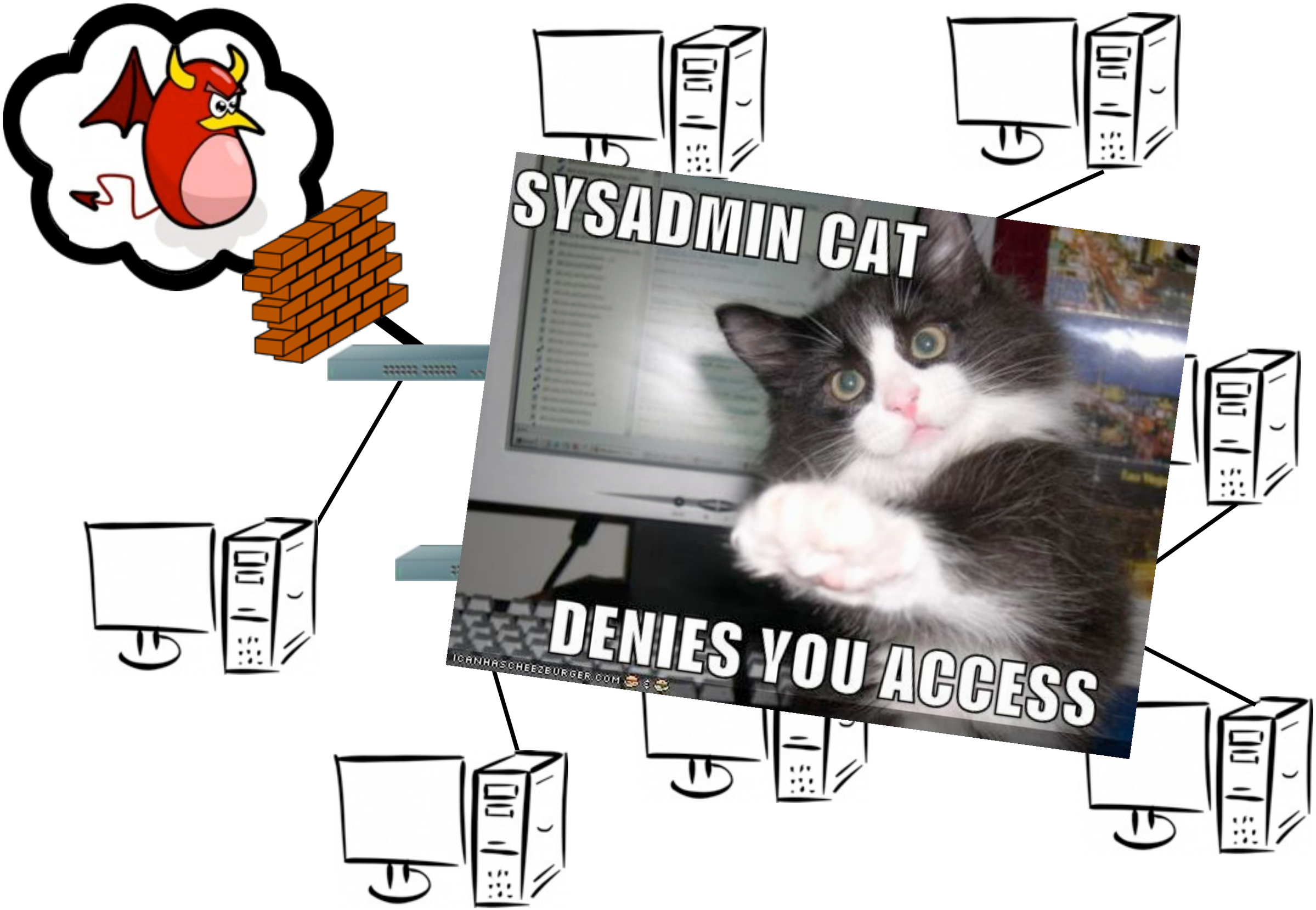

YouTube
facebook
NETFLIX

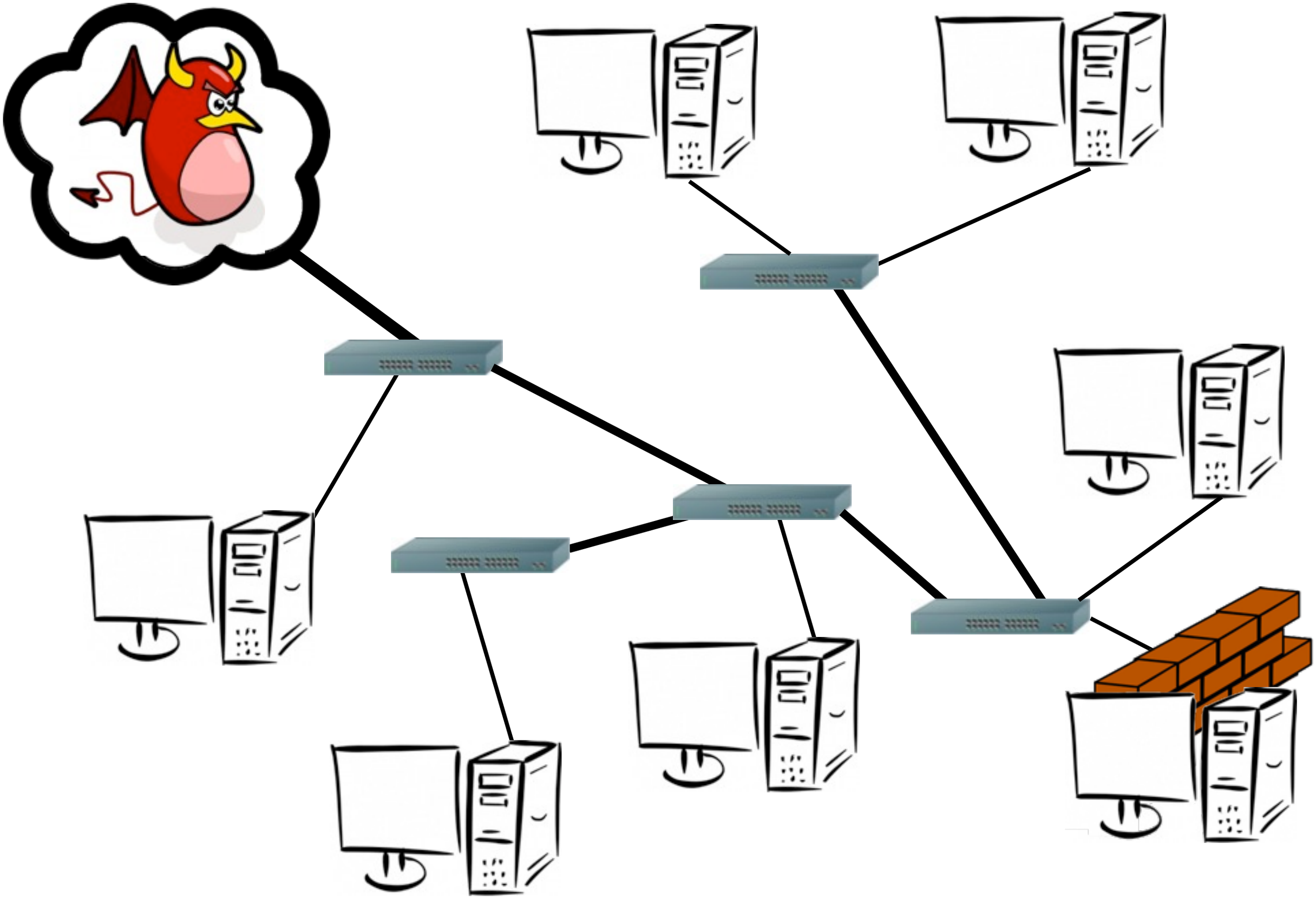

YouTube
facebook
NETFLIX

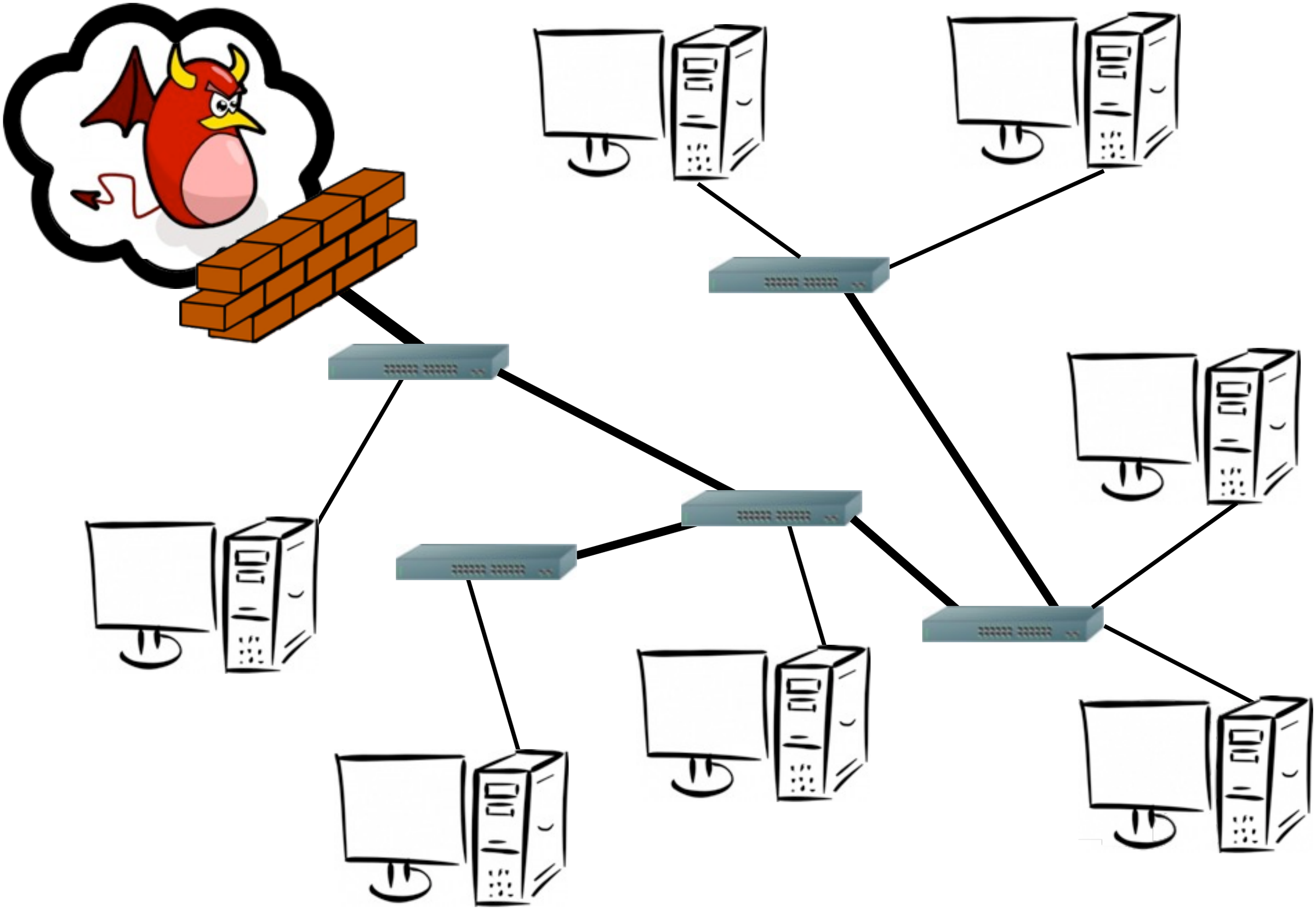

YouTube
facebook

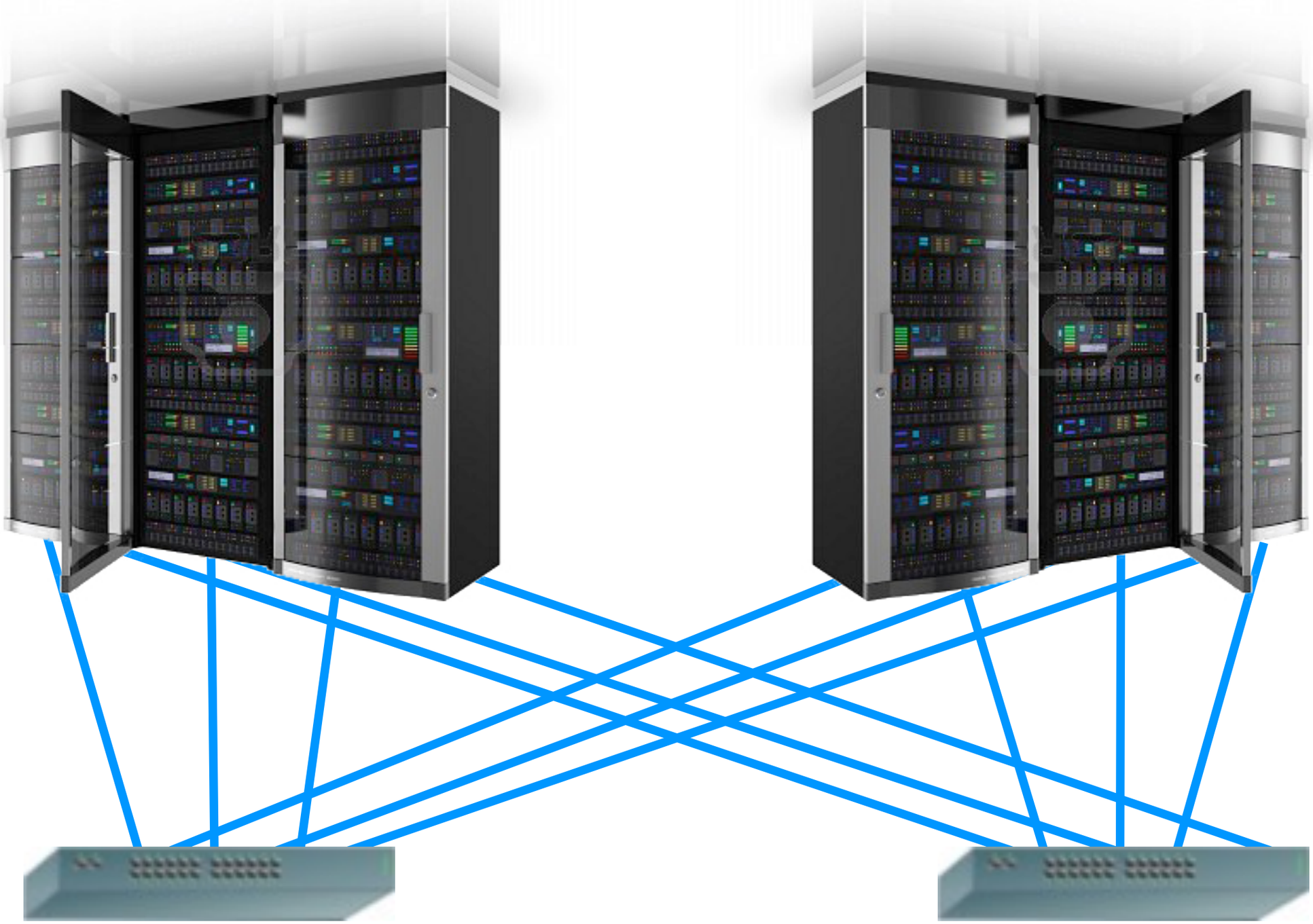


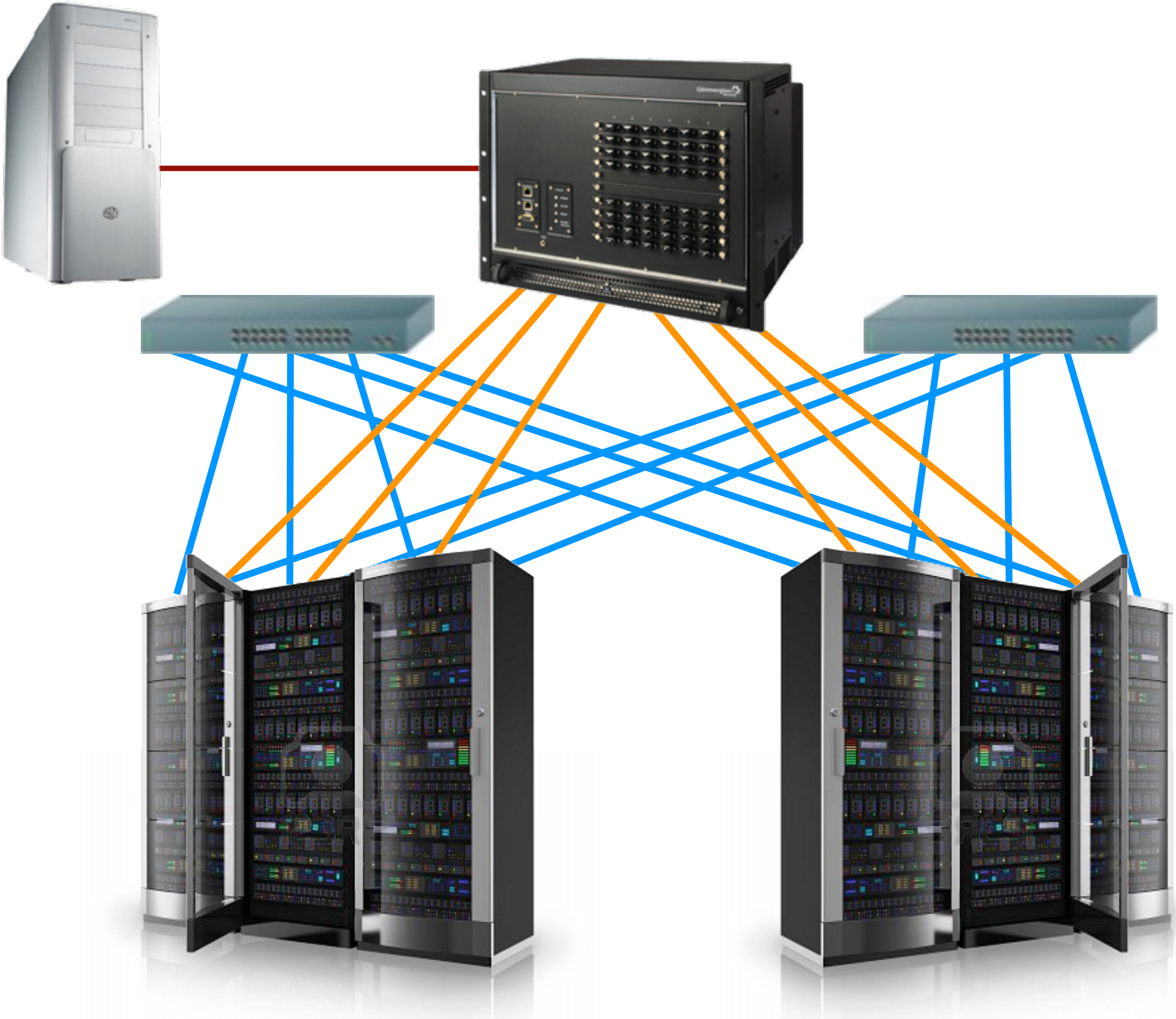


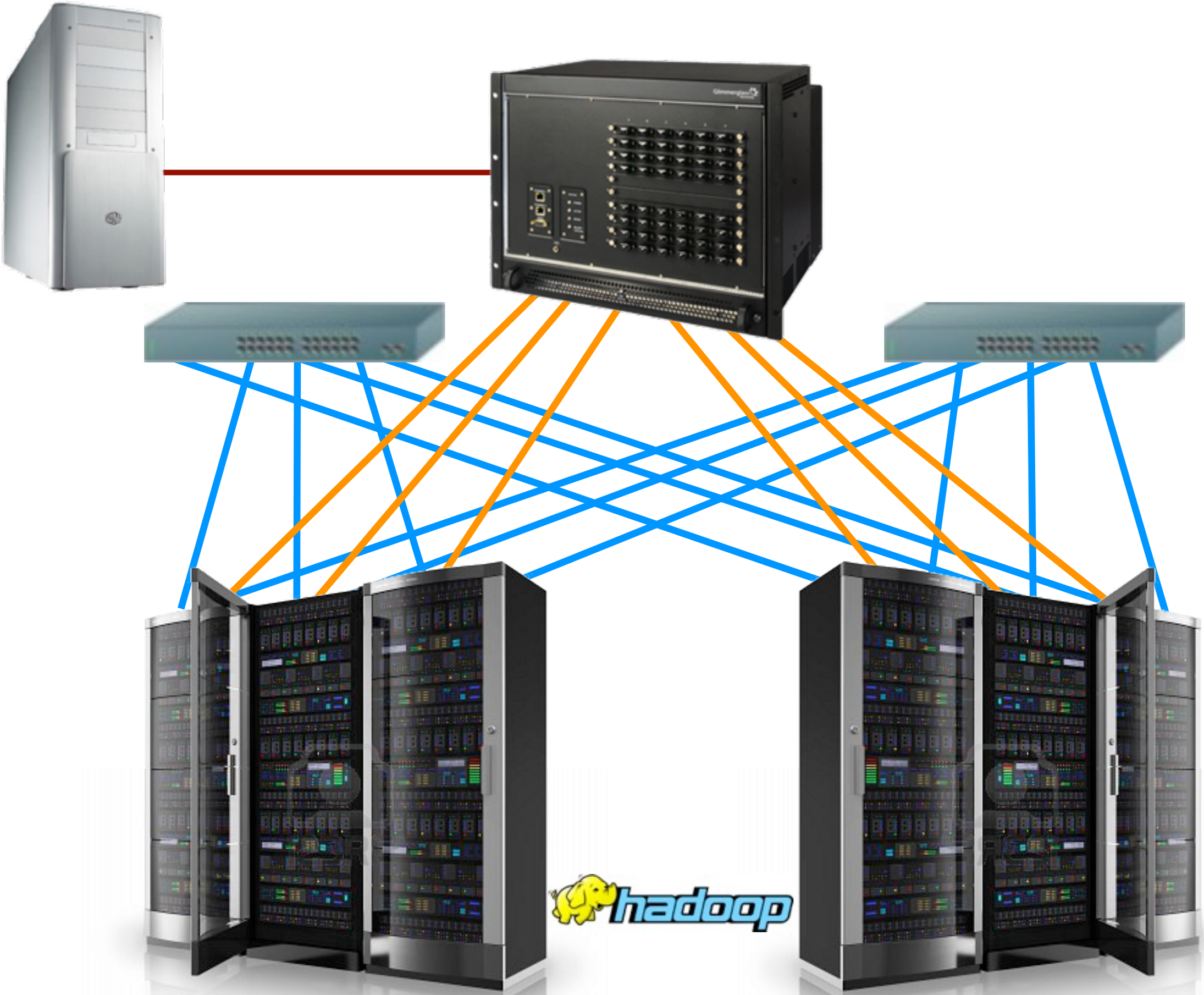

SYSADMIN CAT
DENIES YOU ACCESS
ICANHASCHEEZBURGER.COM

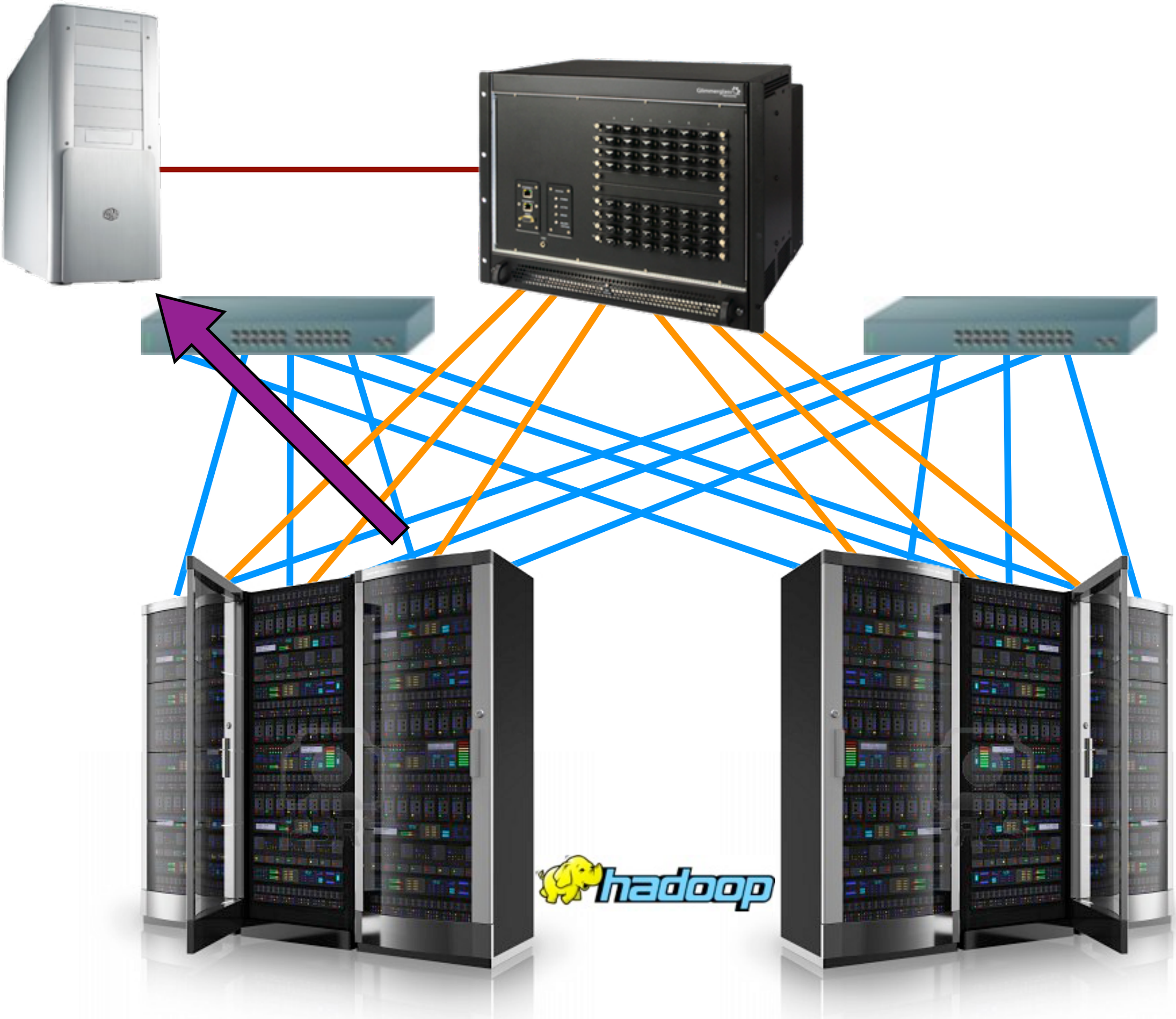


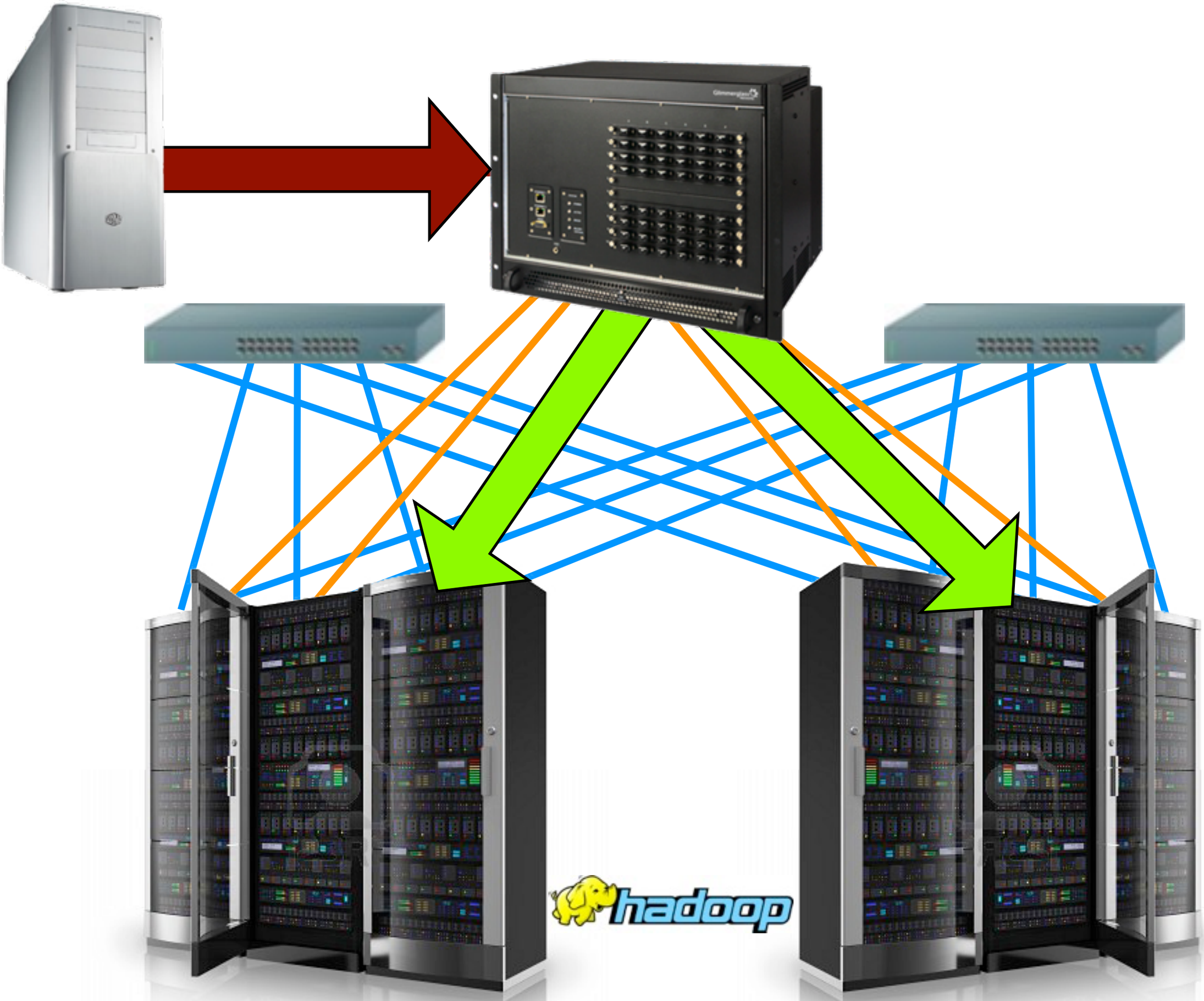




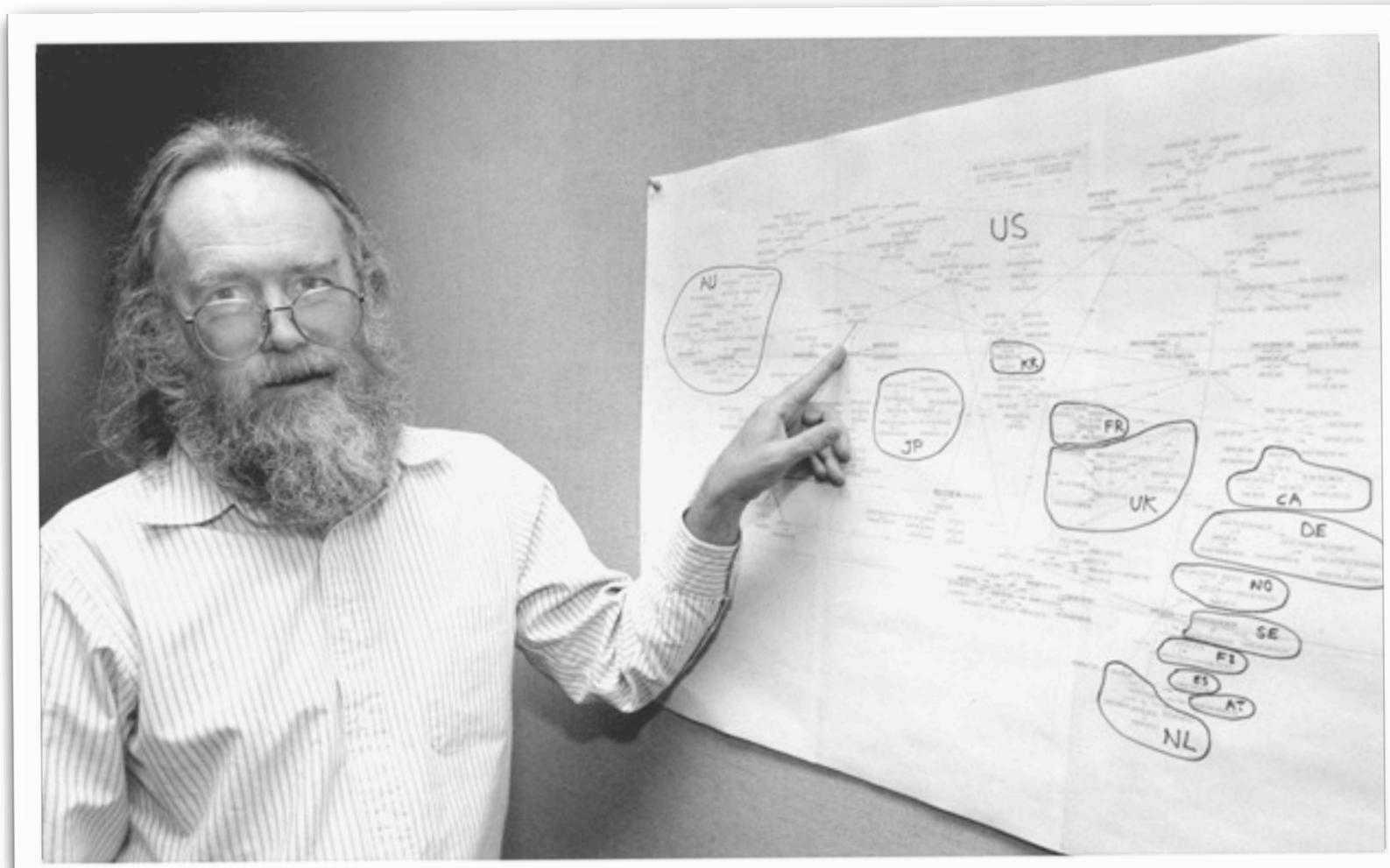



hadoop


hadoop


hadoop

Participatory Networking


US
AU
KR
JP
FR
UK
CA
DE
NO
SE
FI
ES
AT
NL


OCCUPY
EVERYTHING
#OCCUPYWALLST
WE ALREADY KNOW THAT WE OWN EVERYTHING—THE TASK IS TO EXCLUDE THE INTRUSIONS OF CAPITAL AND POWER
Beware
BANKS
BAILED

Participatory Networking

Safe?

Secure?

Fair?

Loop freedom?

Black holes?

Participatory Networking

1. semantics + protocol (Hot-ICE ’12)

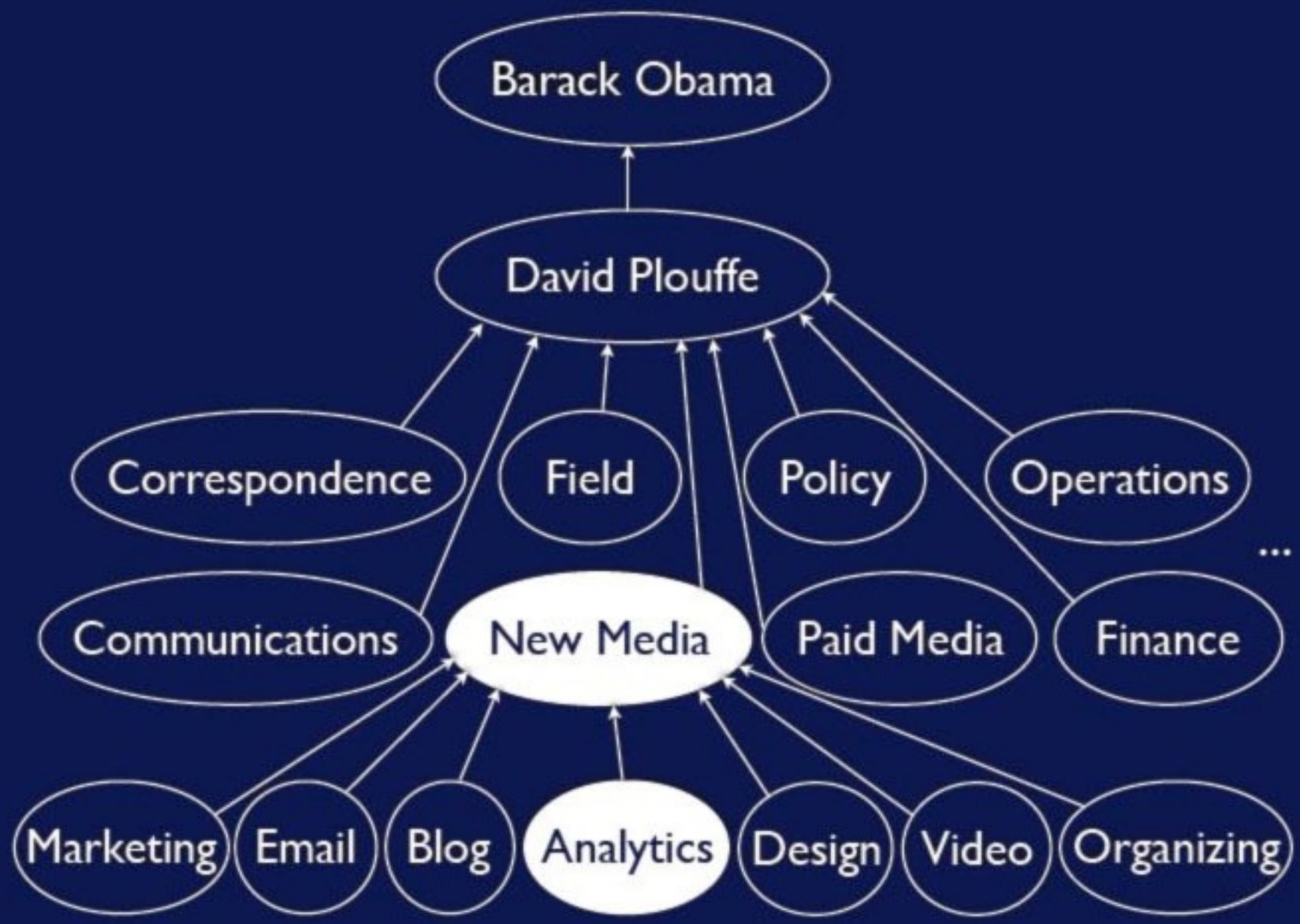
Participatory Networking

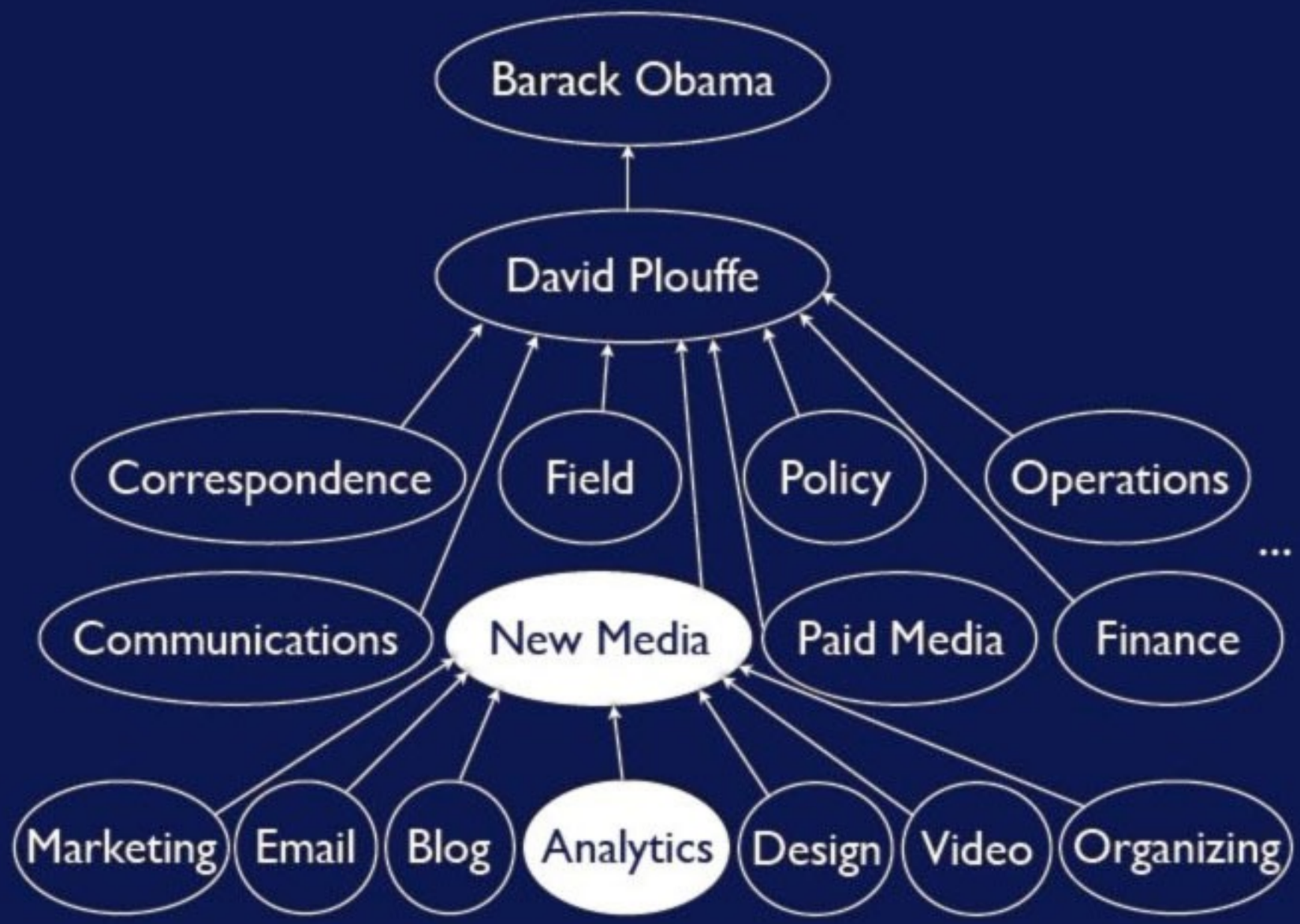
1. semantics + protocol (Hot-ICE ’12)
2. implementation (this talk)

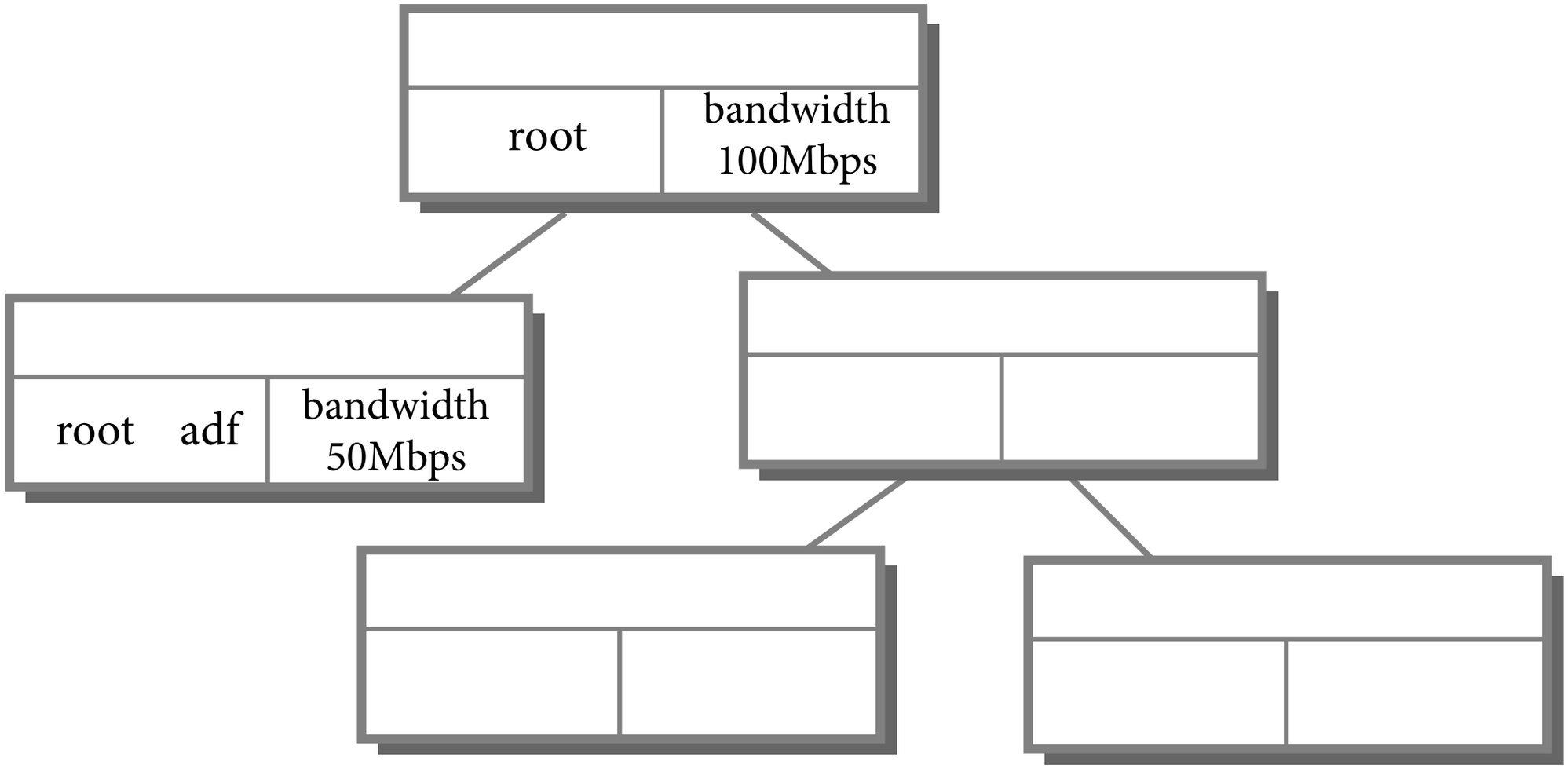
Participatory Networking

1. semantics + protocol (Hot-ICE '12)
2. implementation (this talk) PANE

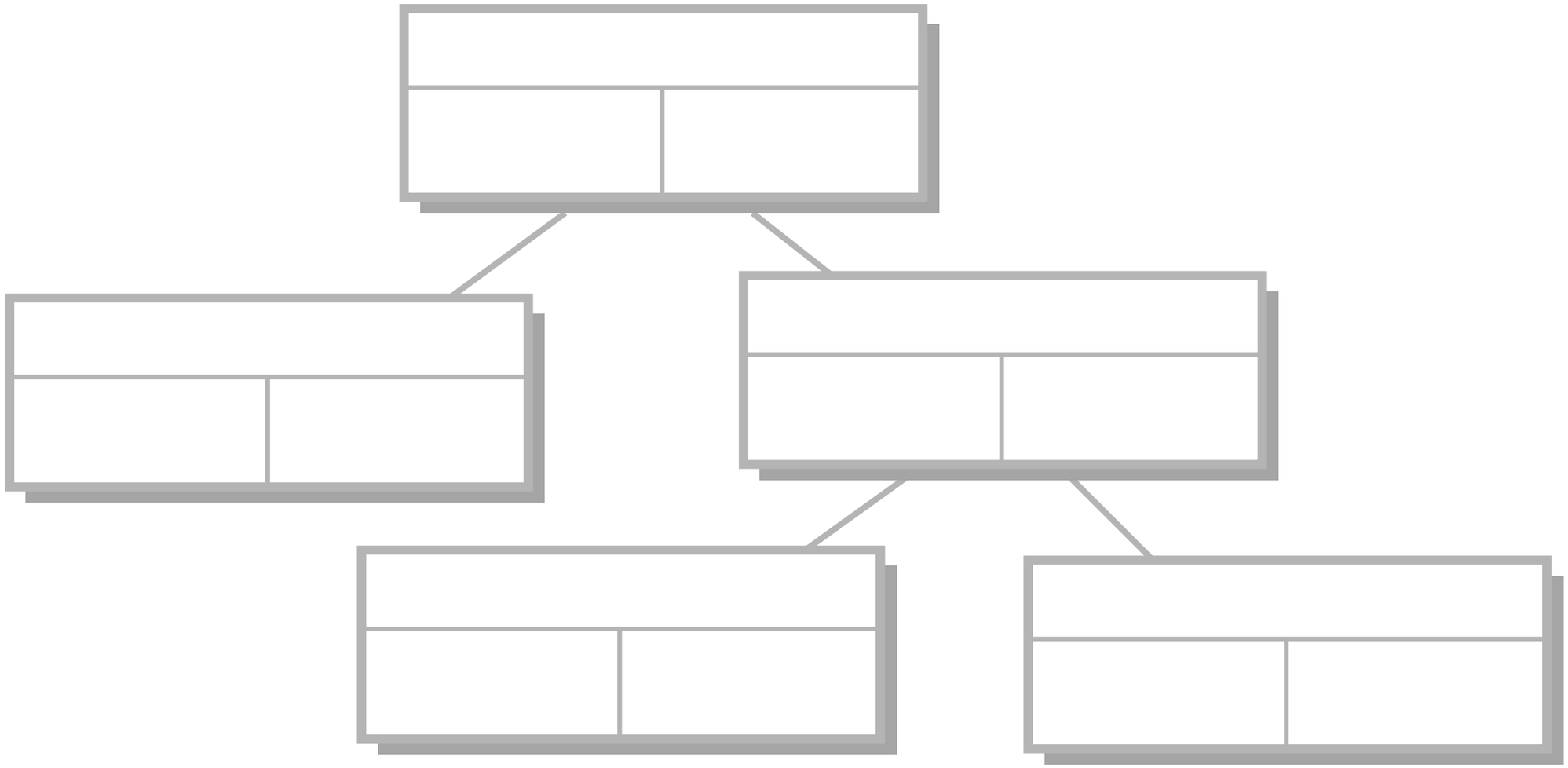
Hierarchical Flow Tables


Barack Obama
David Plouffe
Correspondence
Field
Policy
Operations
...
Communications
New Media
Paid Media
Finance
Marketing
Email
Blog
Analytics
Design
Video
Organizing

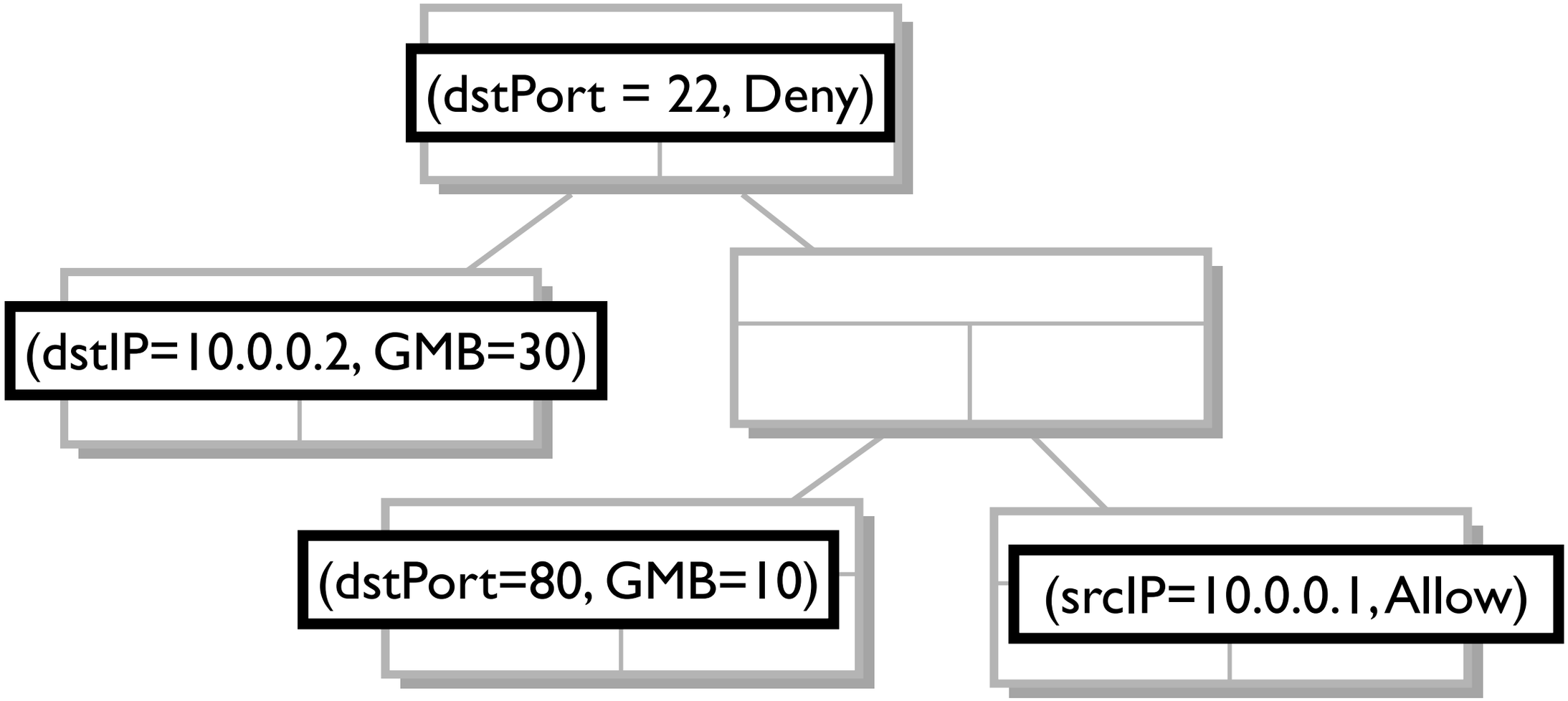

Barack Obama
David Plouffe
Correspondence
Field
Policy
Operations
...
Communications
New Media
Paid Media
Finance
Marketing
Email
Blog
Analytics
Design
Video
Organizing



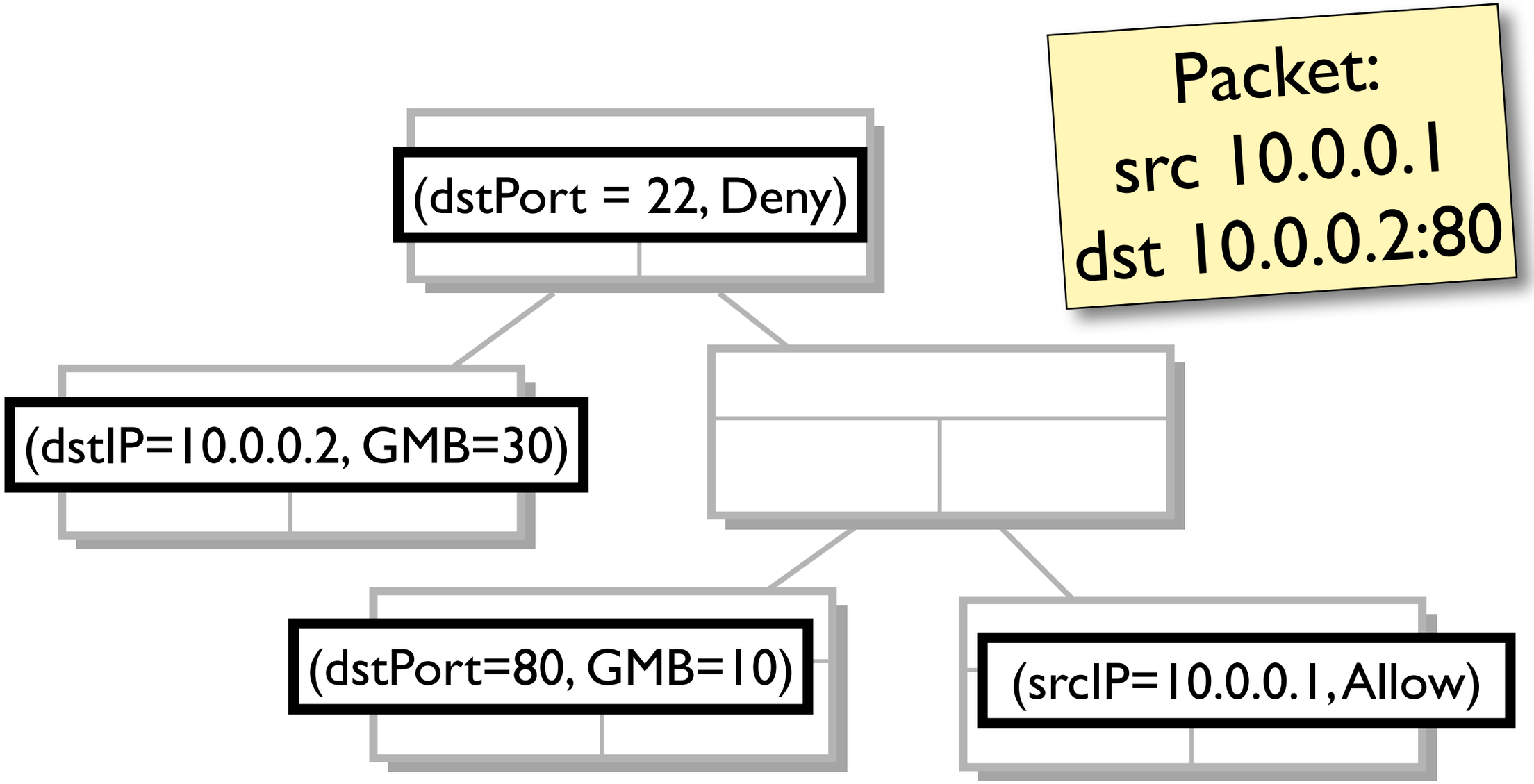
Hierarchy of Privileges



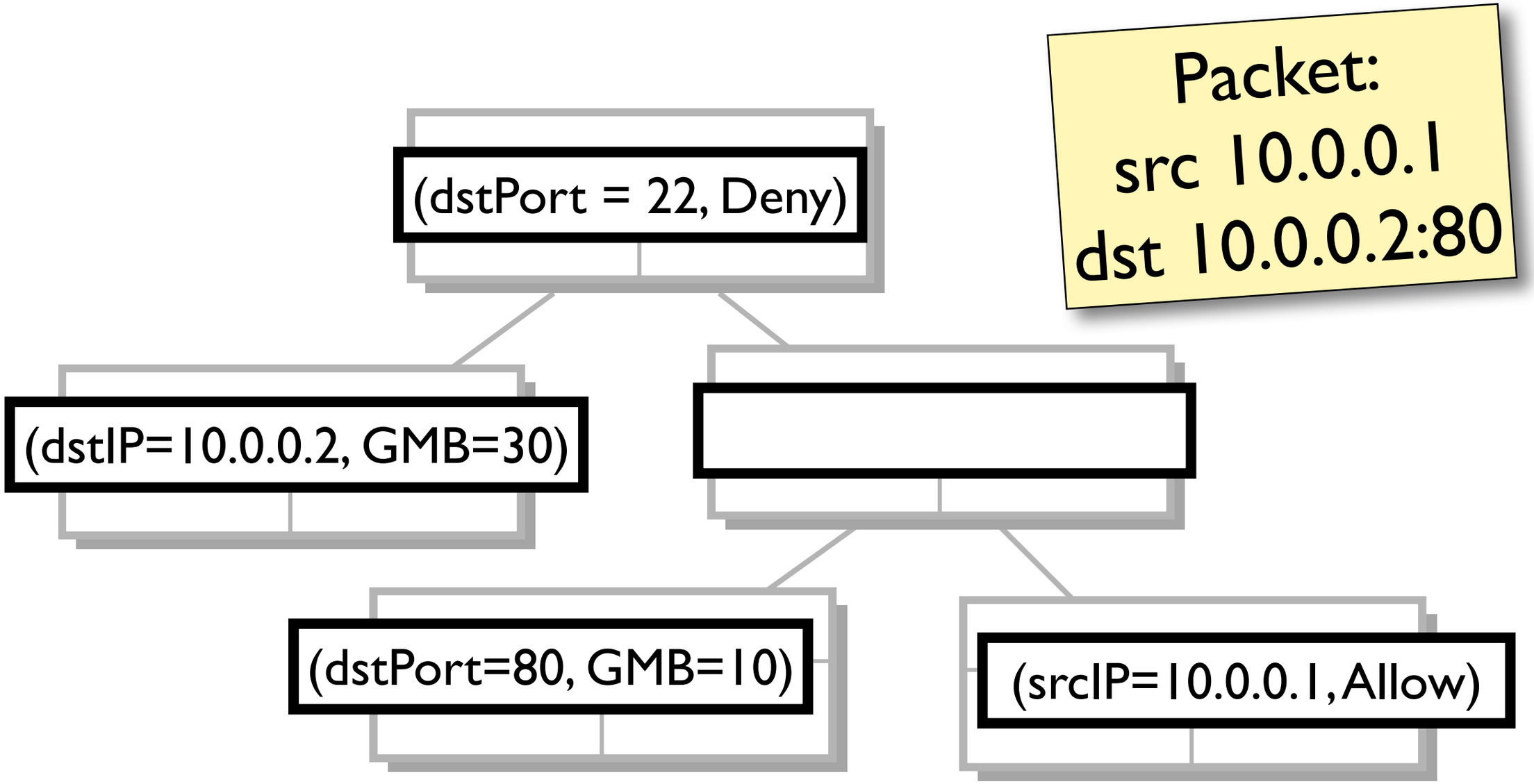
Hierarchy of Policies



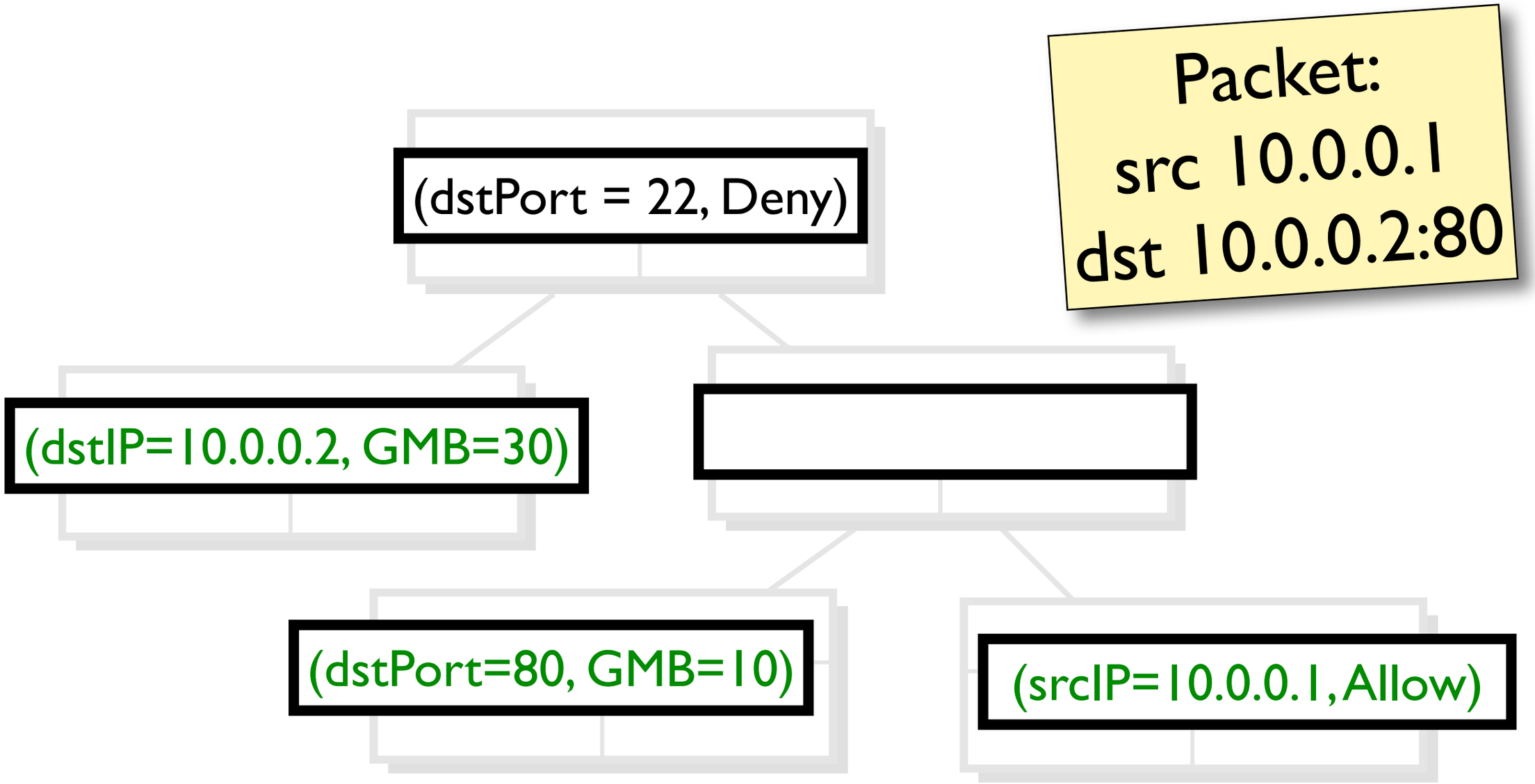
Hierarchy of Policies



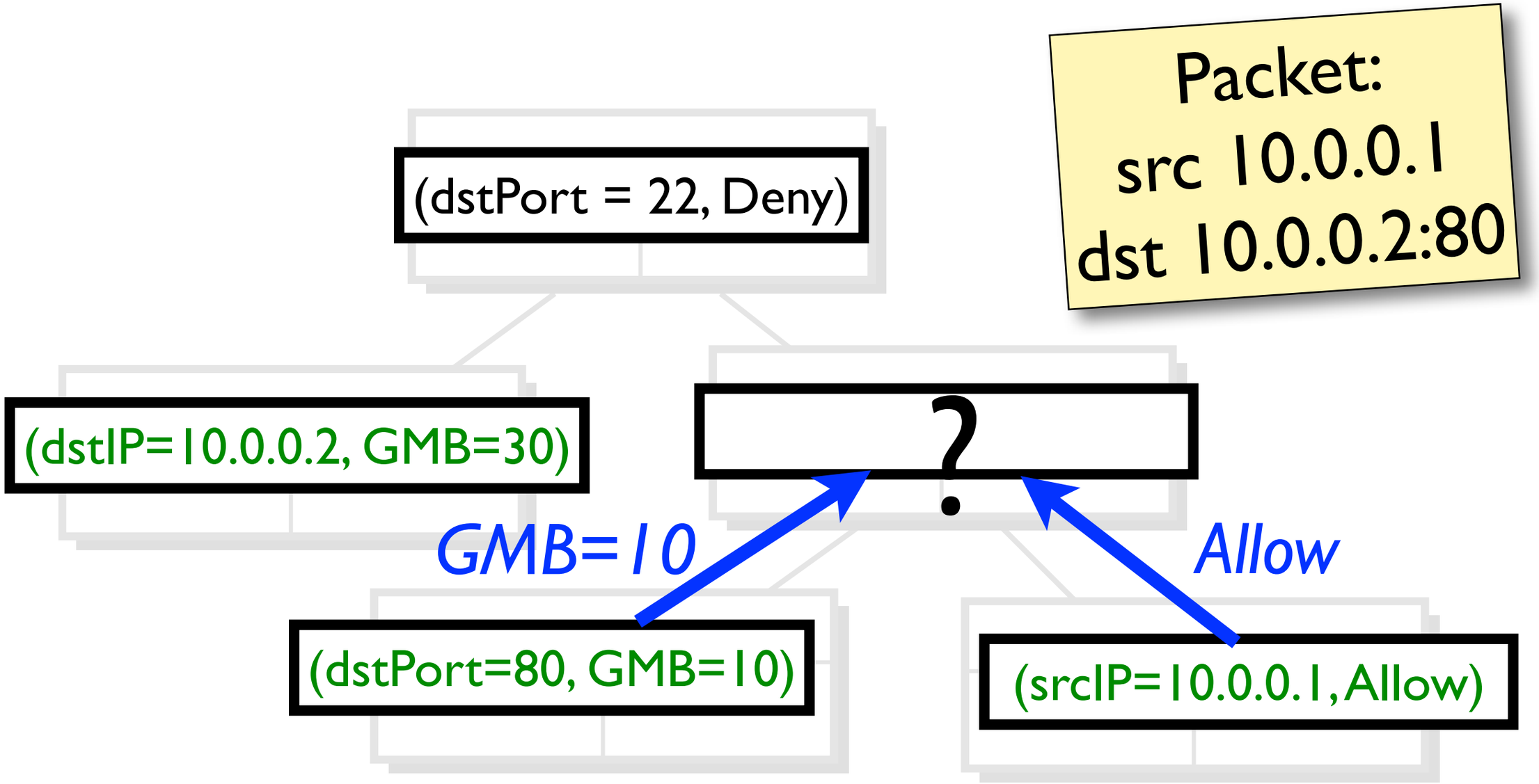
Hierarchy of Policies



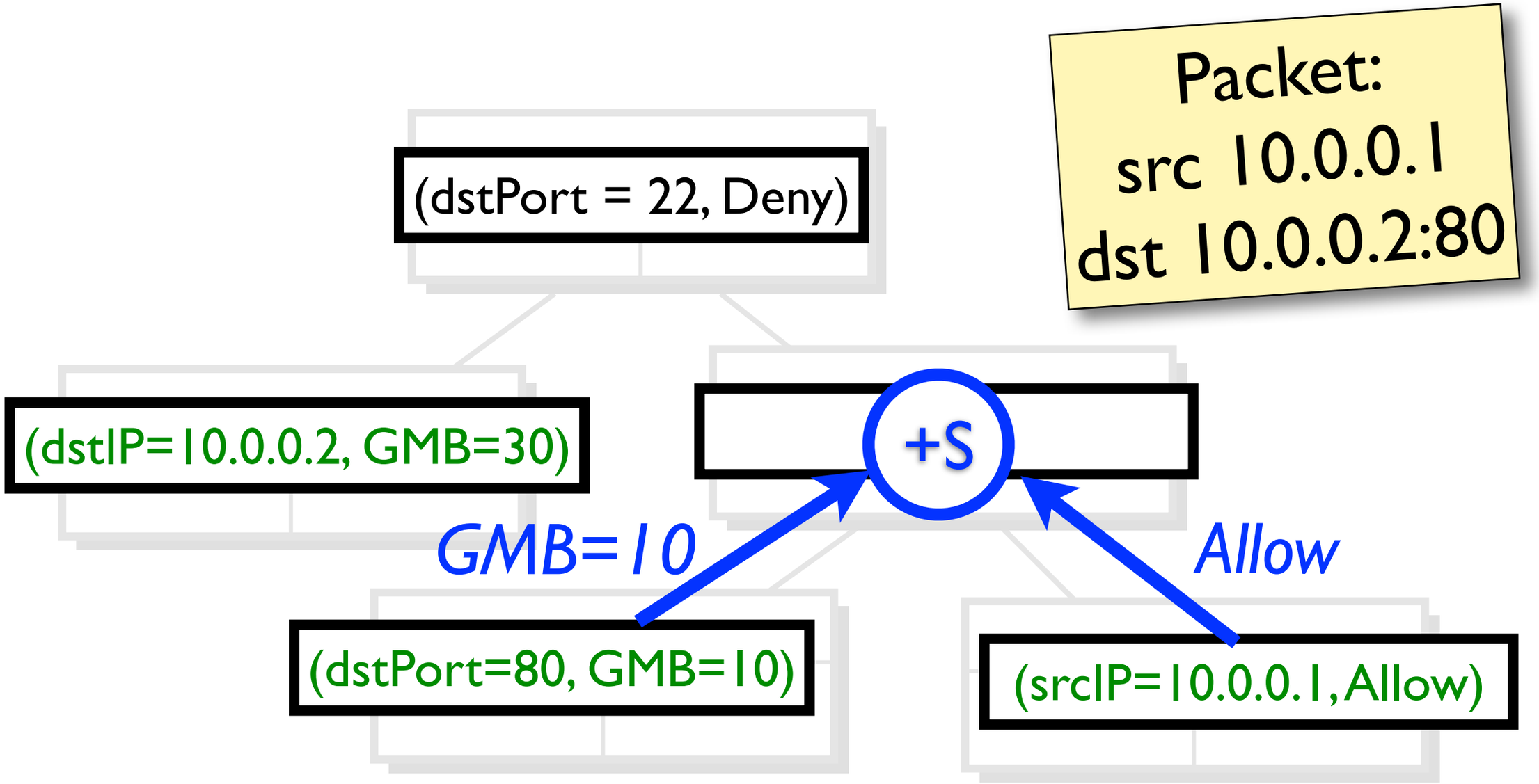
Hierarchical Flow Table



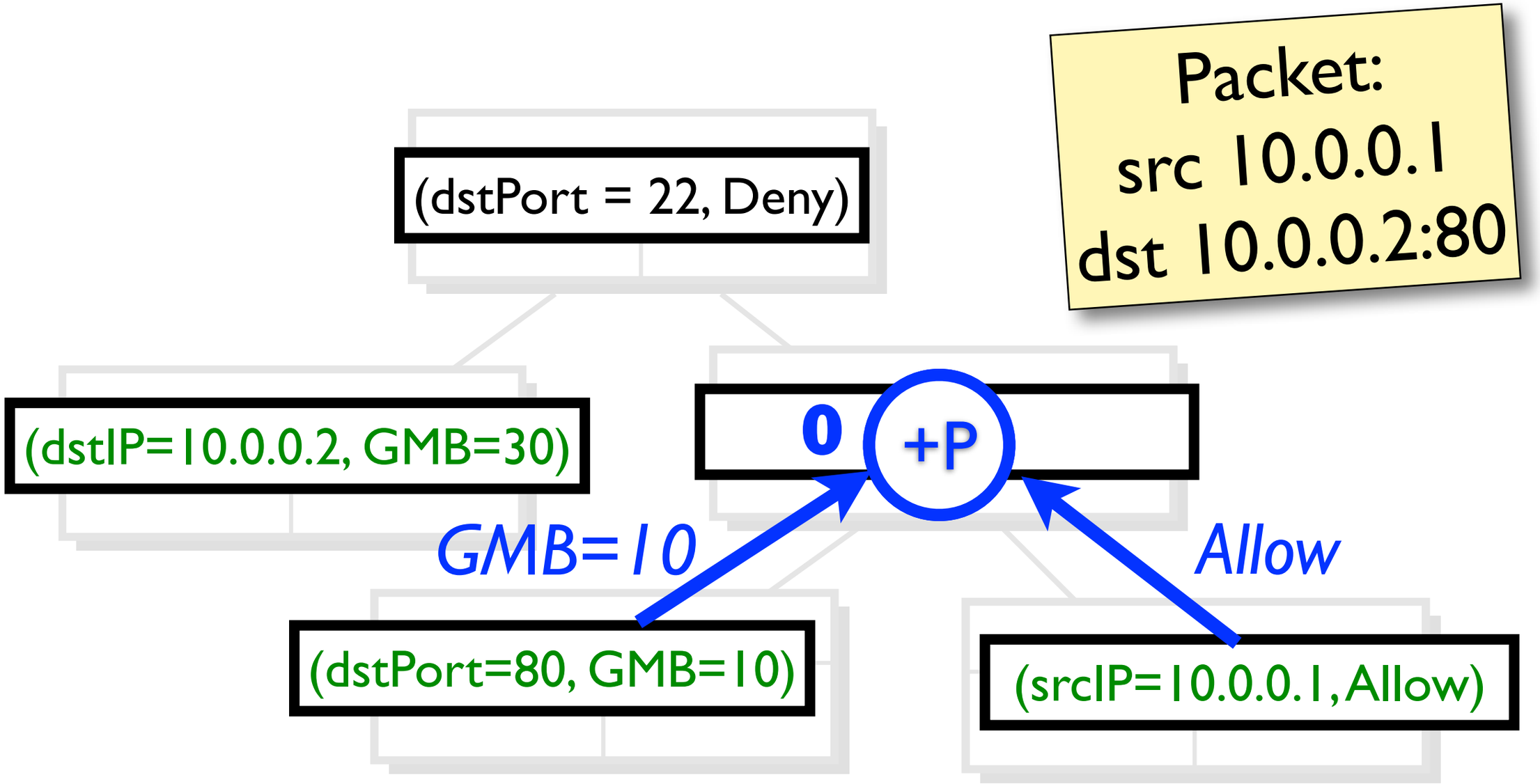
Hierarchical Flow Table



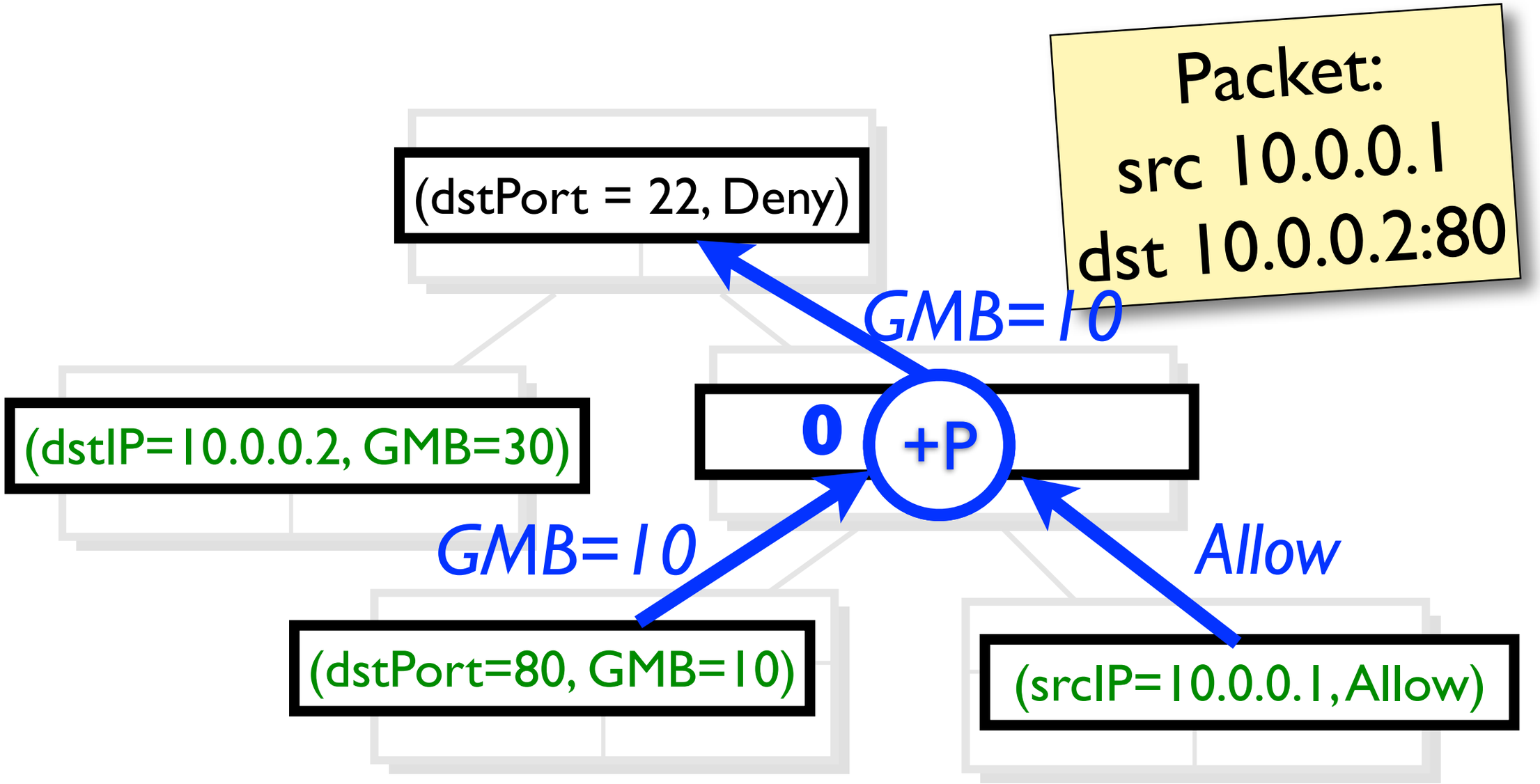
Hierarchical Flow Table



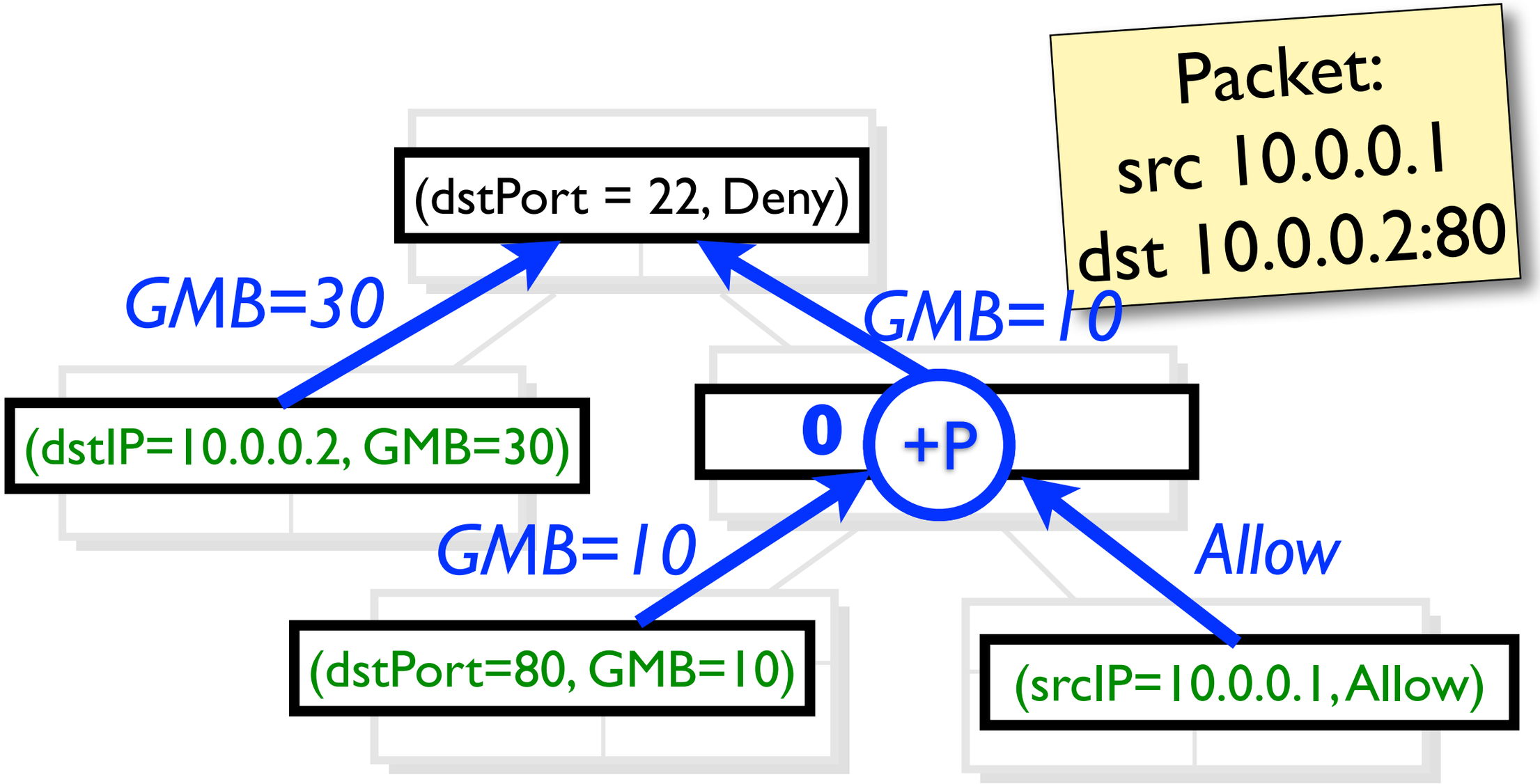
Hierarchical Flow Table



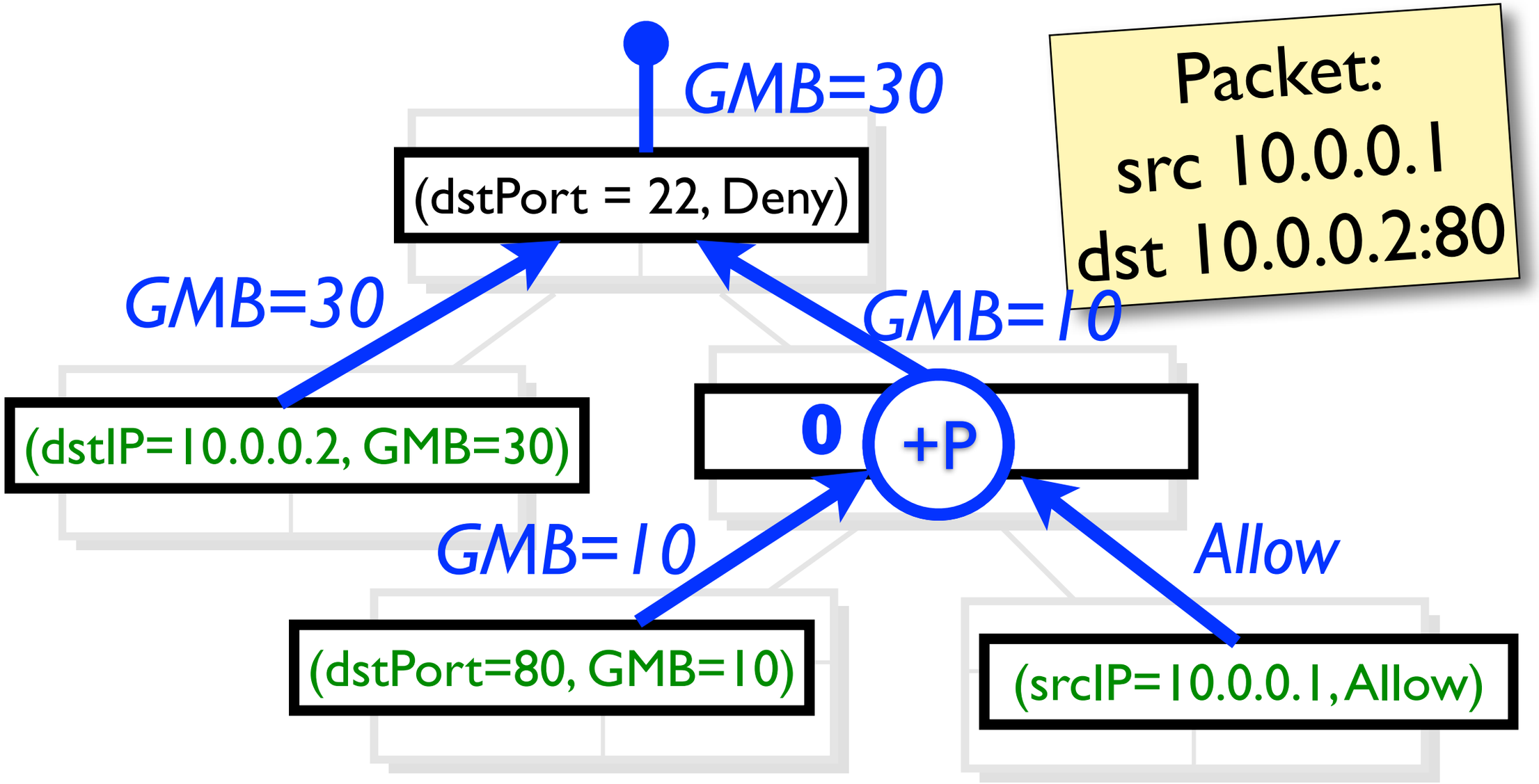
Hierarchical Flow Table



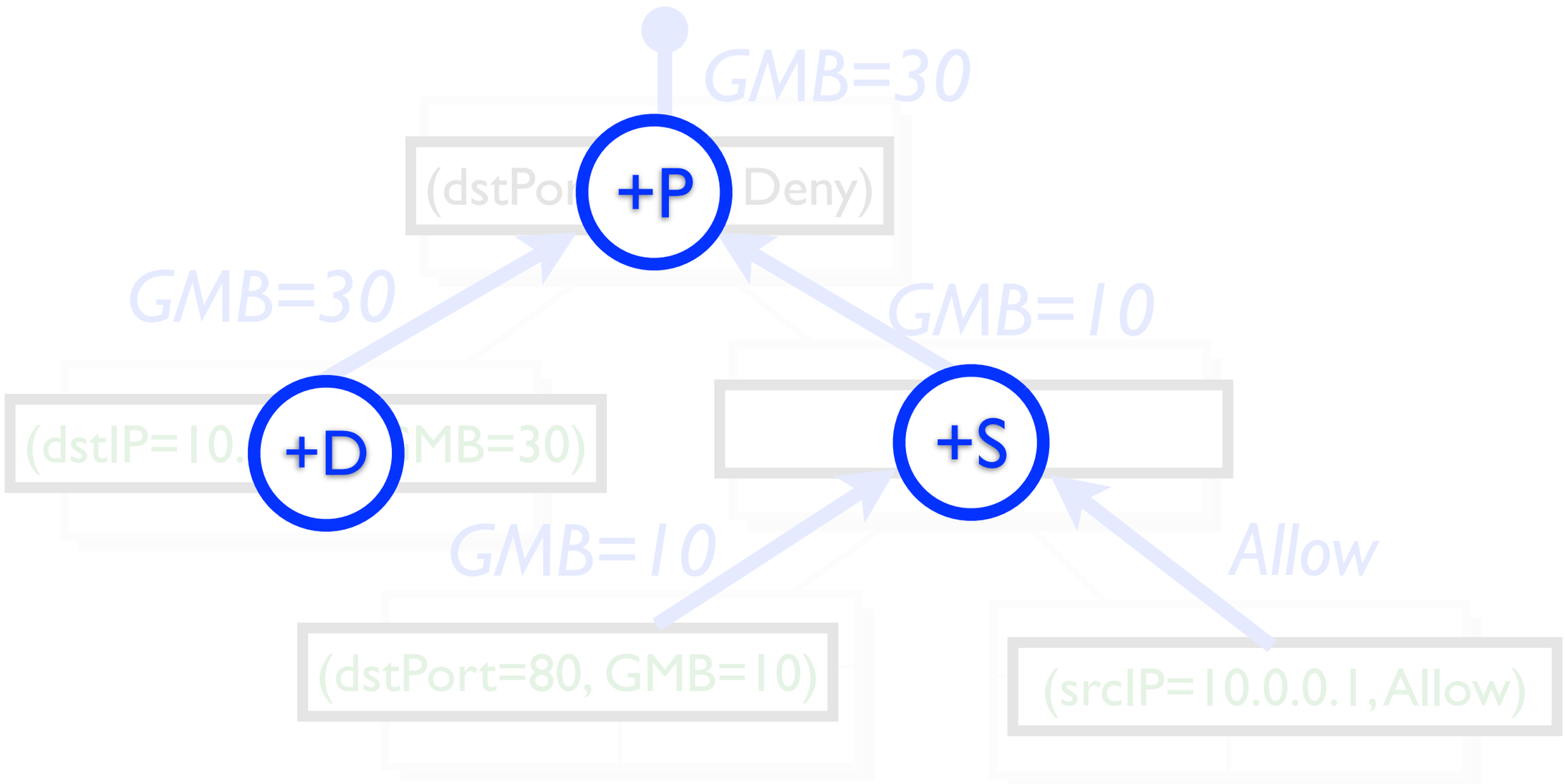
Hierarchical Flow Table



Hierarchical Flow Table

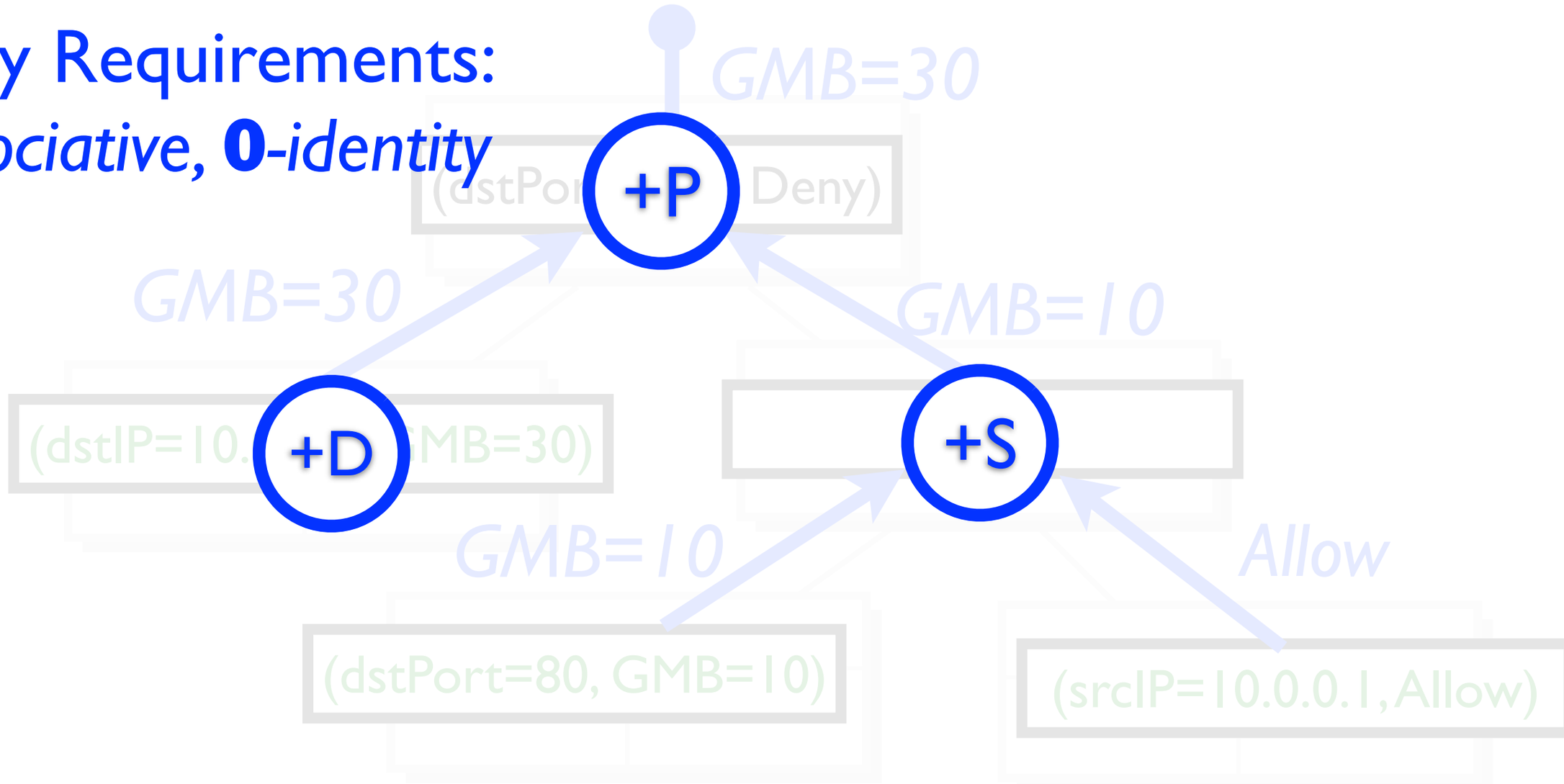


Hierarchical Flow Table



Hierarchical Flow Table

Only Requirements:
*Associative, **0**-identity*



Hierarchical Flow Table

+D *In node*

+S *Sibling*

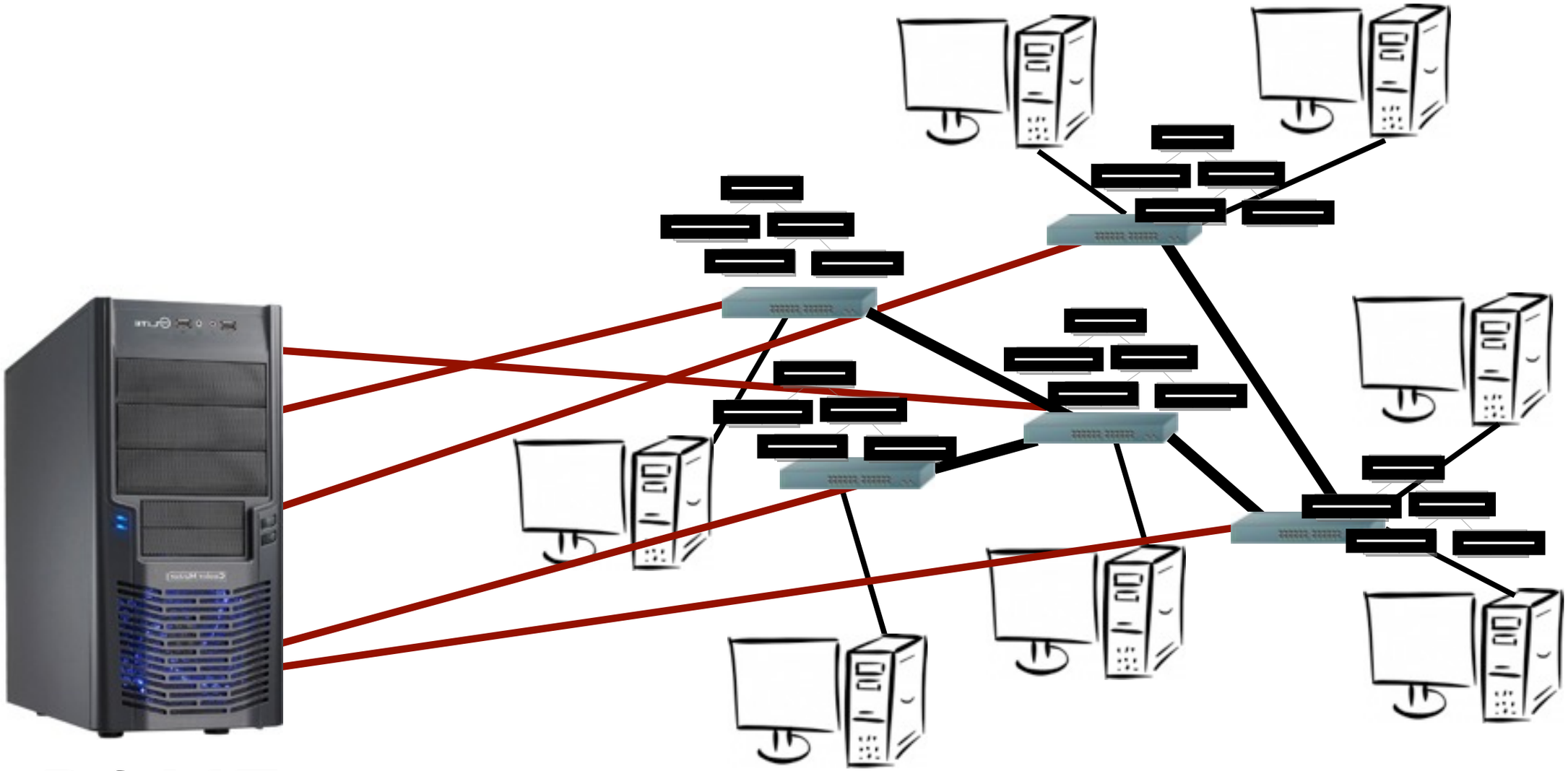
D and S identical.
Deny overrides Allow.
GMB combines as **max**

+P *Parent-Sibling*

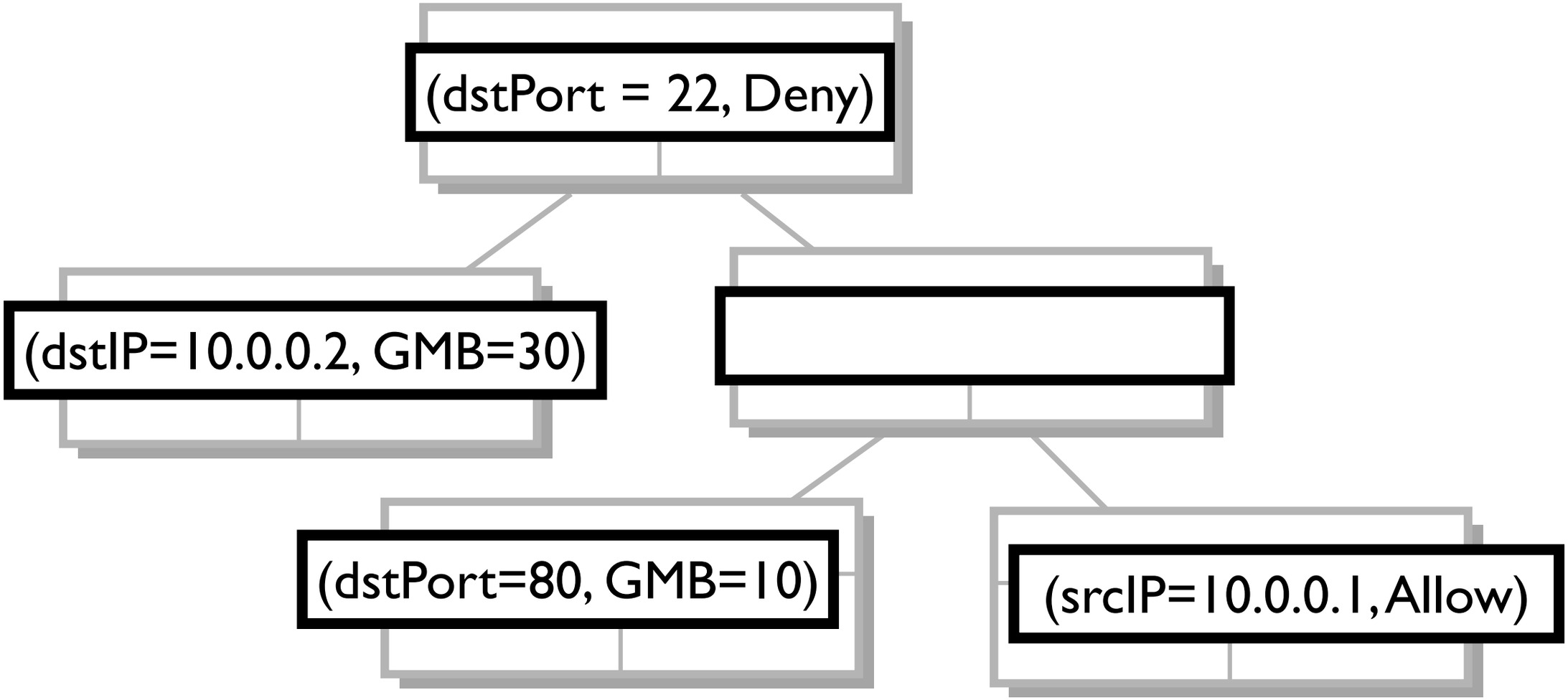
Child overrides Parent
for Access Control
GMB combines as **max**

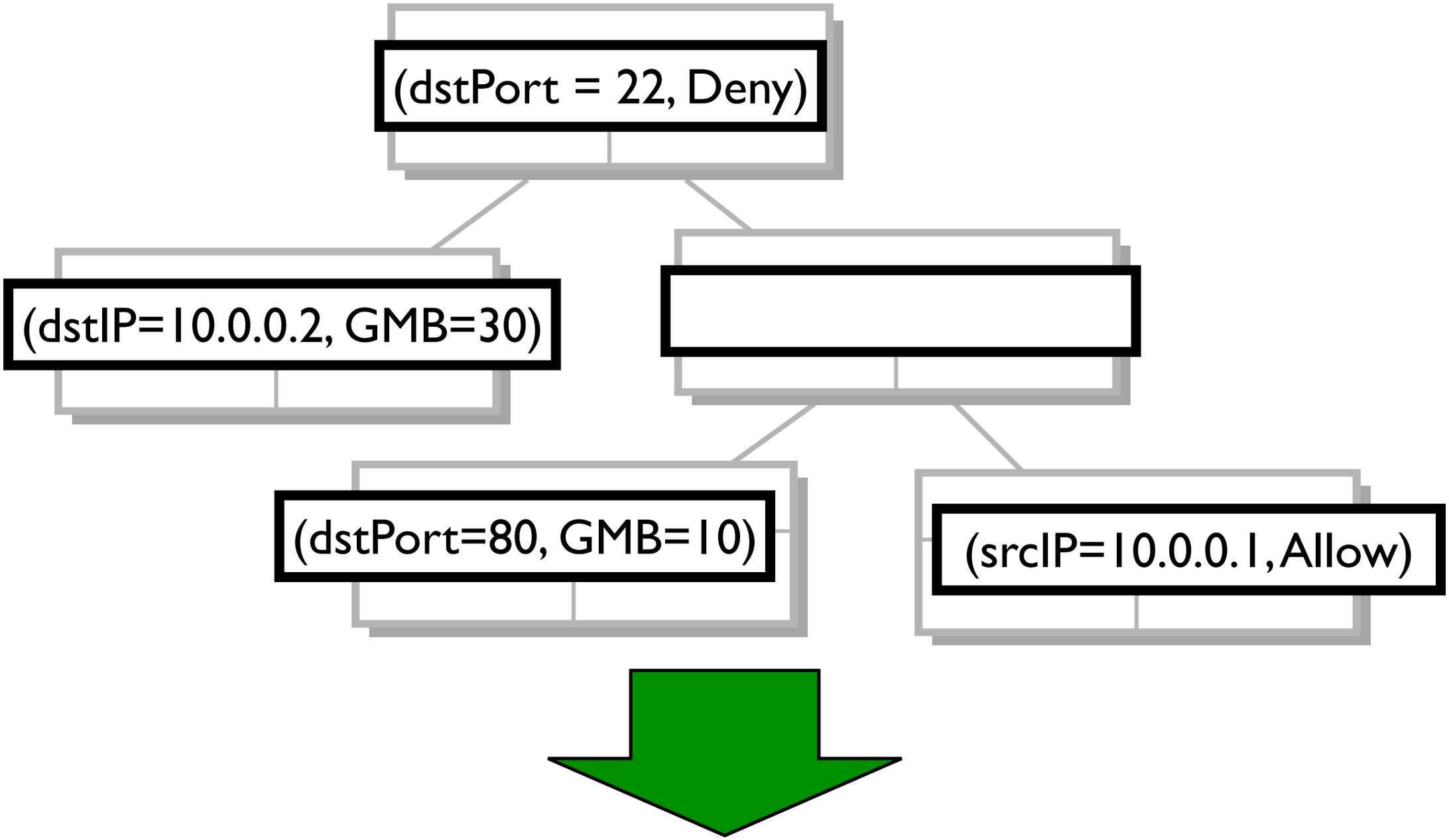
PANE's HFT Operators

Implementation



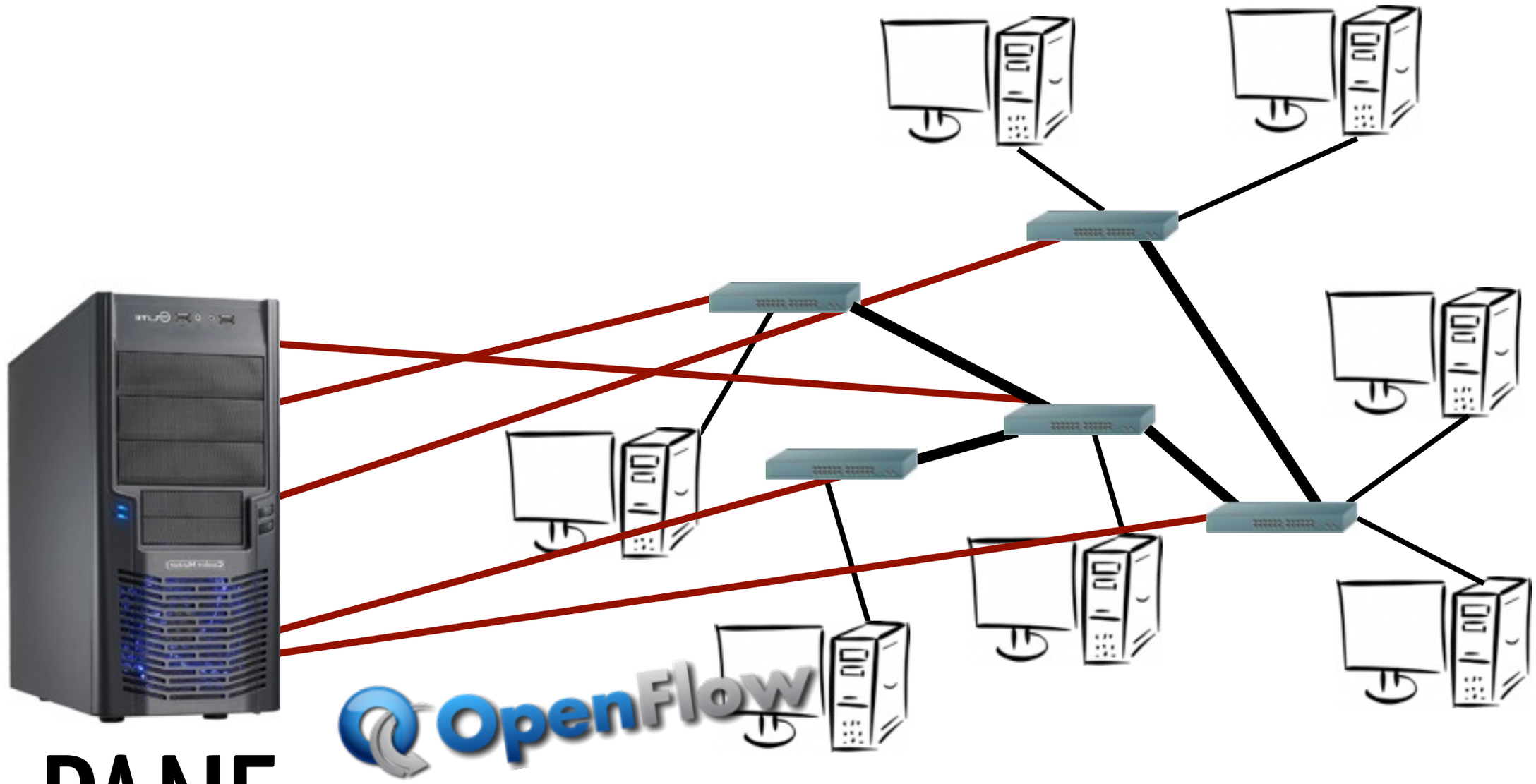
PANE


(dstPort = 22, Deny)
(dstIP=10.0.0.2, GMB=30)
(dstPort=80, GMB=10)
(srcIP=10.0.0.1, Allow)

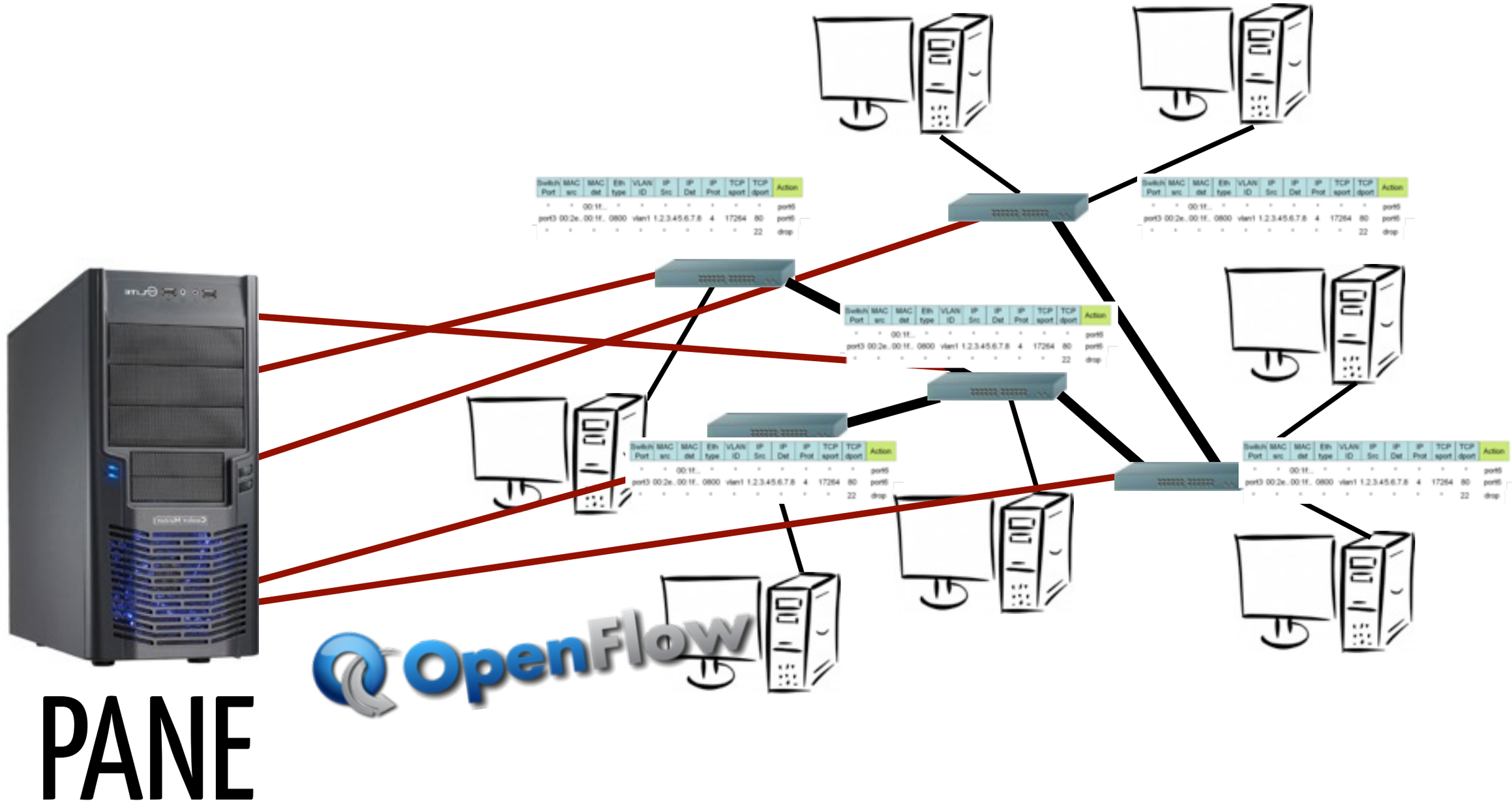


Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

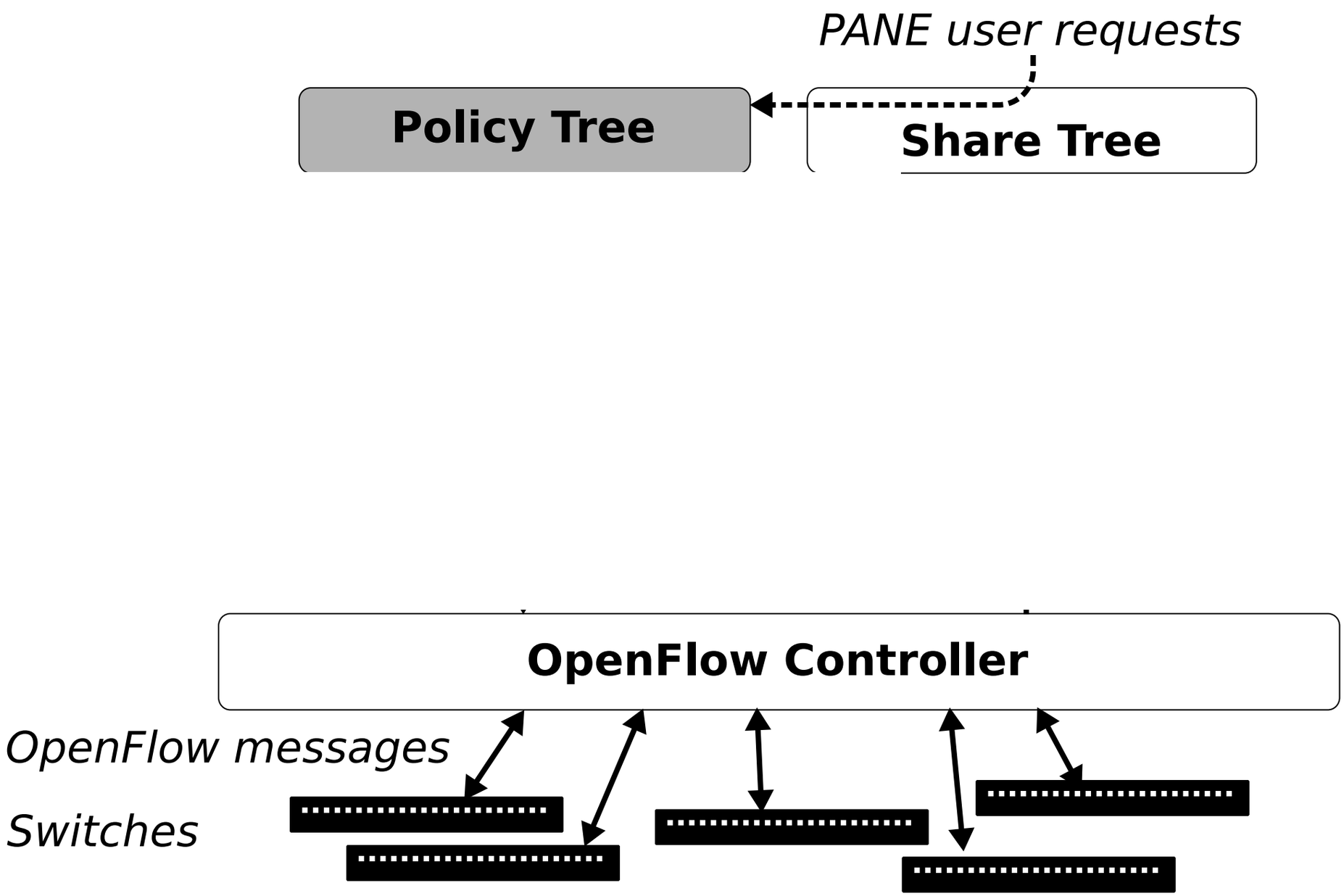
OpenFlow



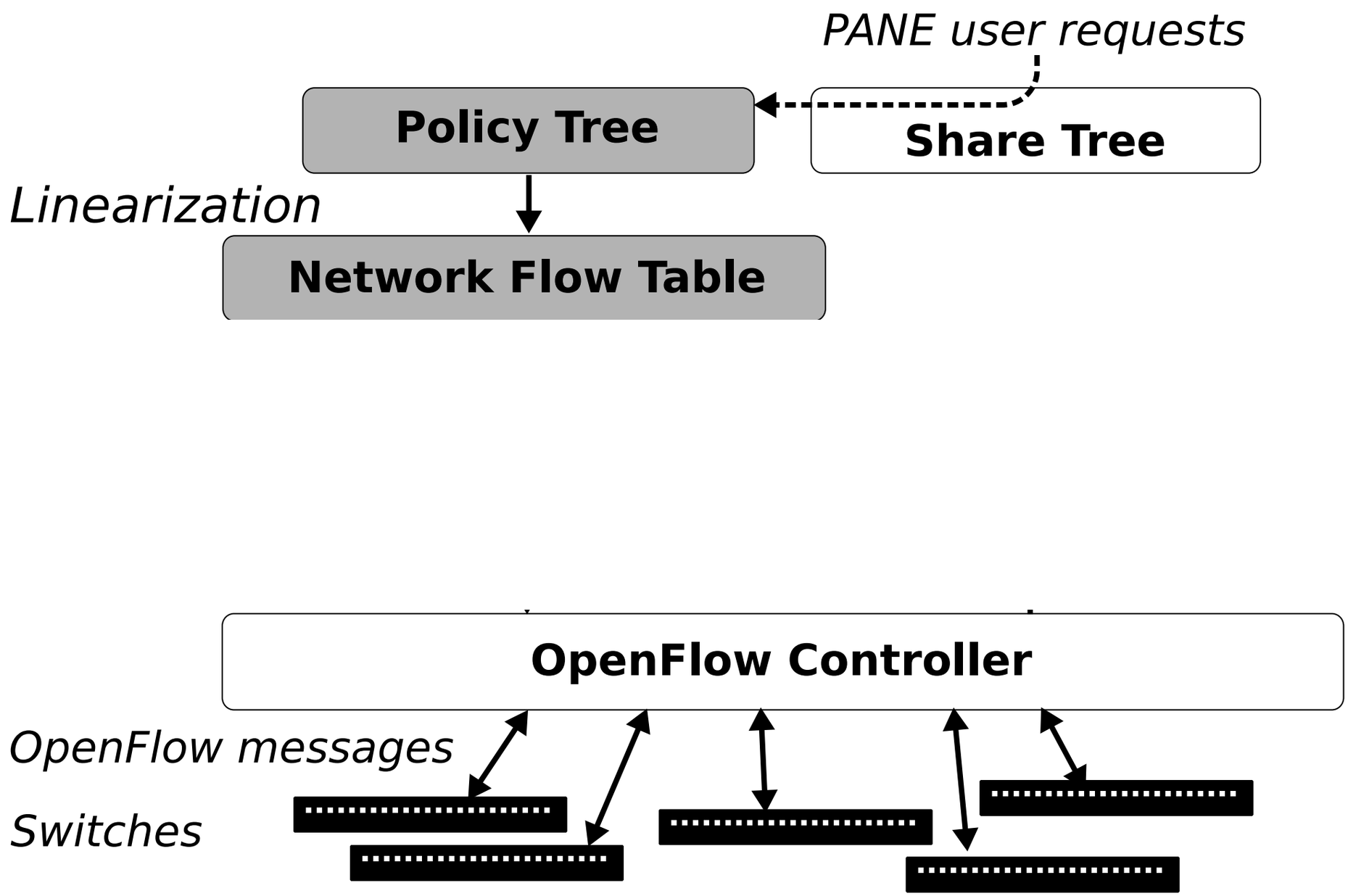
PANE


OpenFlow
PANE

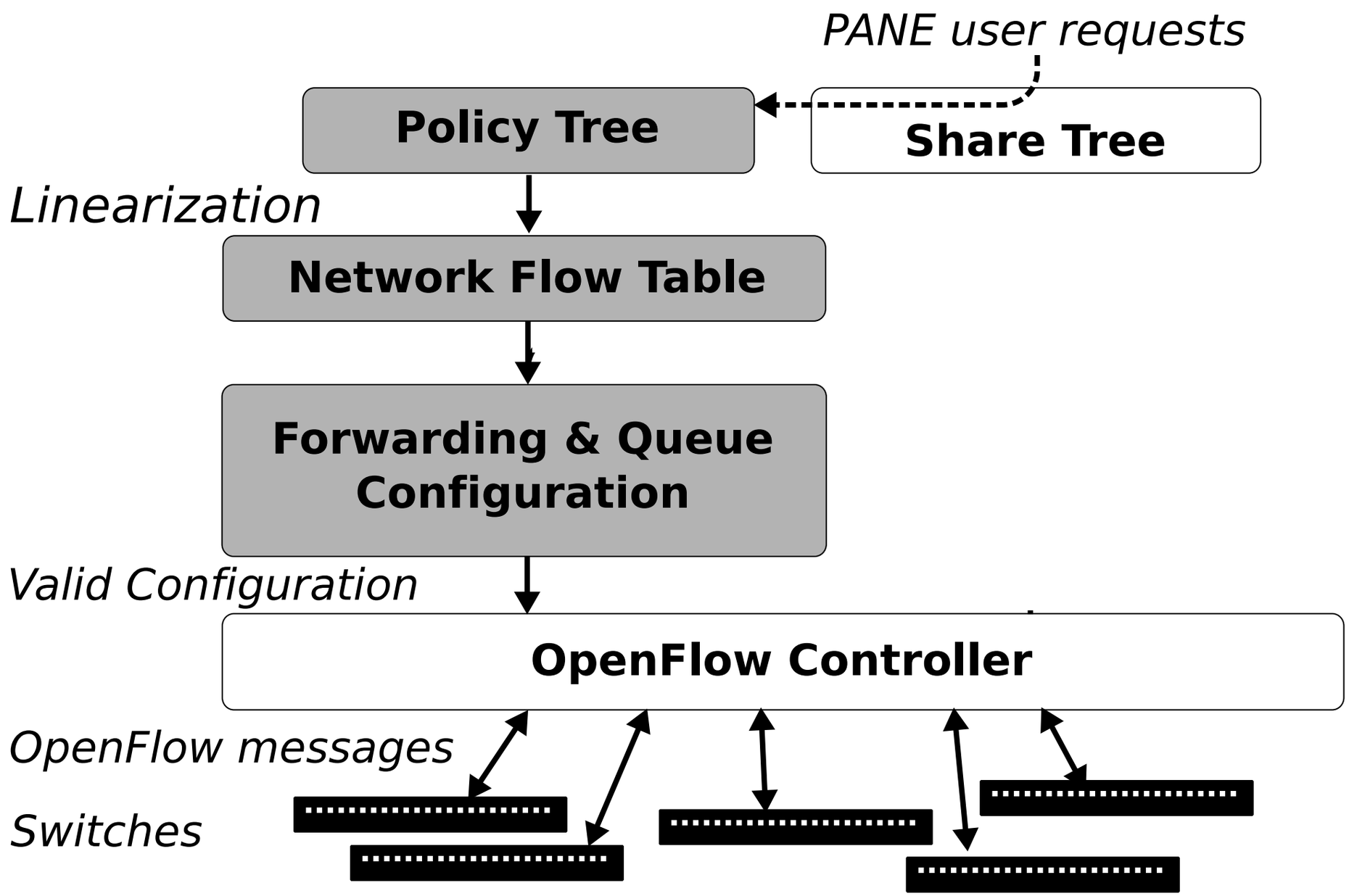
PANE


PANE user requests
Policy Tree
Share Tree
OpenFlow Controller
OpenFlow messages
Switches

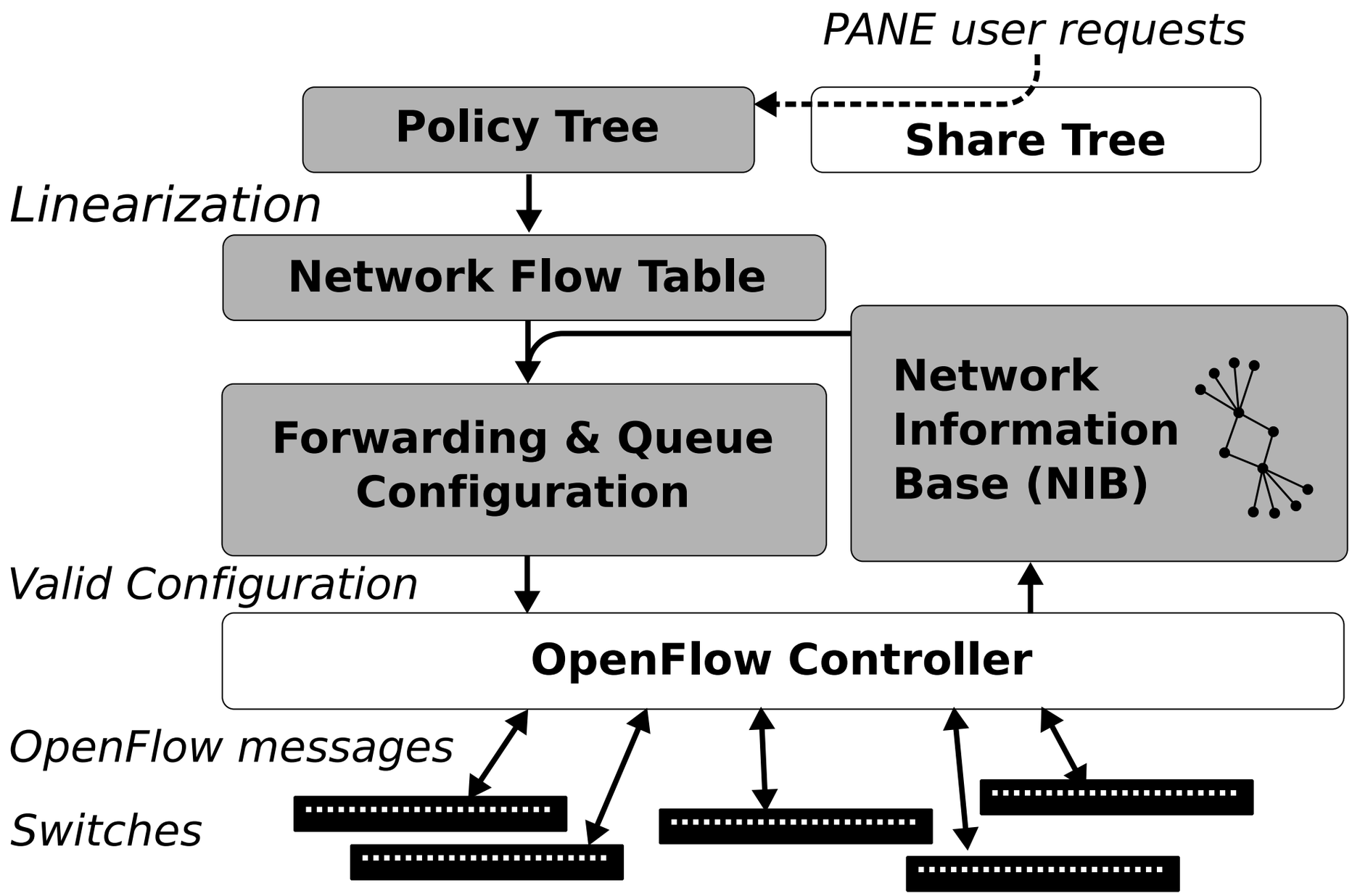
PANE

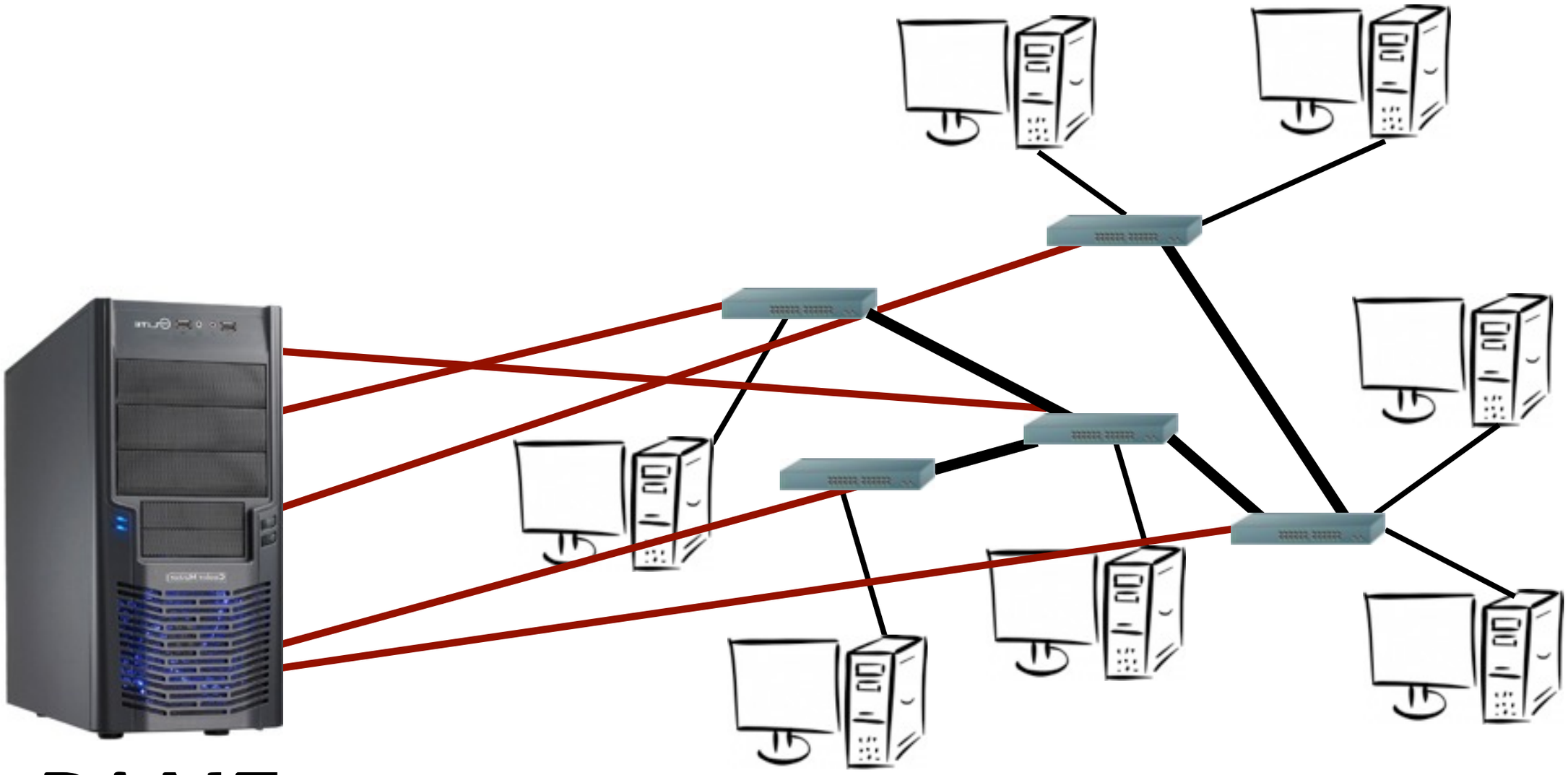

PANE user requests
Policy Tree
Share Tree
Linearization
Network Flow Table
OpenFlow Controller
OpenFlow messages
Switches

PANE

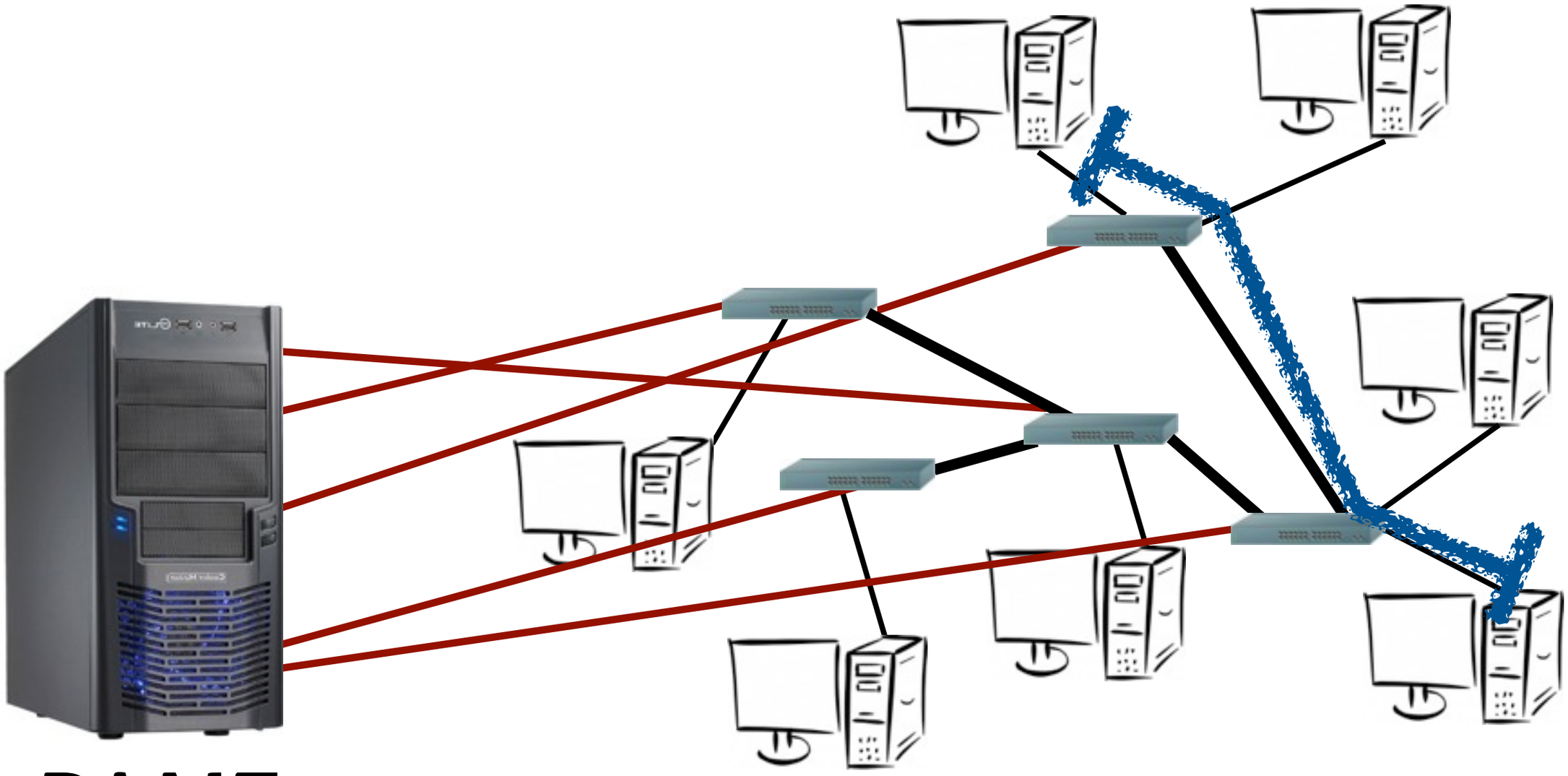

PANE user requests
Policy Tree
Share Tree
Linearization
Network Flow Table
Forwarding & Queue Configuration
Valid Configuration
OpenFlow Controller
OpenFlow messages
Switches

PANE

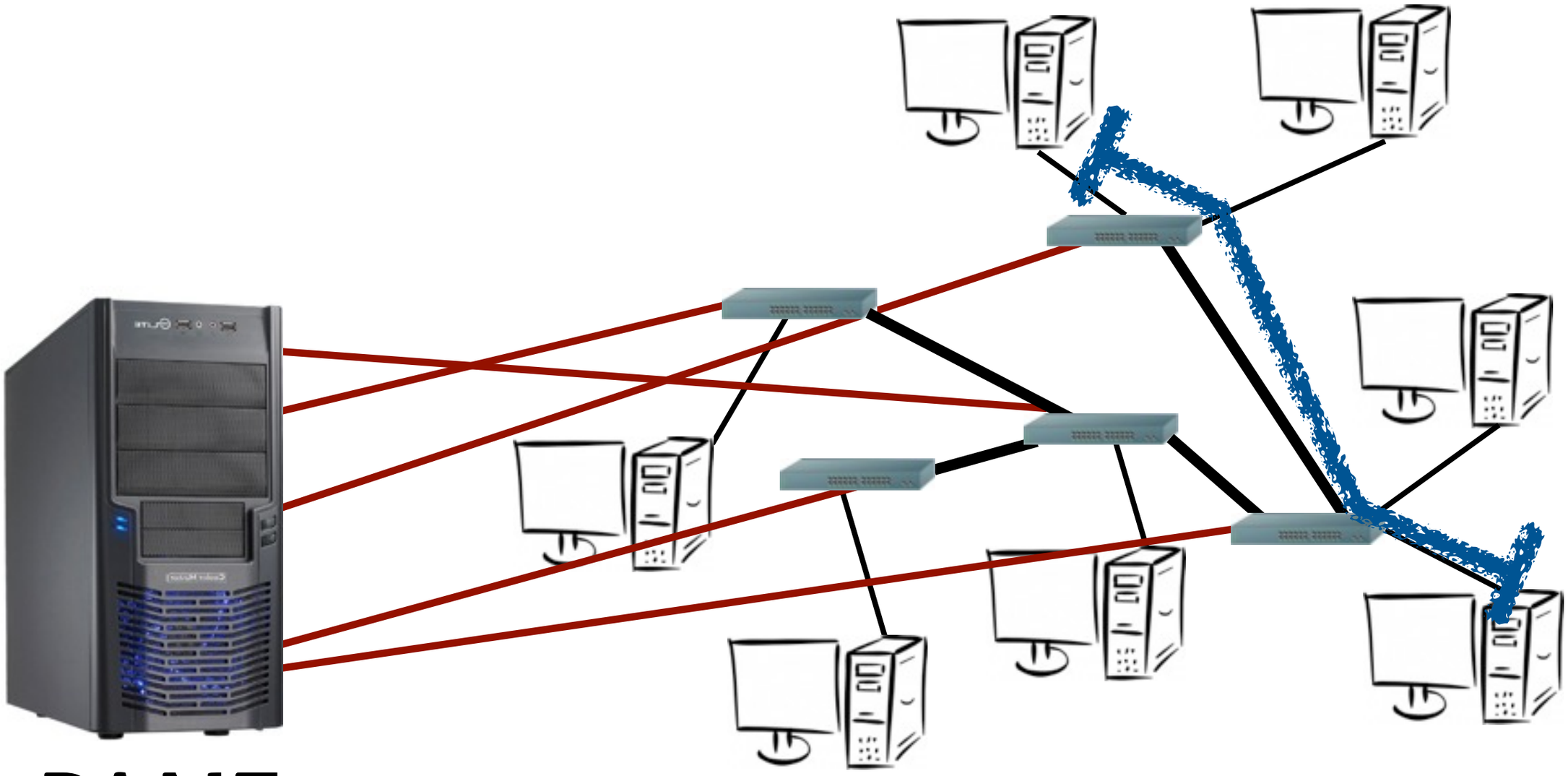

PANE user requests
Policy Tree
Share Tree
Linearization
Network Flow Table
Forwarding & Queue Configuration
Network Information Base (NIB)
Valid Configuration
OpenFlow Controller
OpenFlow messages
Switches



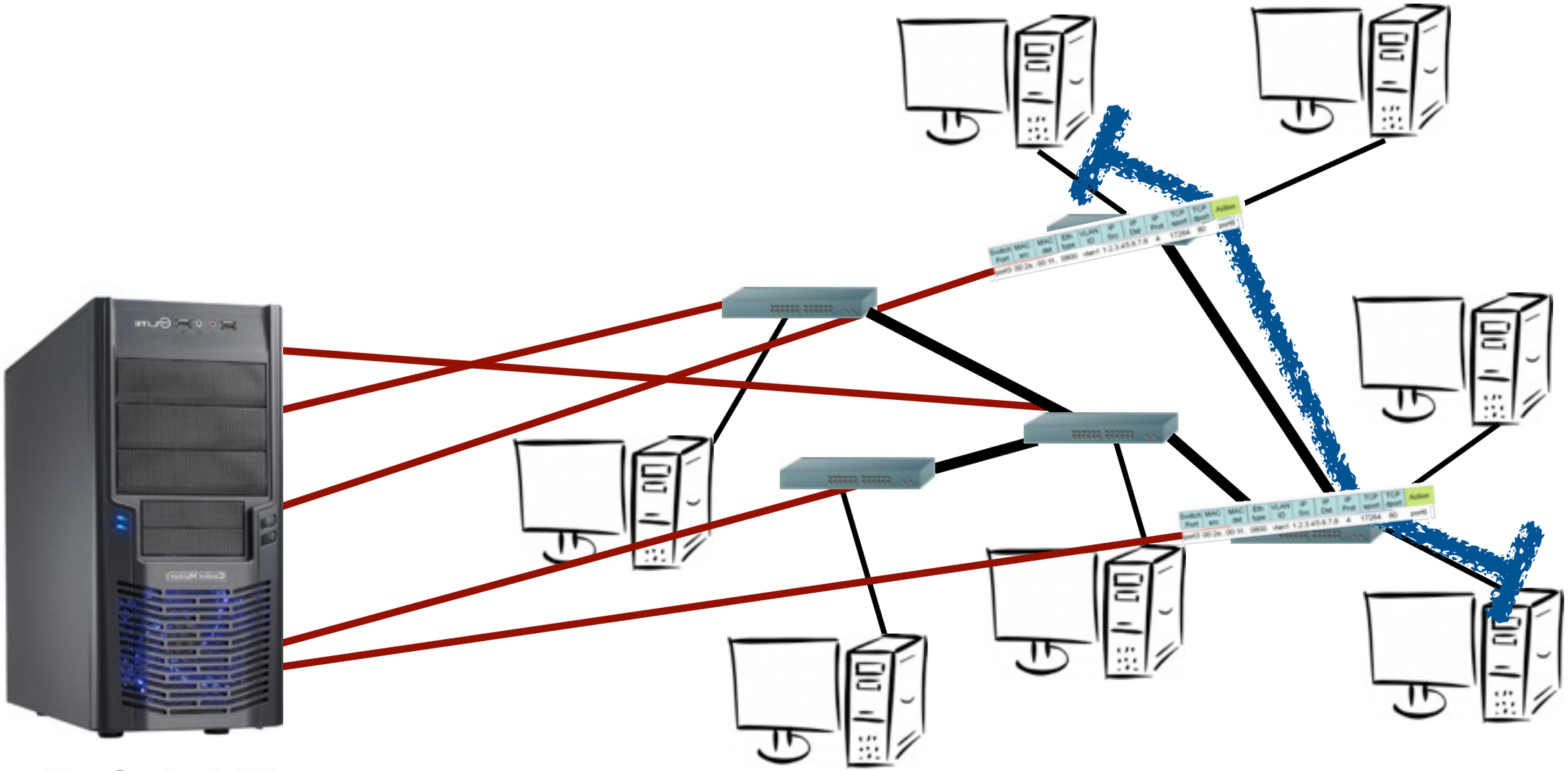
PANE



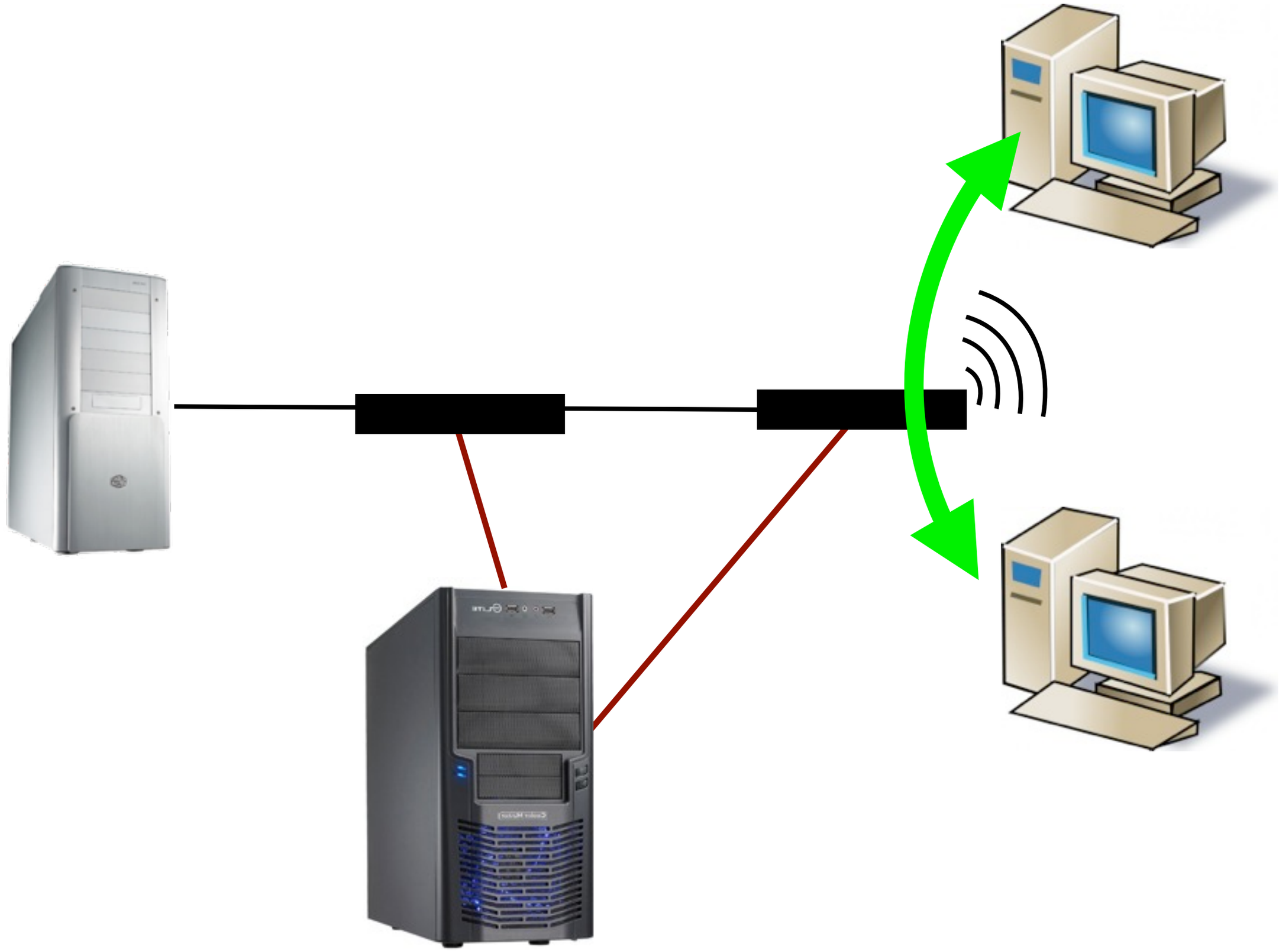
PANE



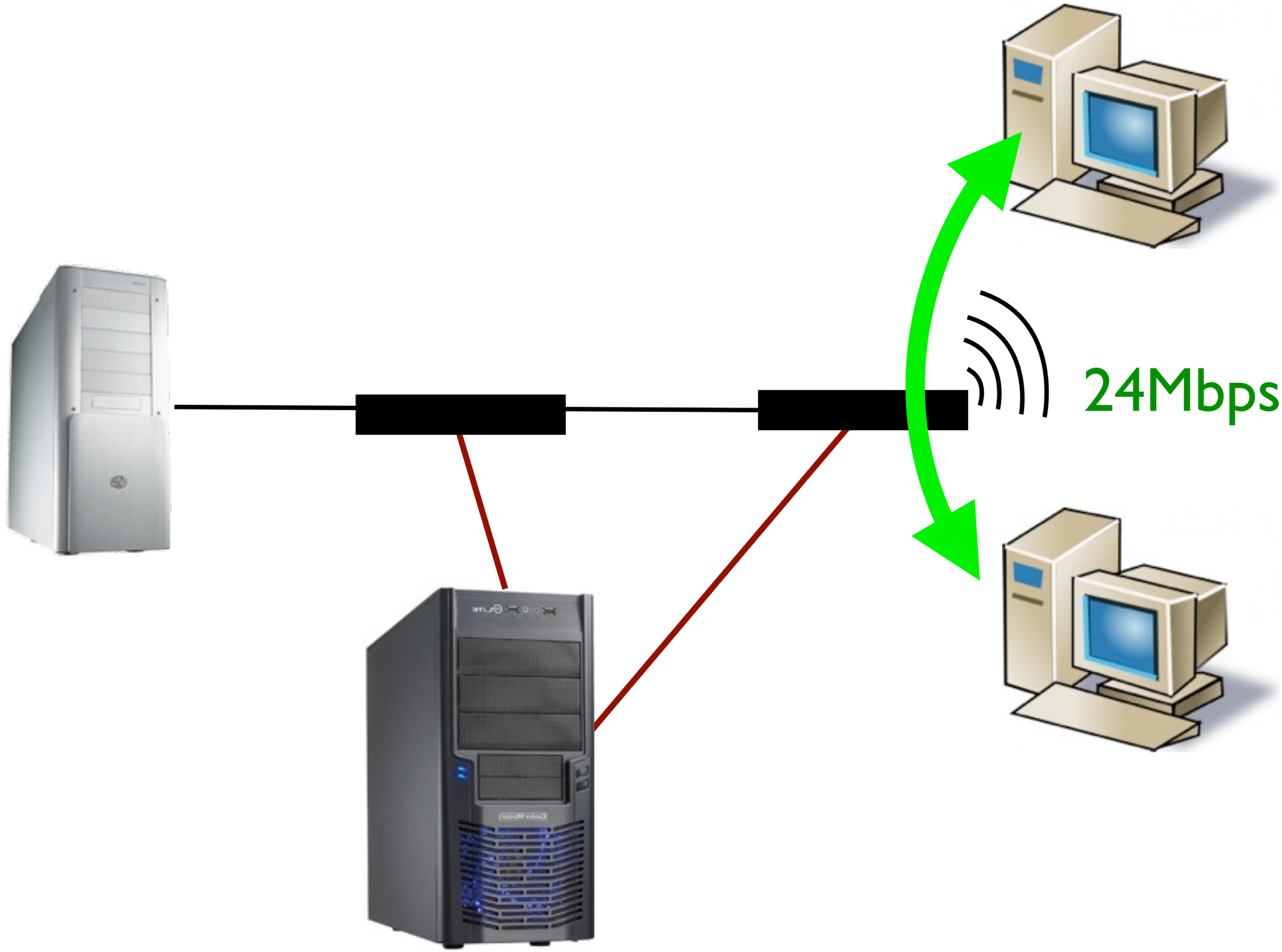
PANE



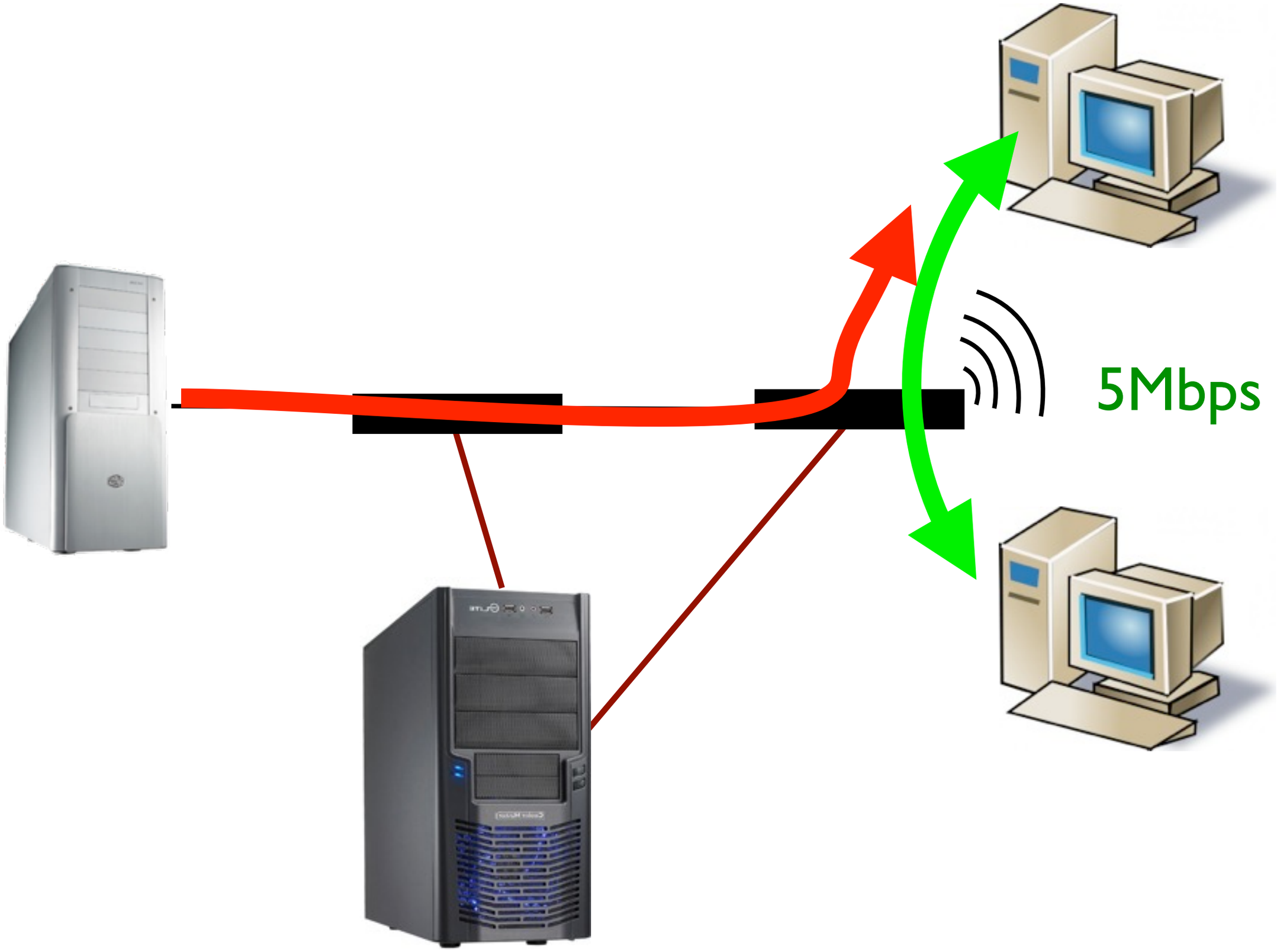
PANE



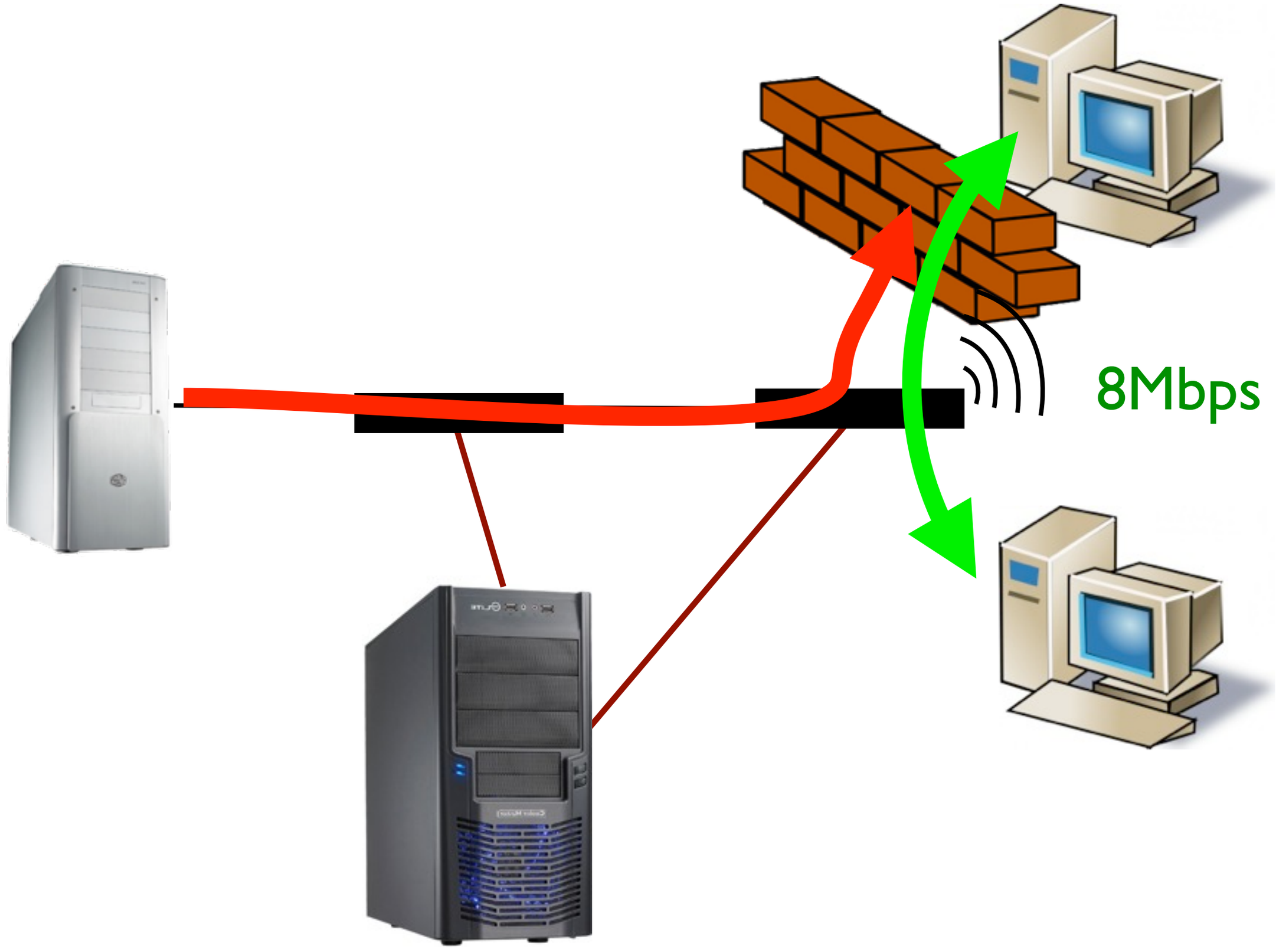
PANE



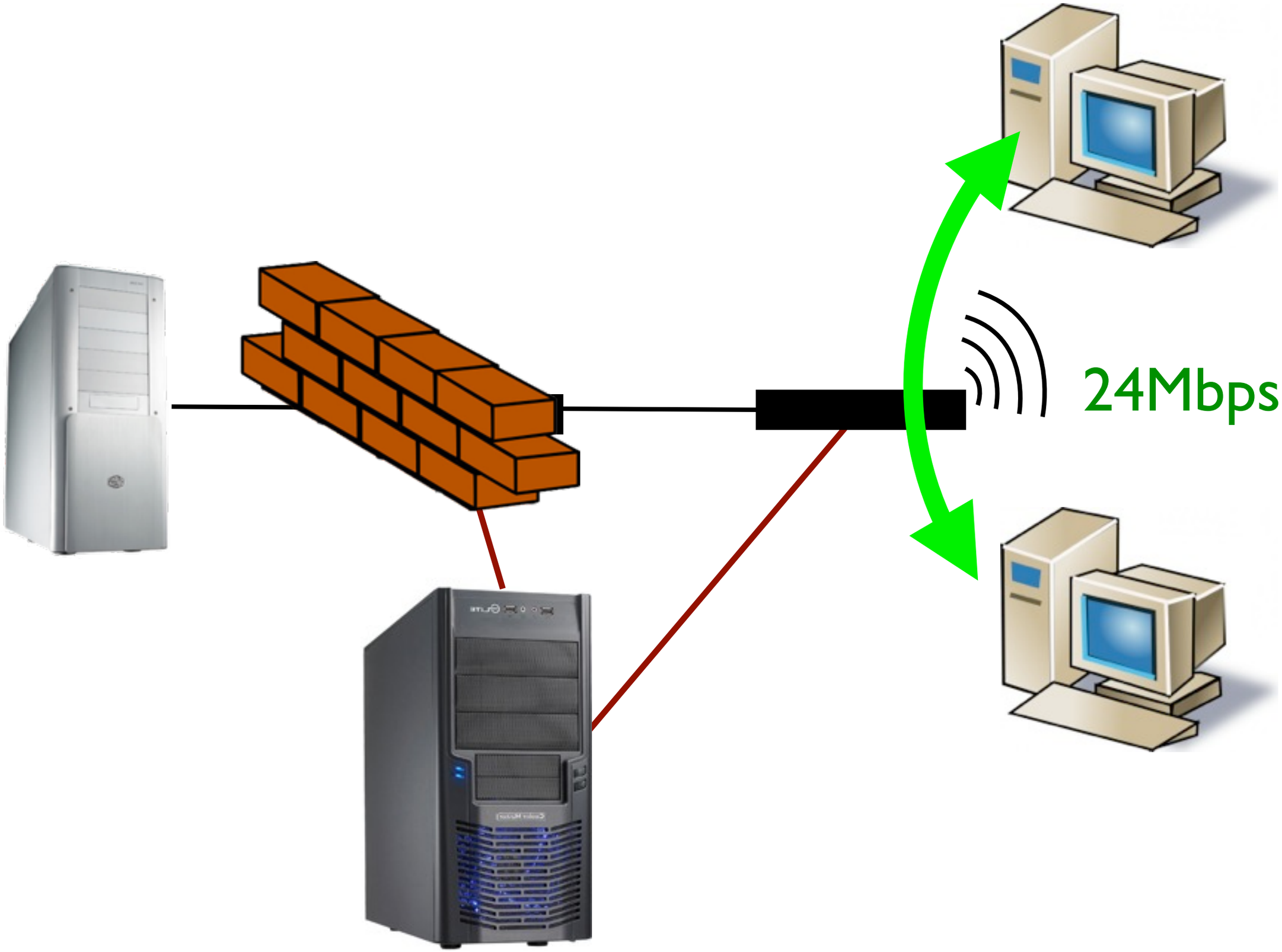
PANE



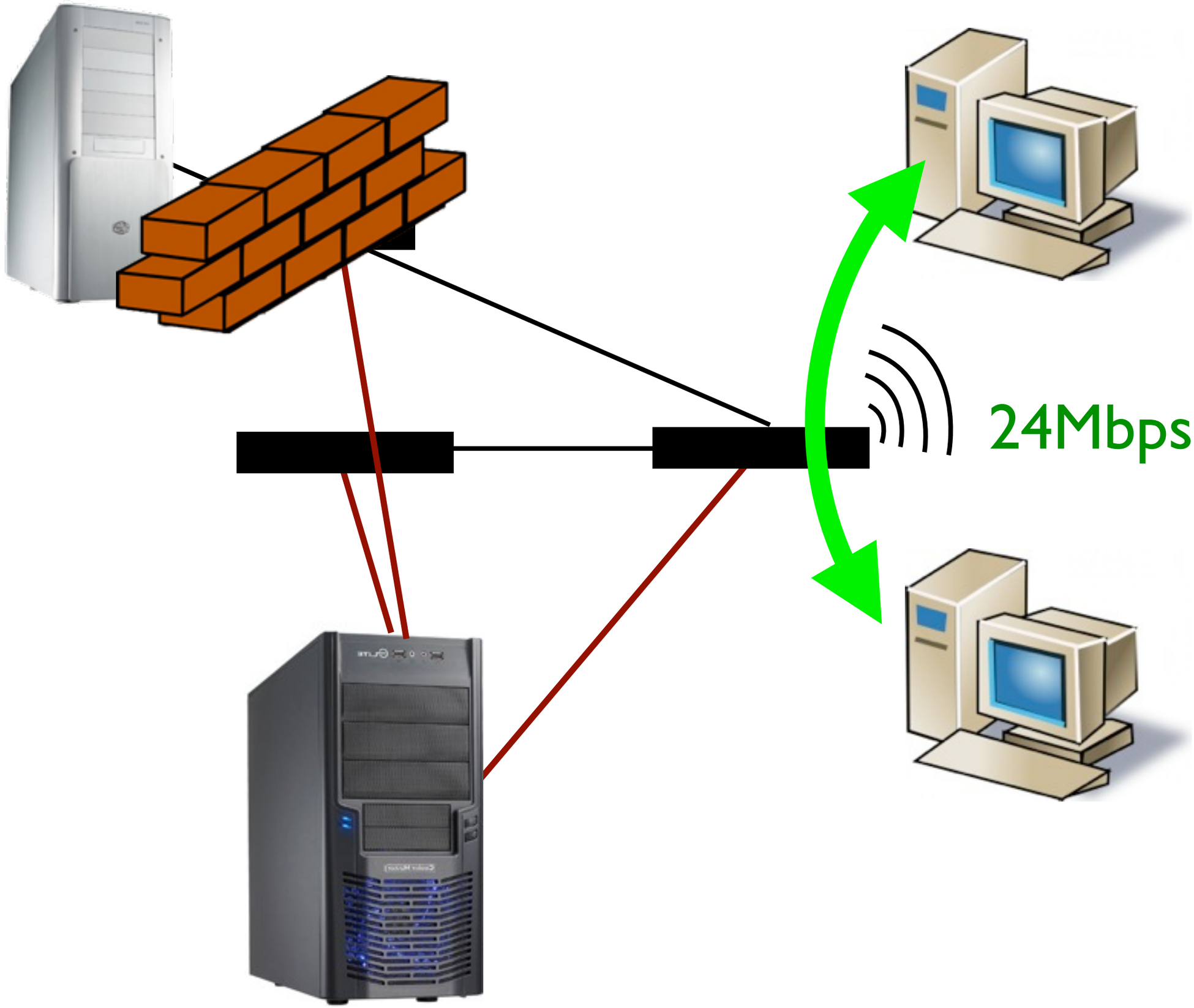
PANE



PANE

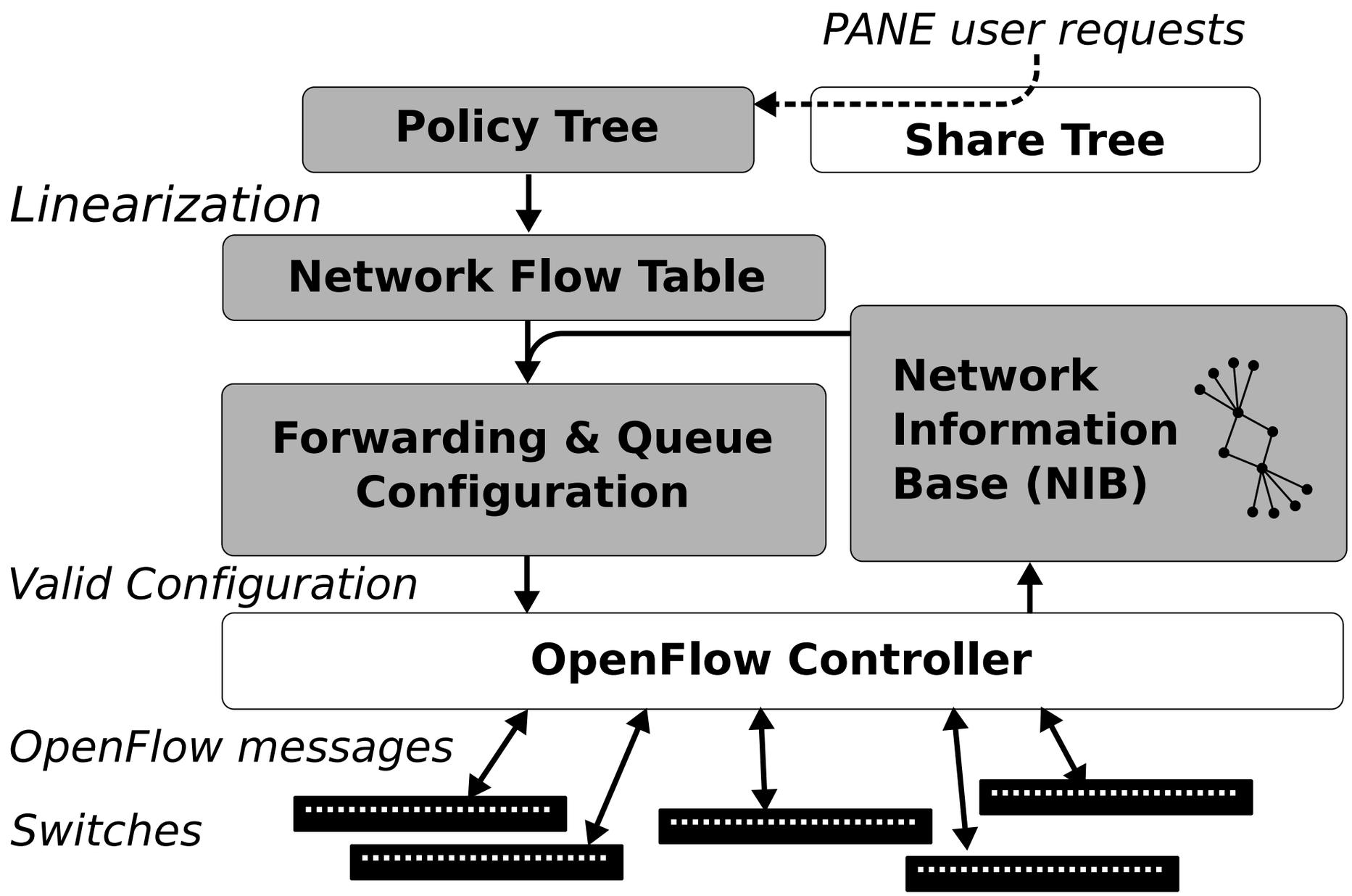


PANE

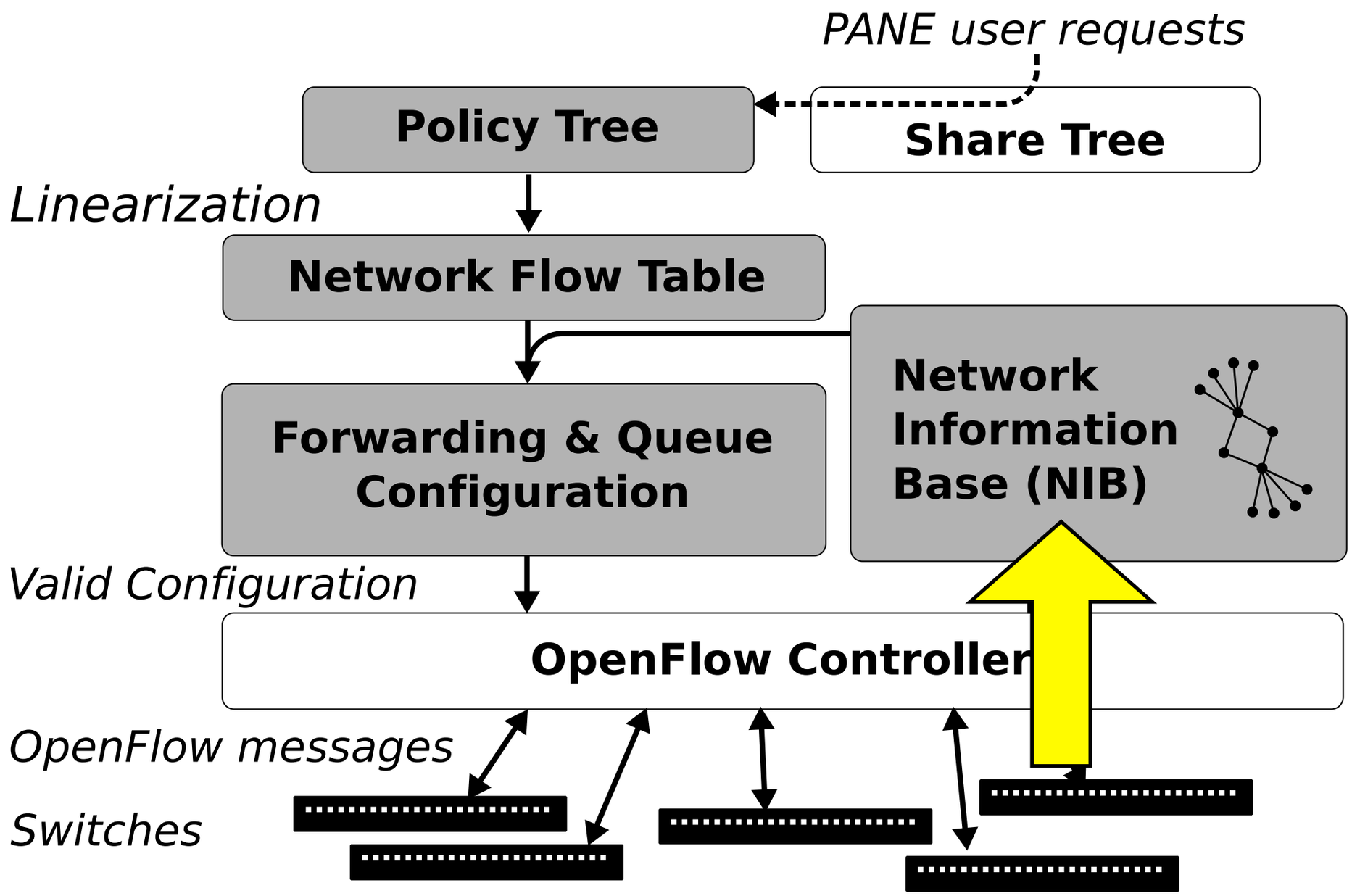


PANE

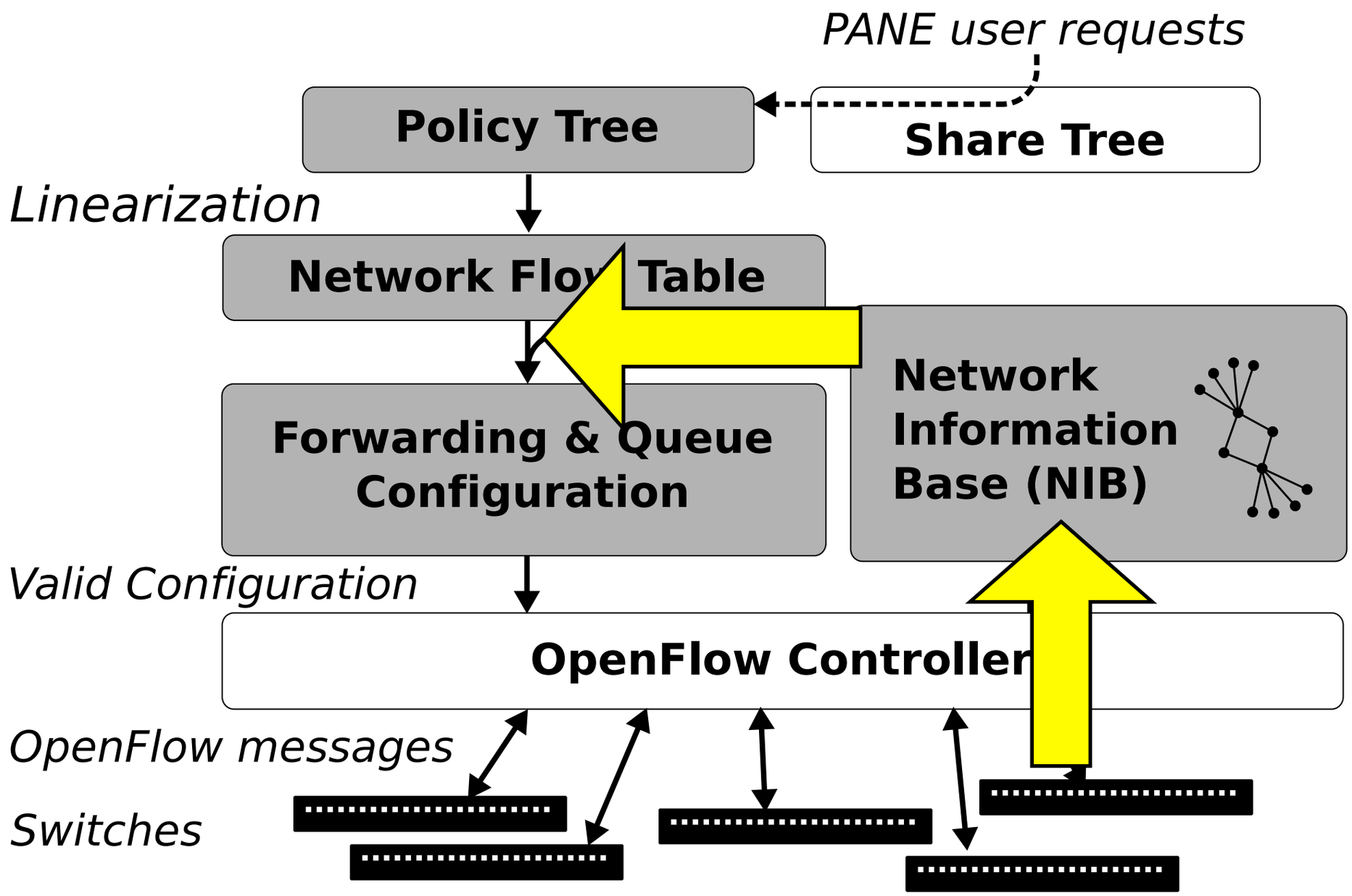
PANE


PANE user requests
Policy Tree
Share Tree
Linearization
Network Flow Table
Forwarding & Queue Configuration
Network Information Base (NIB)
Valid Configuration
OpenFlow Controller
OpenFlow messages
Switches

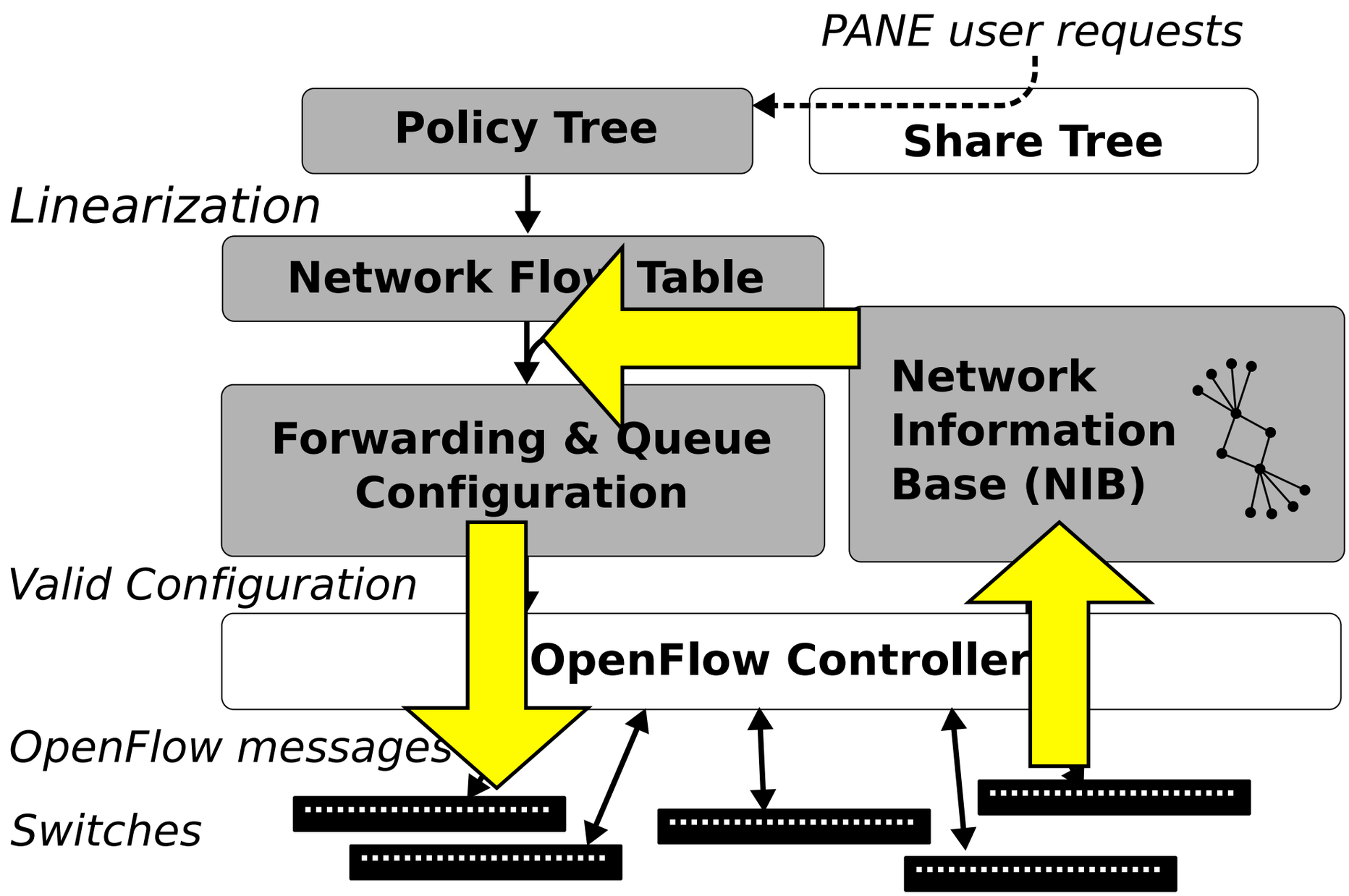
PANE


PANE user requests
Policy Tree
Share Tree
Linearization
Network Flow Table
Forwarding & Queue Configuration
Network Information Base (NIB)
Valid Configuration
OpenFlow Controller
OpenFlow messages
Switches

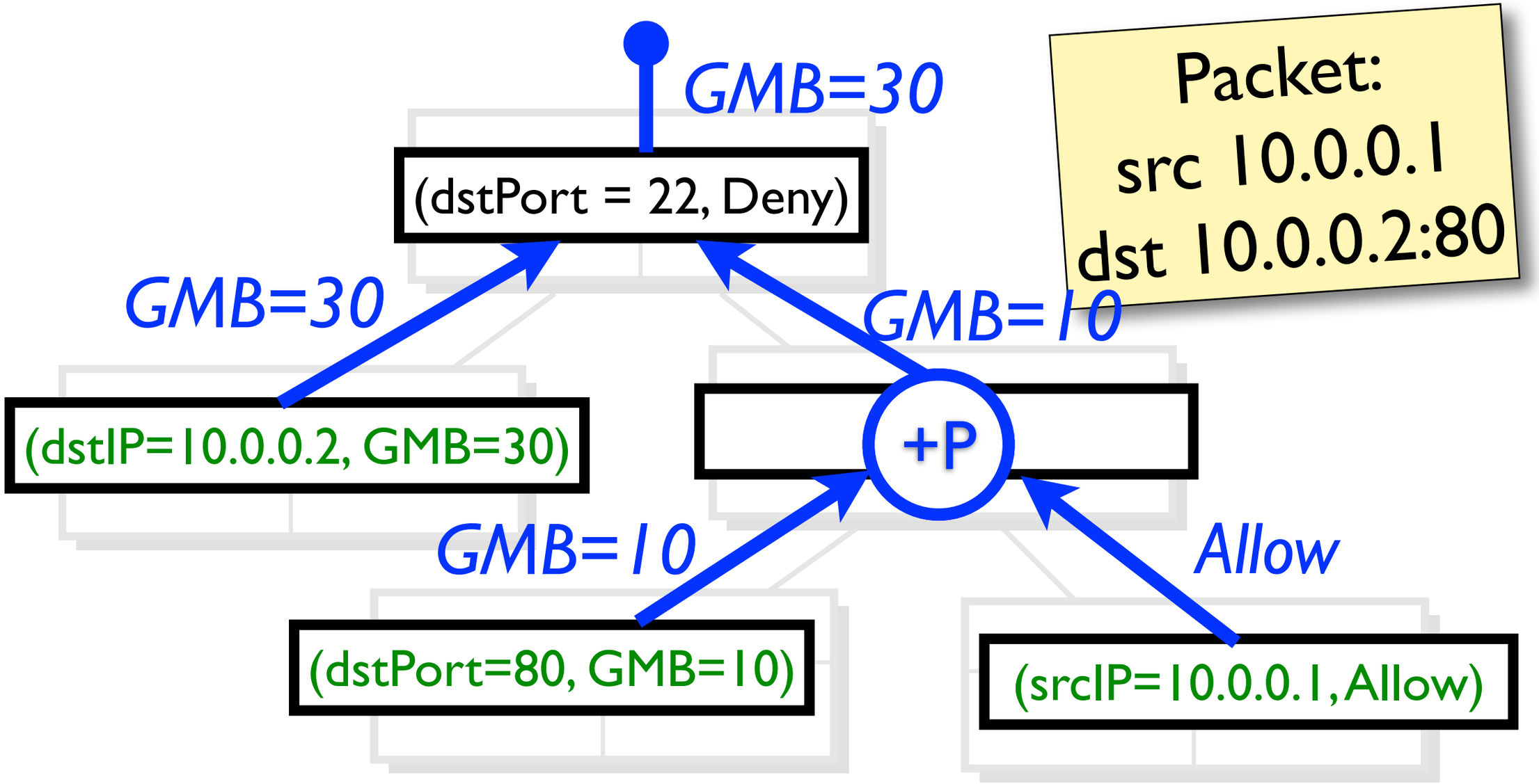
PANE


PANE user requests
Policy Tree
Share Tree
Linearization
Network Flow Table
Forwarding & Queue Configuration
Network Information Base (NIB)
Valid Configuration
OpenFlow Controller
OpenFlow messages
Switches

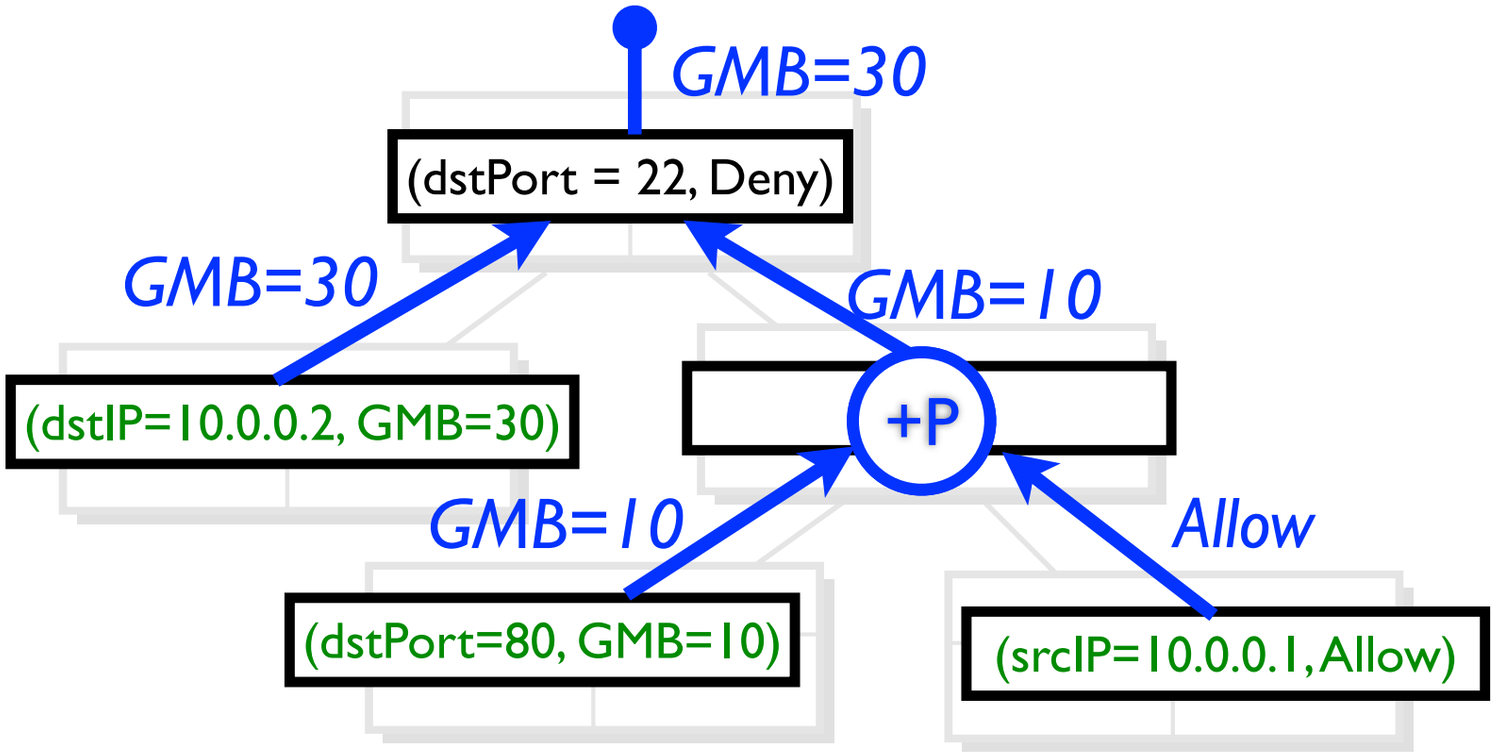
PANE


PANE user requests
Policy Tree
Share Tree
Linearization
Network Flow Table
Forwarding & Queue Configuration
Network Information Base (NIB)
Valid Configuration
OpenFlow Controller
OpenFlow messages
Switches

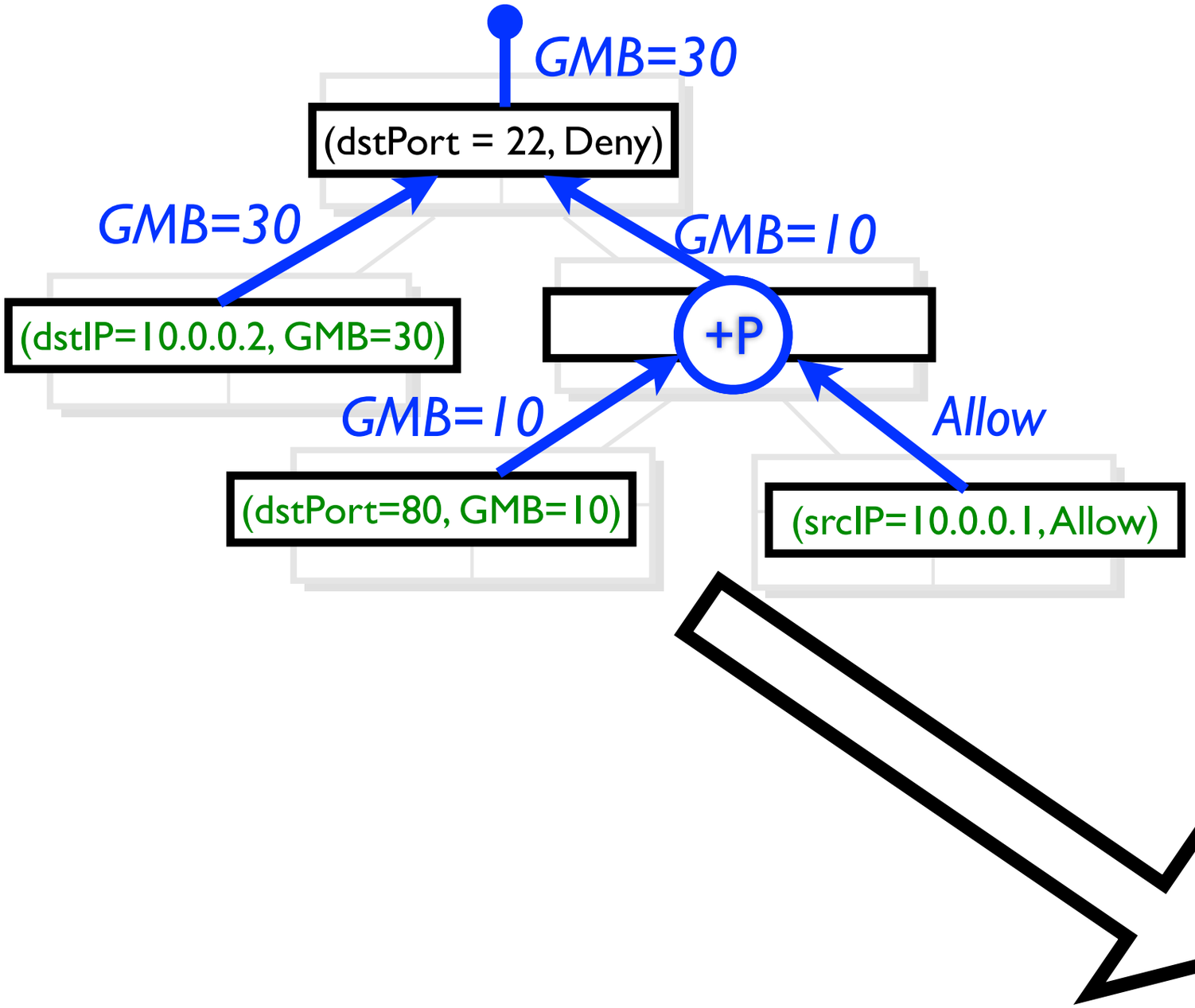
Proof of Correctness



Hierarchical Flow Tables

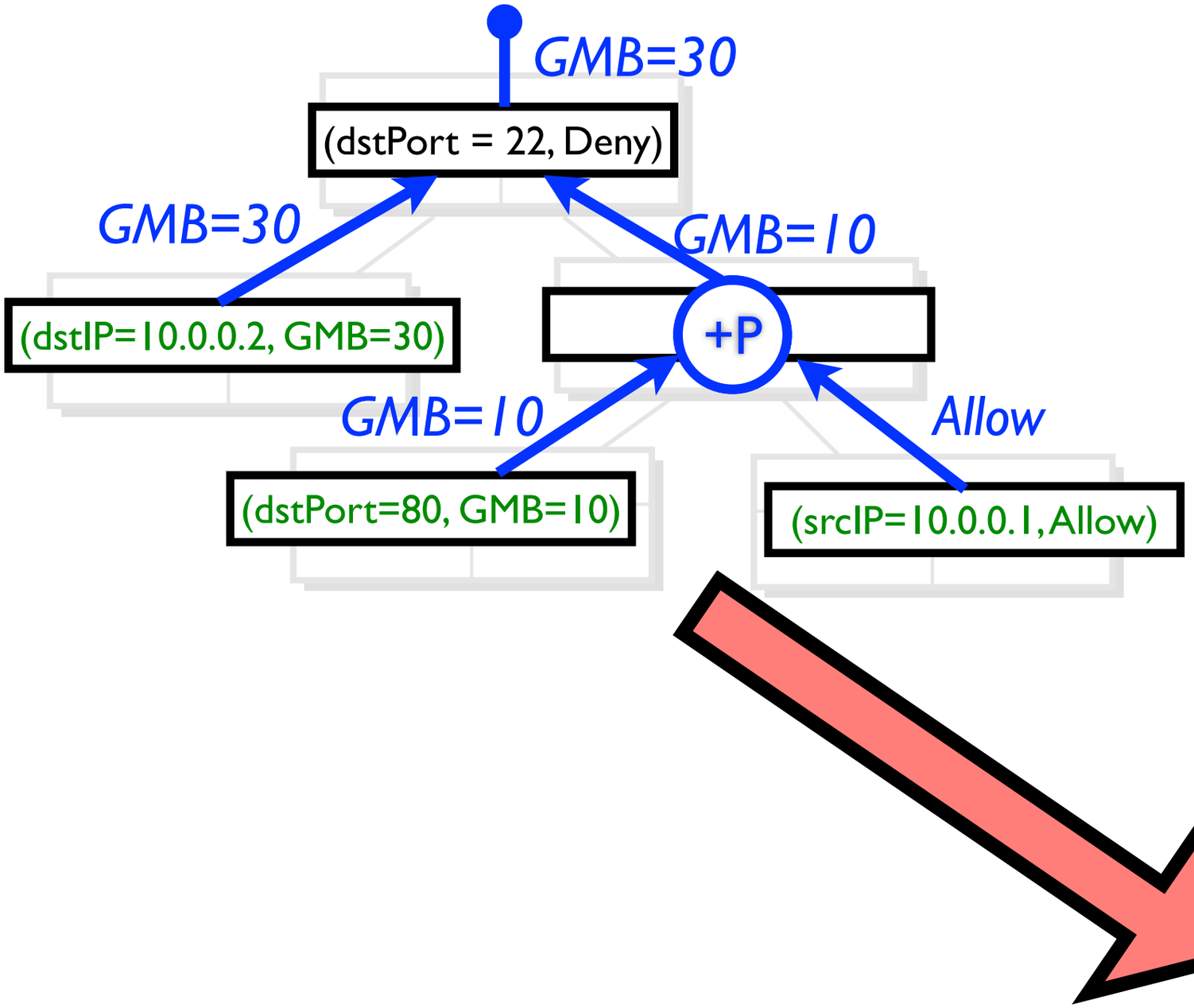


Compiler Correctness



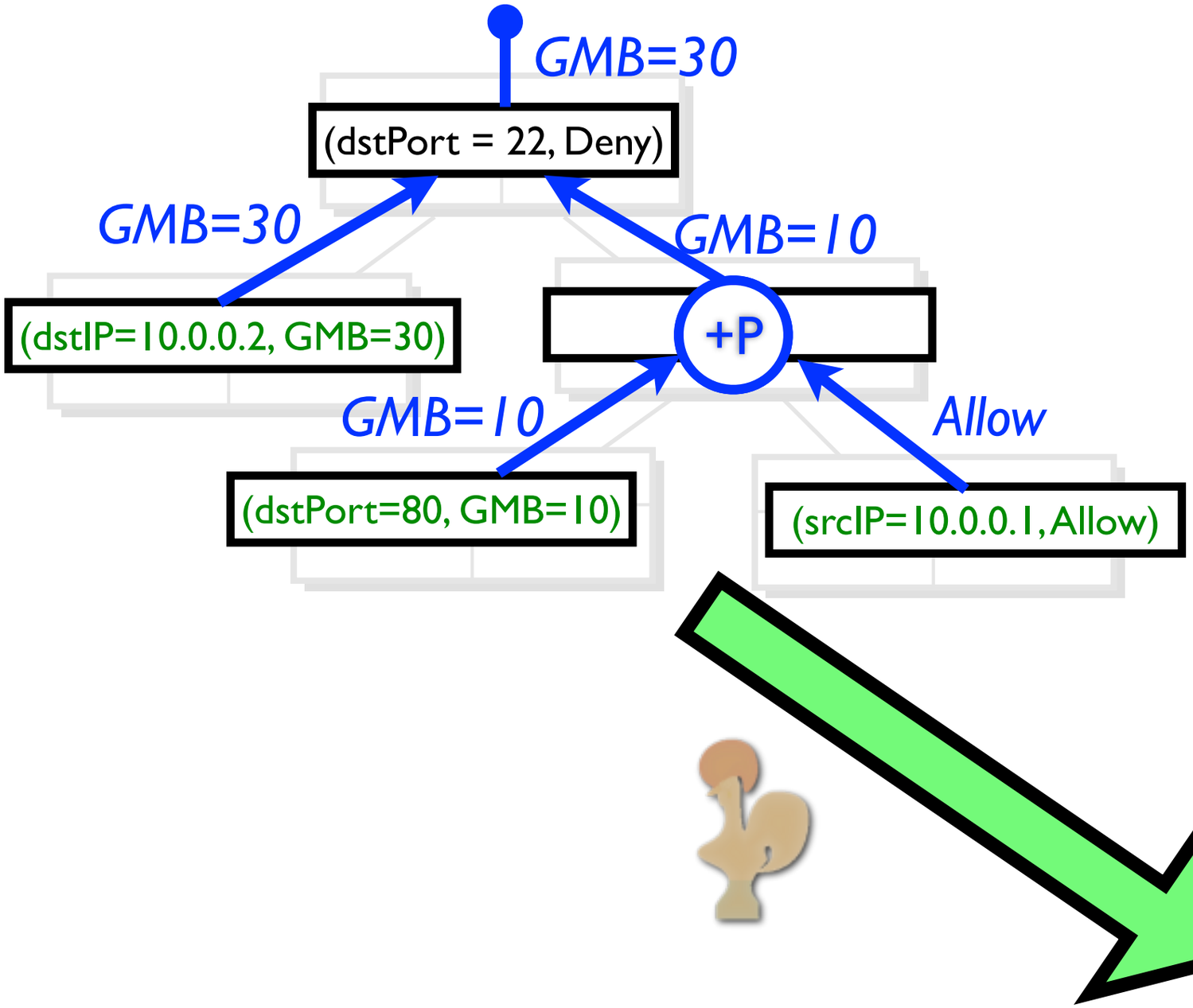
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

Compiler Correctness



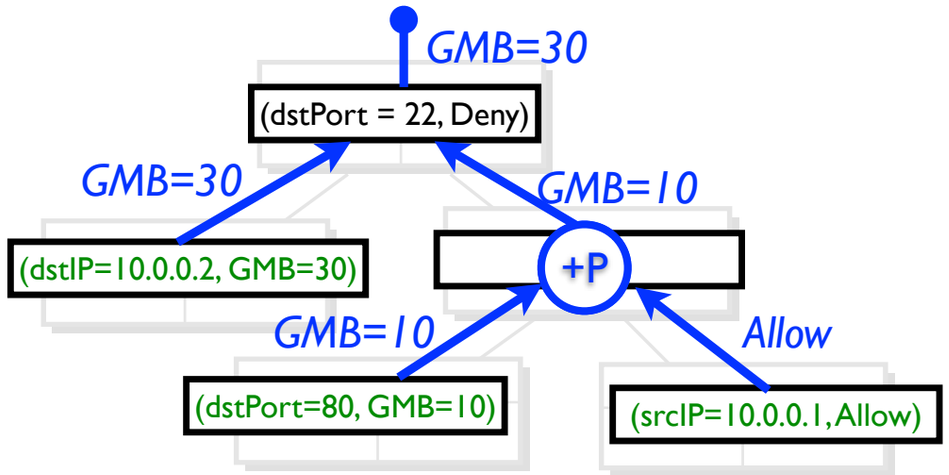
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

Compiler Correctness



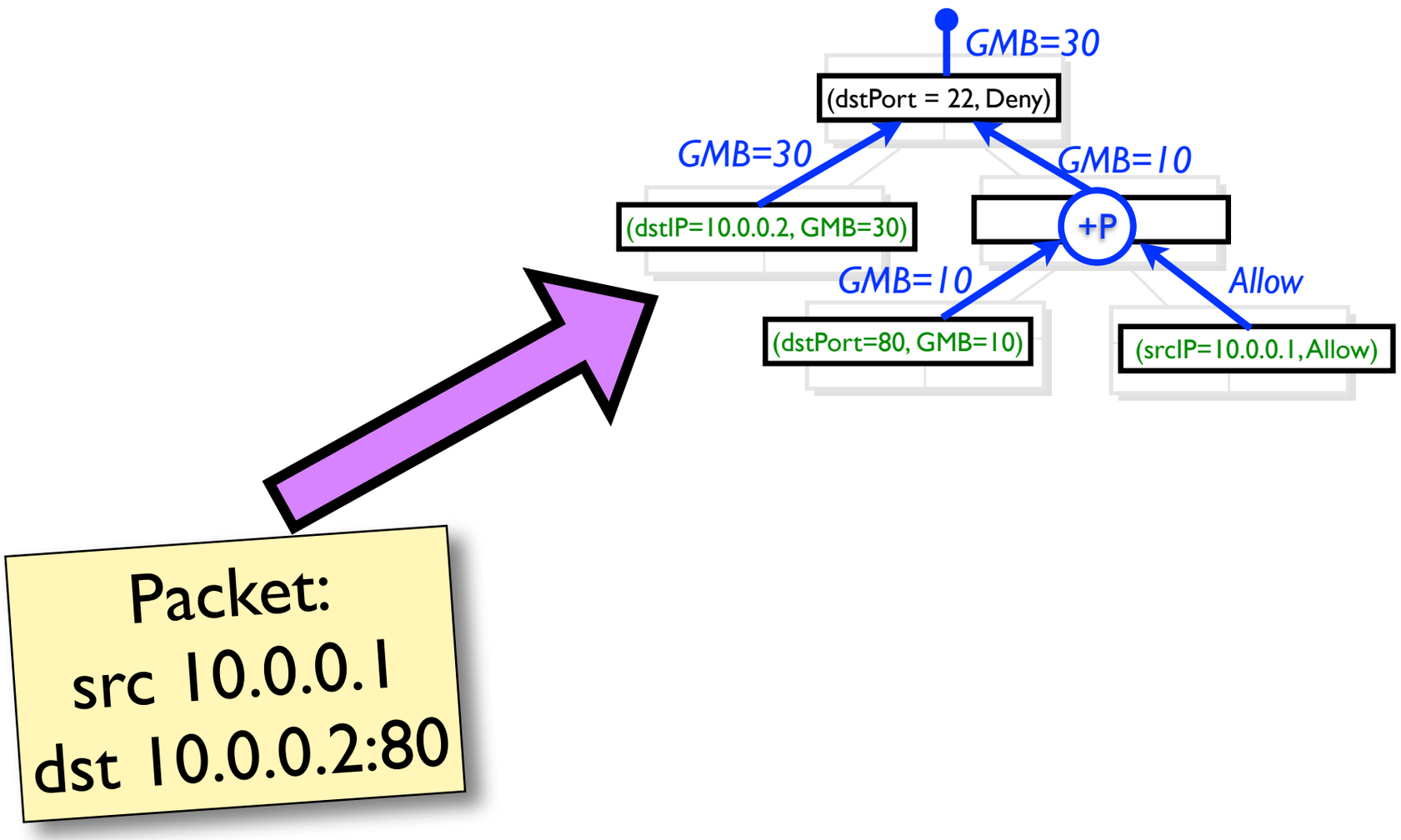
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

Coq Proof Assistant

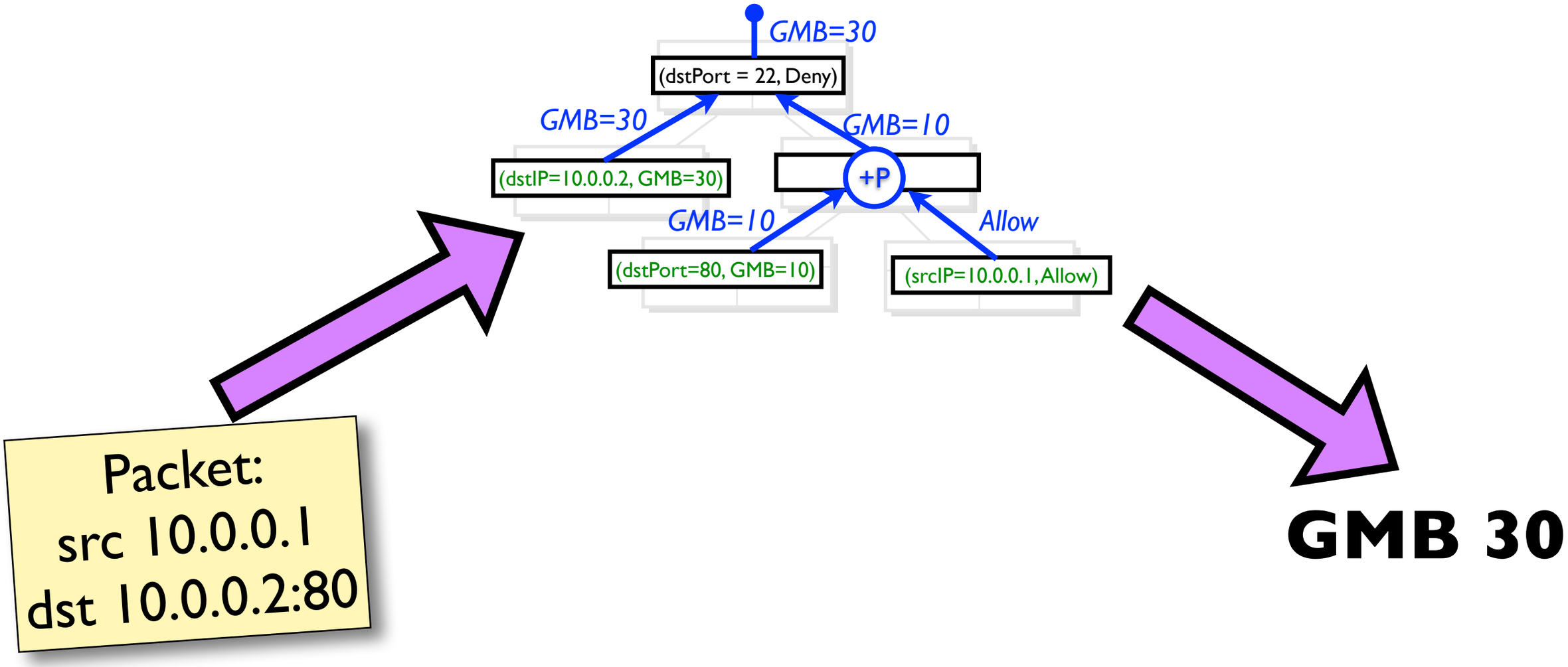


Packet:
src 10.0.0.1
dst 10.0.0.2:80

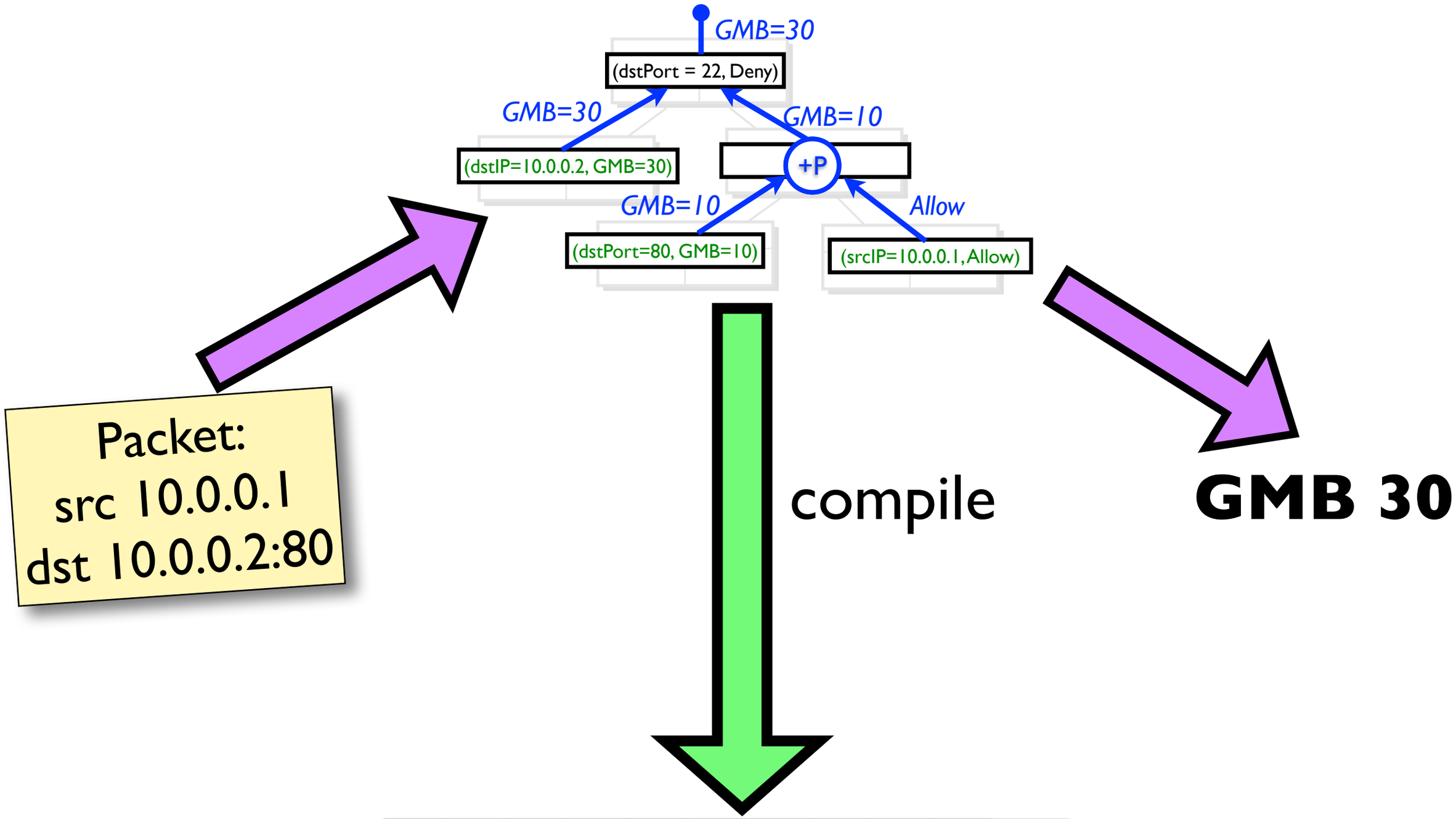
Theorem


GMB=30
(dstPort = 22, Deny)
GMB=30
GMB=10
(dstIP=10.0.0.2, GMB=30)
+P
GMB=10
Allow
(dstPort=80, GMB=10)
(srcIP=10.0.0.1,Allow)
Packet:
src 10.0.0.1
dst 10.0.0.2:80

Theorem

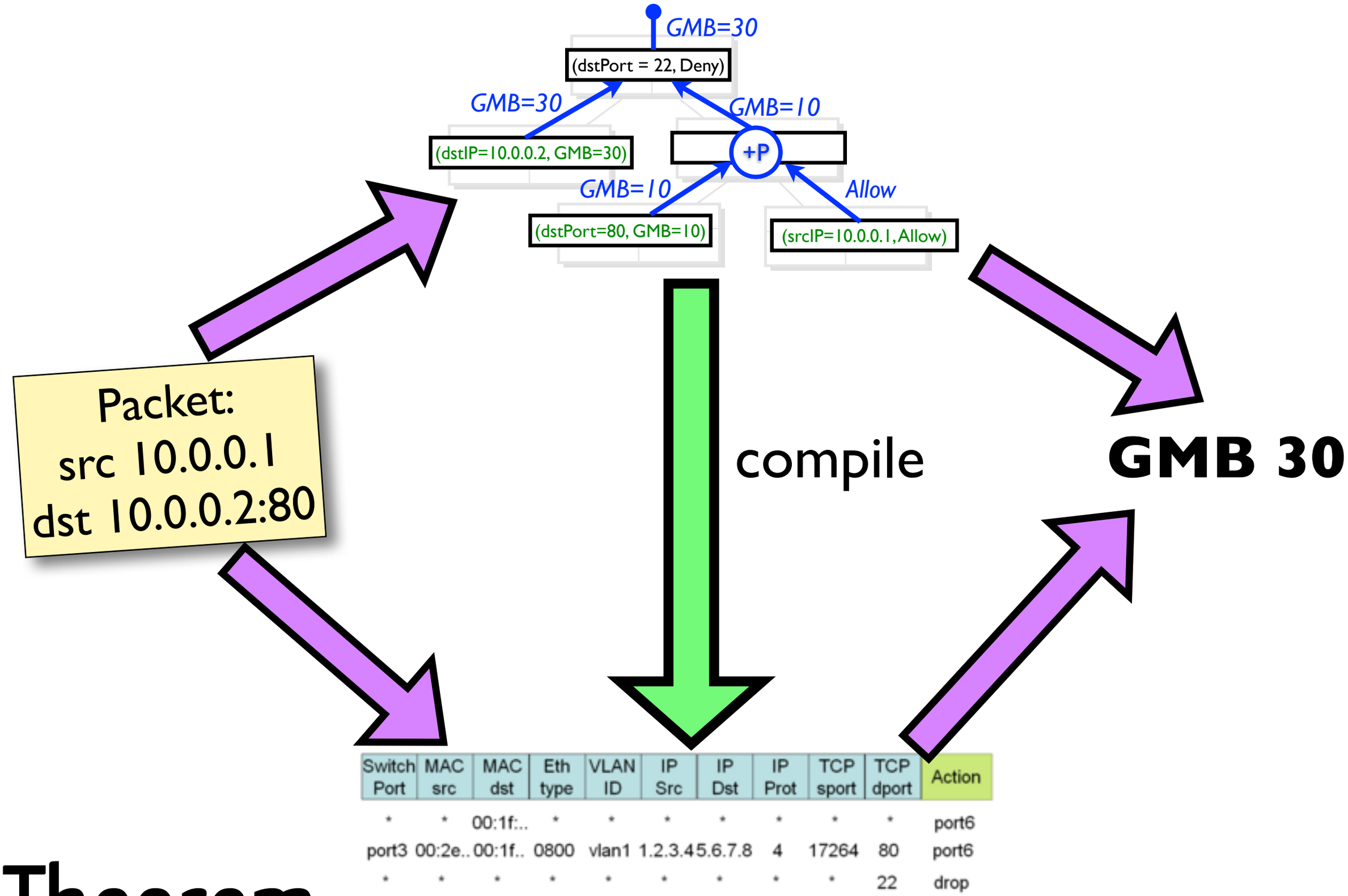

GMB=30
(dstPort = 22, Deny)
GMB=30
GMB=10
(dstIP=10.0.0.2, GMB=30)
+P
GMB=10
Allow
(dstPort=80, GMB=10)
(srcIP=10.0.0.1, Allow)
Packet:
src 10.0.0.1
dst 10.0.0.2:80
GMB 30

Theorem



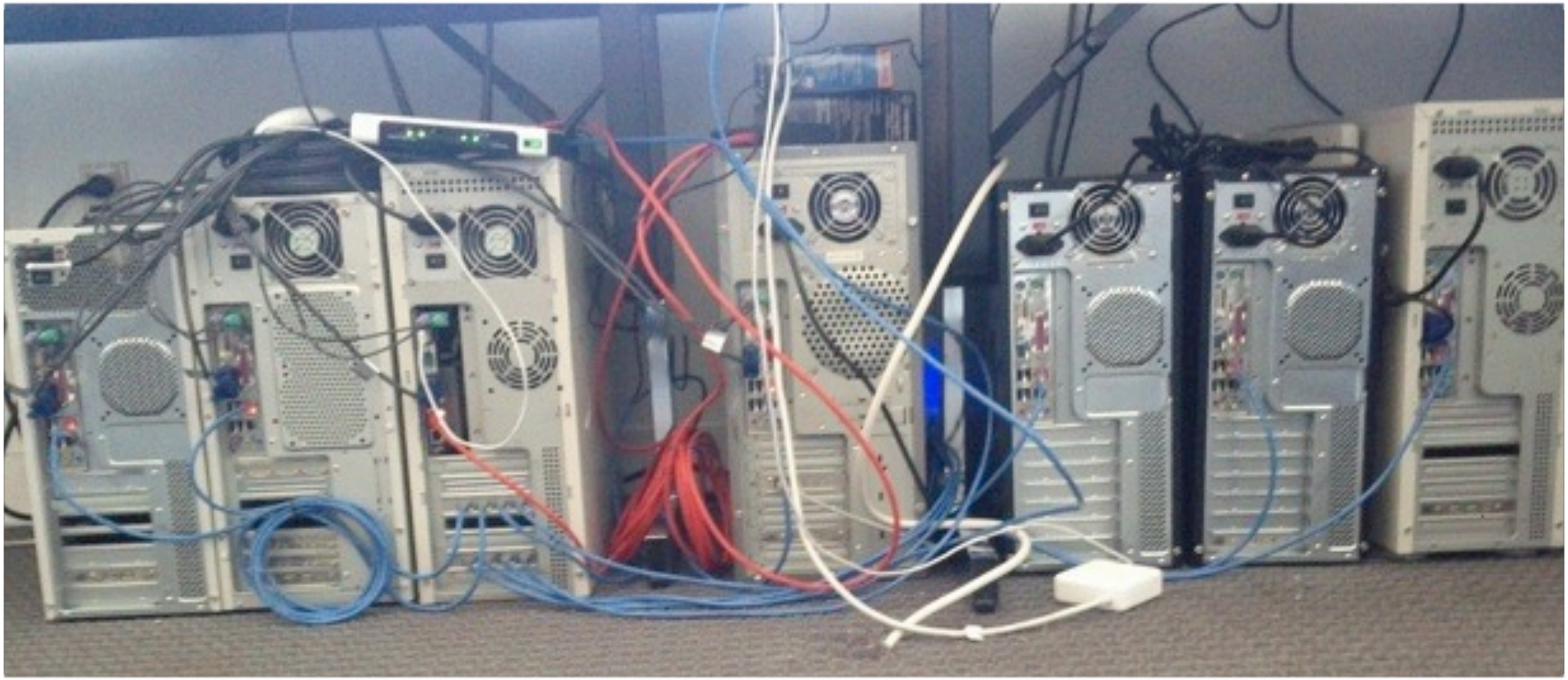
Switch Port	MAC src	MAC dst	Eth type	VLAN ID	IP Src	IP Dst	IP Prot	TCP sport	TCP dport	Action
*	*	00:1f:..	*	*	*	*	*	*	*	port6
port3	00:2e..	00:1f..	0800	vlan1	1.2.3.4	5.6.7.8	4	17264	80	port6
*	*	*	*	*	*	*	*	*	22	drop

Theorem


GMB=30
(dstPort = 22, Deny)
GMB=30
GMB=10
(dstIP=10.0.0.2, GMB=30)
+P
GMB=10
Allow
(dstPort=80, GMB=10)
(srcIP=10.0.0.1, Allow)
Packet:
src 10.0.0.1
dst 10.0.0.2:80
compile
GMB 30
Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Action
* * 00:1f:.. * * * * * * * port6
port3 00:2e.. 00:1f.. 0800 vlan1 1.2.3.4 5.6.7.8 4 17264 80 port6
* * * * * * * * * 22 drop

Theorem

Current Status



1. working controller

Current Status

1. working controller
2. client libraries

Current Status

1. working controller
2. client libraries
3. pane.cs.brown.edu

Current Status

1. working controller
2. client libraries
3. pane.cs.brown.edu
4. github.com/brownsys/pane

Current Status

Questions?

Andrew Ferguson
adf@cs.brown.edu

Co-authors

- Arjun Guha
- Chen Liang
- Rodrigo Fonseca
- Shriram Krishnamurthi




Questions?

Andrew Ferguson
adf@cs.brown.edu

Backup Slides