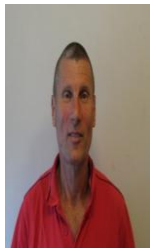
On the Complexity of Verifying Stateful Networks

A. Panda S. Shenker Y. Velner K. Alpernas A. Rabinovich M. Sagiv S. Shoham






Milestones

- [91] Logic programming for static analysis
- [95] nterprocedural Analysis
  - Context free reachability
  - Susan Horwitz & Tom Reps
- [03]CSSV: Proving the absence of buffer overrun
  - Dor, Rodeh, PLDI'03, Airbus
- [96-] Shape Analysis
  - Reasoning about heap reachability
  - TVLA

"Things like even software verification, this has been the Holy Grail of computer science for many decades but now in some very key areas, for example, driver verification we're building tools that can do actual proof about the software and how it works in order to guarantee the reliability." **Bill Gates, April 18, 2002. Keynote address at WinHec 2002**

Panaya Impact Analysis Tool

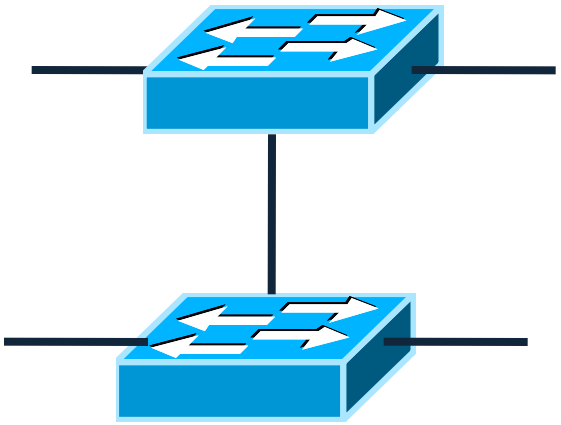
- Yossi Cohen and Nurit Dor
- Acquired by Infosys

The Internet: A Remarkable Story

- Tremendous success
  - From research experiment to global infrastructure
- Brilliance of under-specifying
  - Network: best-effort packet delivery
  - Hosts: arbitrary applications
- Enables innovation in applications
  - Web, P2P, VoIP, social networks, virtual worlds
- But, change is easy *only* at the edge… ☹

Inside the Net: A Different Story…

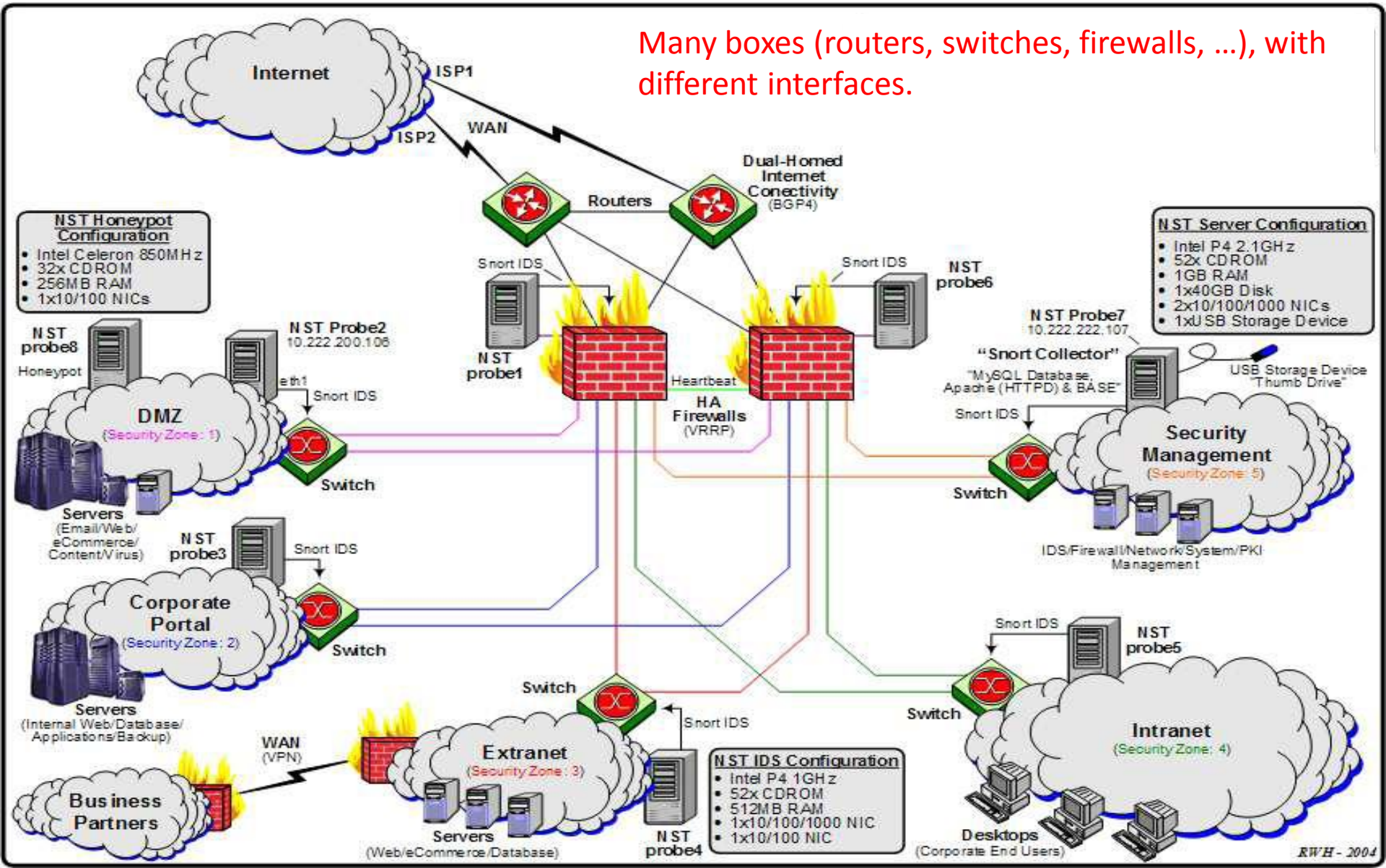
- Closed equipment
  - Software bundled with hardware
  - Vendor-specific interfaces
- Over specified
  - Slow protocol standardization
- Few people can innovate
  - Equipment vendors write the code
  - Long delays to introduce new features



Impacts performance, security, reliability, cost…

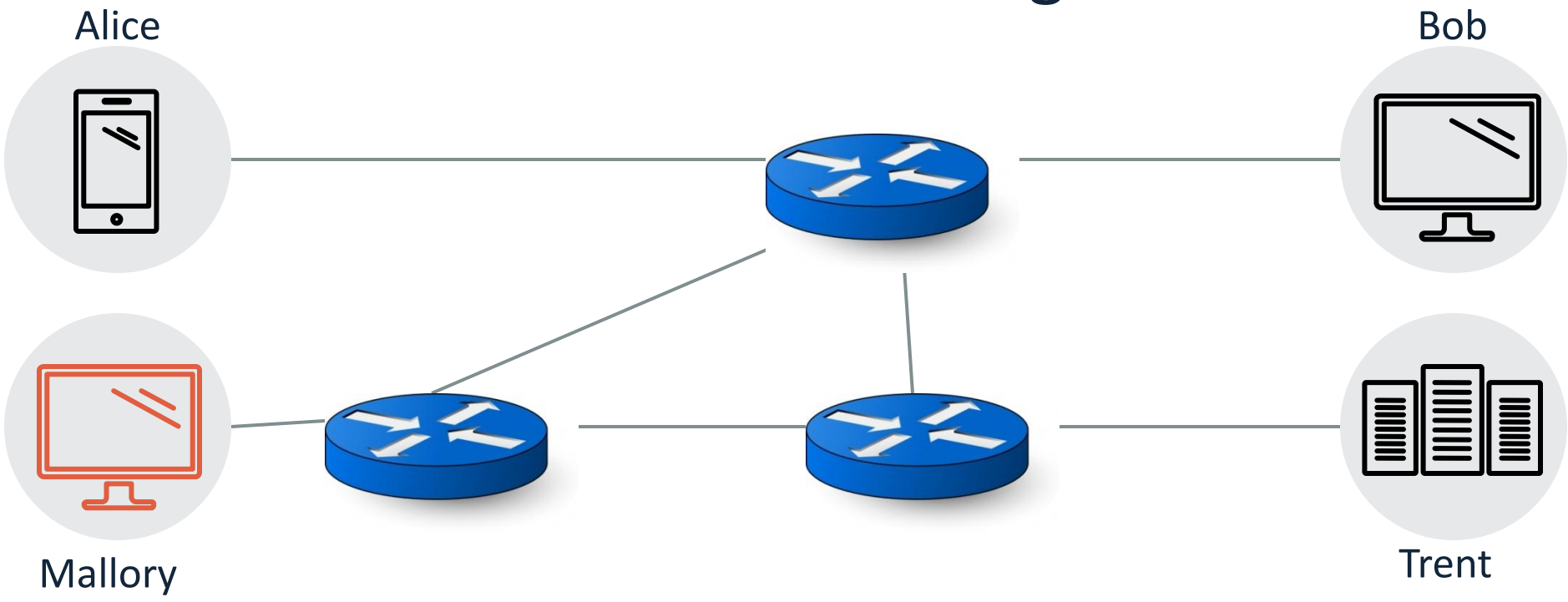
Do We Need Innovation Inside?

Many boxes (routers, switches, firewalls, …), with different interfaces.



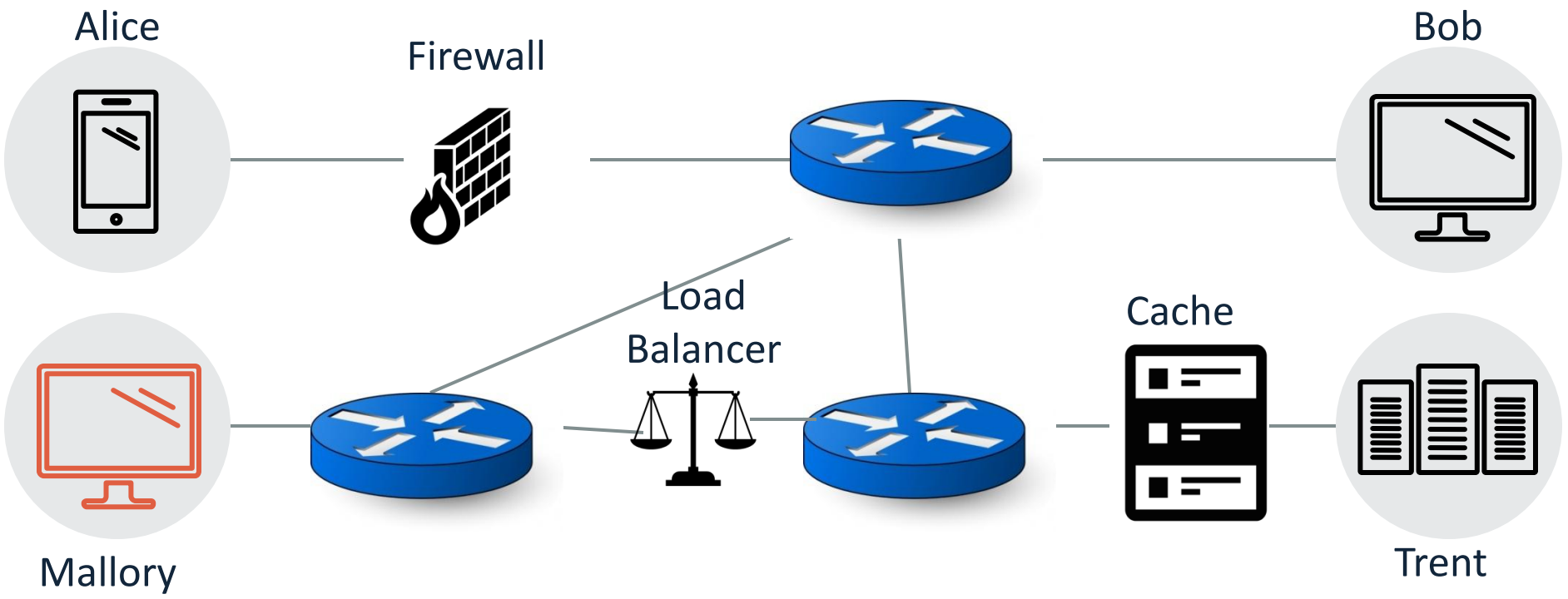
Classical Networking

Ted Stevens was right



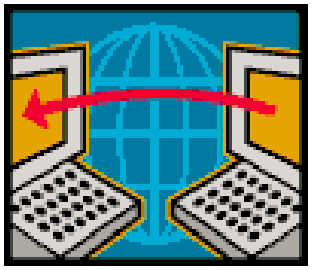
- Networks provide end-to-end connectivity
- Just contain host and switches
- All interesting processing at the hosts

Security & Performance



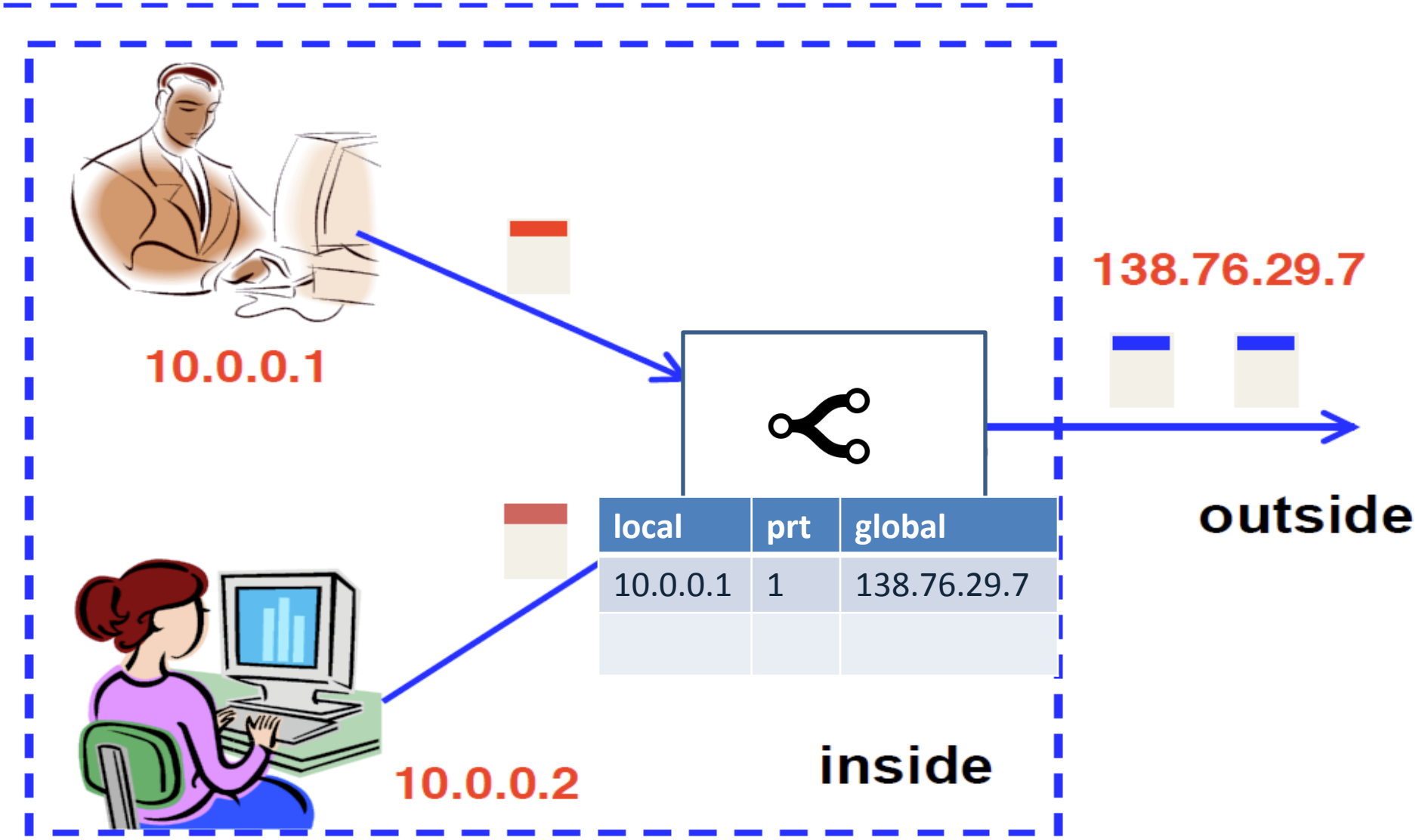
- Security (firewalls, IDSs,…)
- Performance (caches, load balancers,…)
- New functionality (proxies,…)

Middleboxes

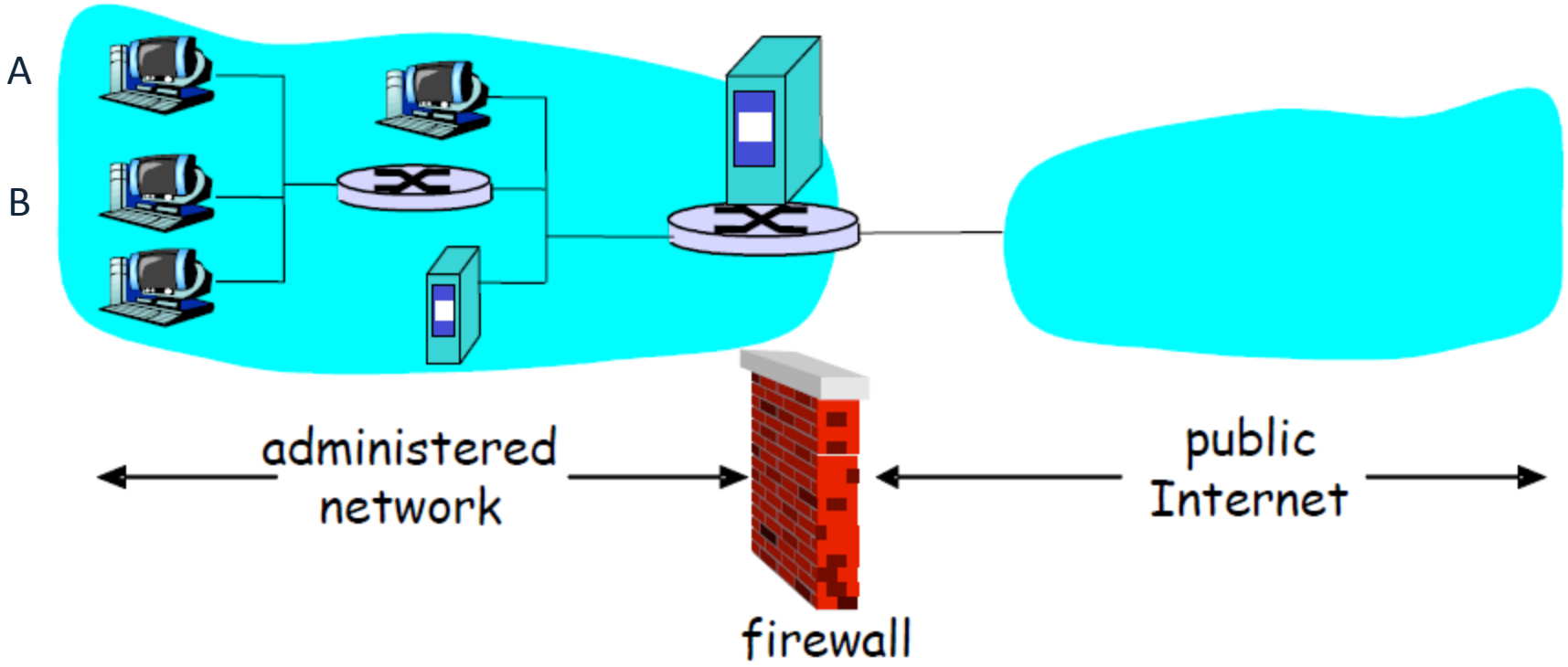


- Middleboxes are intermediaries
  - Interposed in-between the communicating hosts
  - Often without knowledge of one or both parties
- Examples
  - Network address translators (NAT)
  - Firewall
  - Traffic shapers
  - Intrusion detection systems (IDSs)
  - Transparent Web proxy caches
  - Application accelerators

NAT


10.0.0.1
138.76.29.7
local
prt
global
10.0.0.1
1
138.76.29.7
outside
10.0.0.2
inside

Firewalls

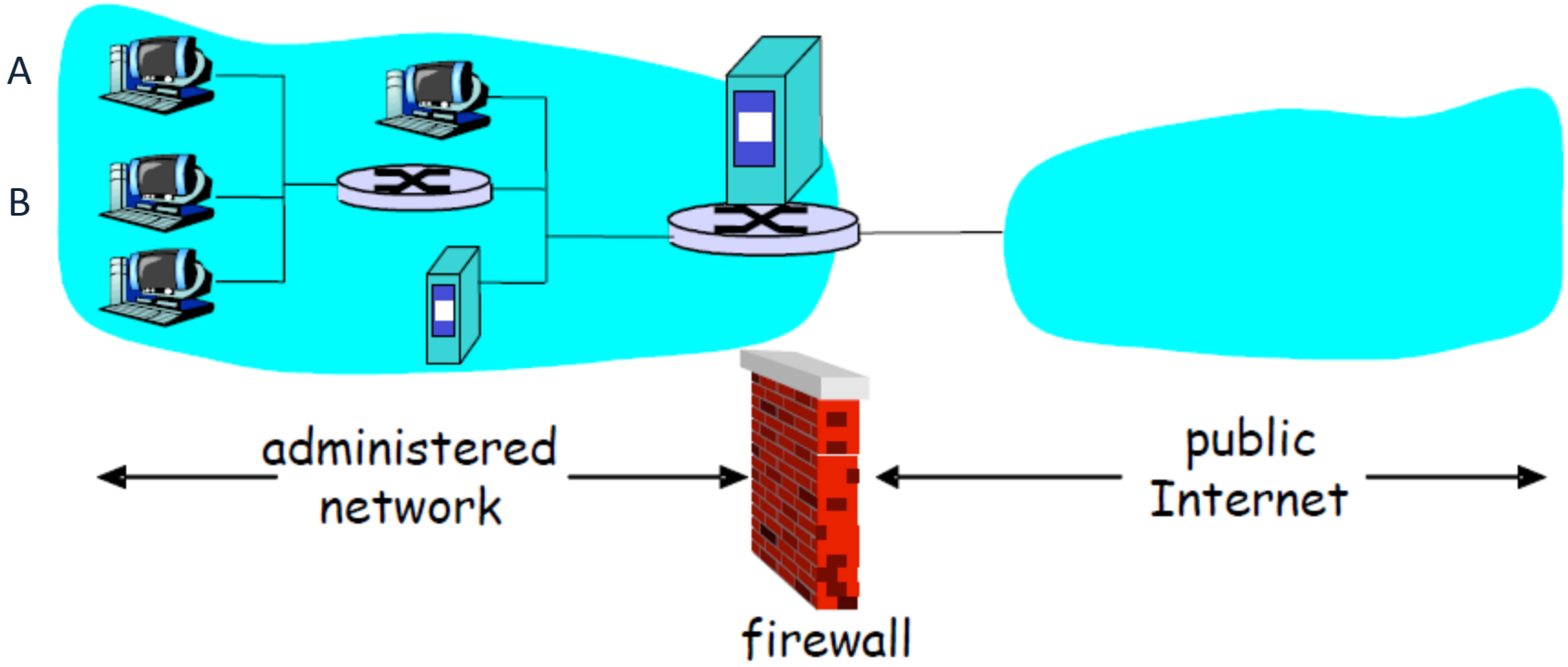


Trusted Hosts
H

A → H

H → A

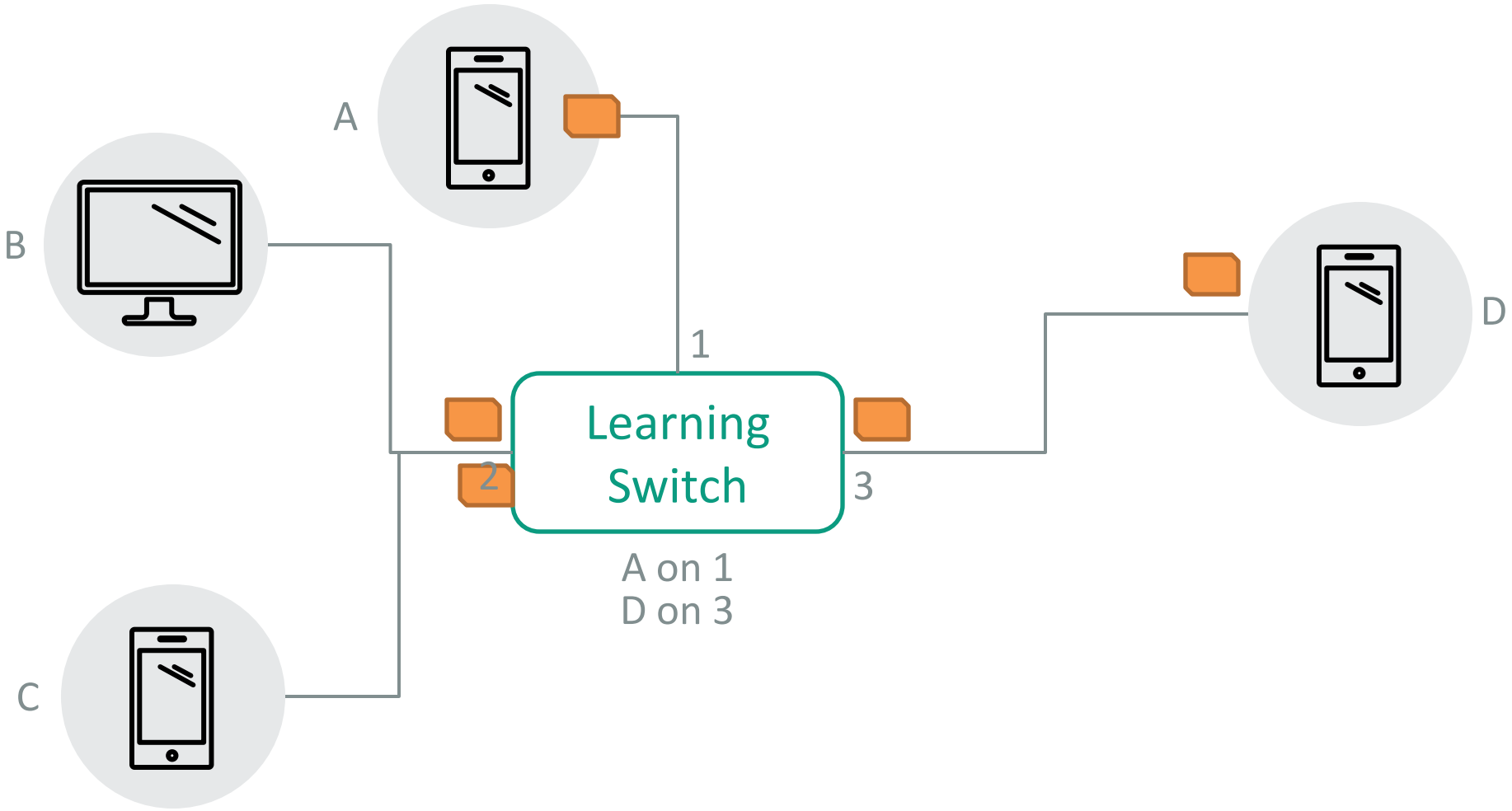
Firewalls



Trusted Hosts
H

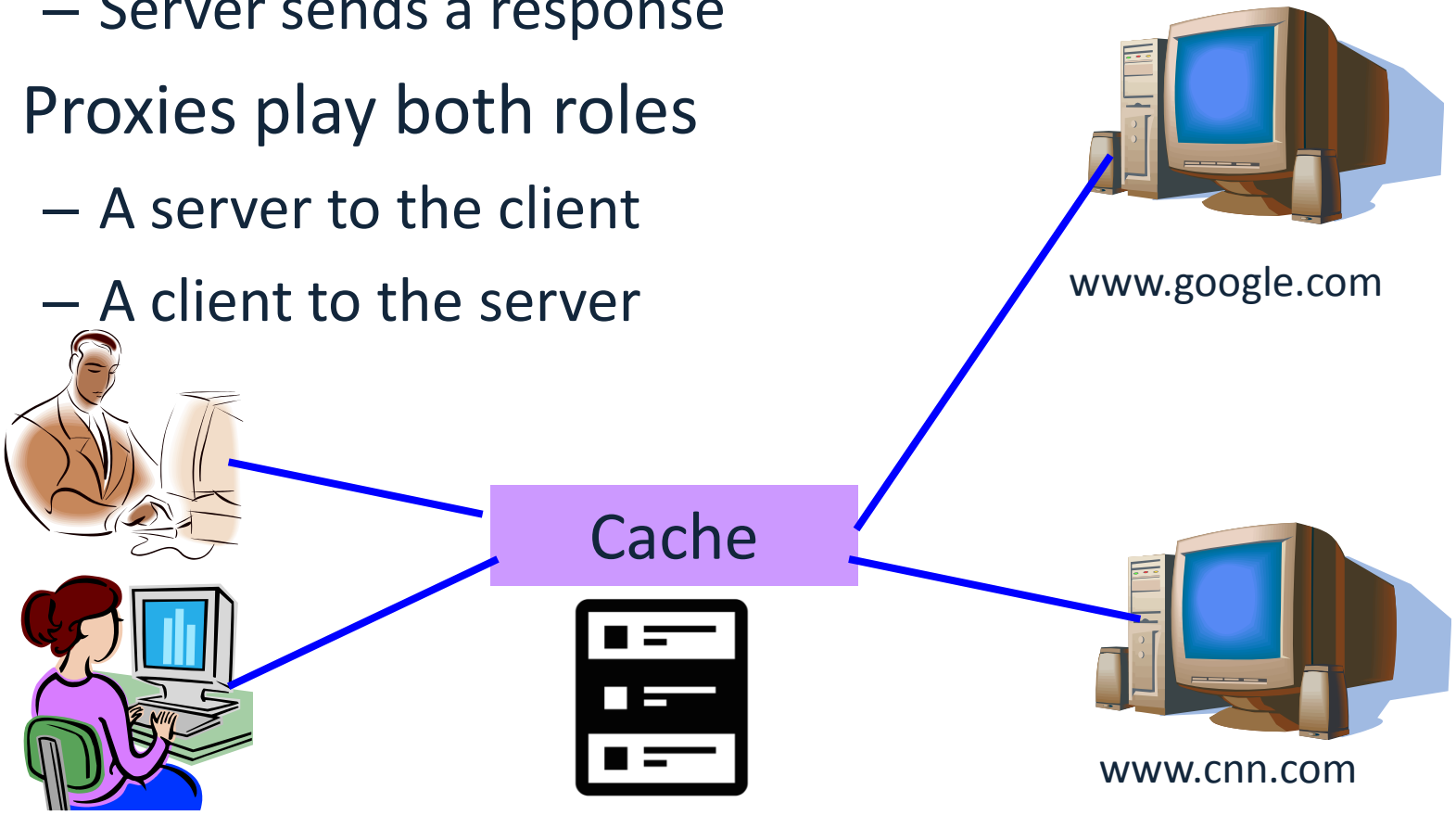
H→B

Learning Switch



Web Clients and Servers

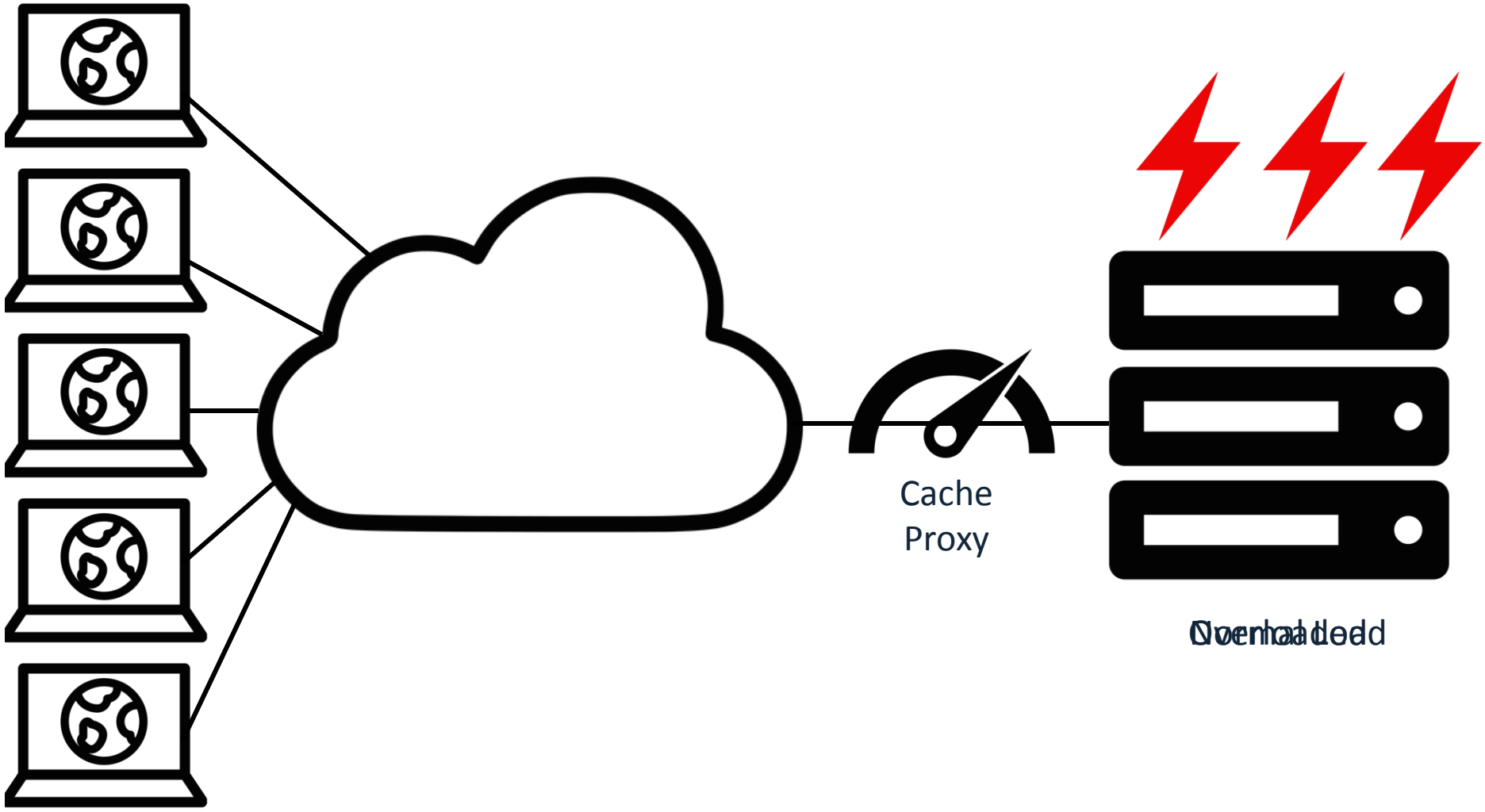
- Most Web applications use client-server protocol
  - Client sends a request
  - Server sends a response
- Proxies play both roles
  - A server to the client
  - A client to the server



Two Views of Middleboxes

- An abomination (toevah)
  - Violation of layering
  - Breaks the functional model
  - Responsible for many subtle bugs
- A practical necessity
  - Significant part of the network
  - Solving real and pressing problems
  - Needs that are not likely to go away
  - Local functionality enhancements

Local enhancements: Riverbed



Middlebox code can get complex

- Source code complexity
  - Bro Network Intrusion
    - 101,500 lines of C++, Python, Perl, Awk, Lex, Yacc
  - Snort IDS 220,000 C, …
  - Pfsense 476438 locs of C,php,scripts,…
- Hard to specify correctness
  - What is a correct IDS?

Middlebox code can get complex

- Source code complexity
  - Bro Network Intrusion
    - 101,500 lines of C++, Python, Perl, Awk, Lex, Yacc
  - Snort IDS 220,000 C, …
  - Pfsense 476438 locs of C,php,scripts,…
- Hard to specify correctness
  - What is a correct IDS?

Programming error

- The middlebox code fails to implement the required functionality
- Incorrect intrusion detection system
  - 10 CVE reports for pfsense in 2014, a popular firewall
  - CVE on Firewall hardware from Palo Alto Networks (2010)
    - Misinterprets HTTP cookie options, etc
- Heartbleed bug
  - allows anyone on the Internet to read the memory of the systems protected by the vulnerable versions of the OpenSSL software

- Requires code analysis

Hypothesis

- There are only few types of middleboxes
- Can abstract the model of middleboxes as finite state machines

Misconfiguration errors

- Do the topology and the middlebox configuration implement the specification?
- Responsible for 43% of network failures [IMC:RJ13]

[IMC:RJ13] R. Potharaju and N. Jain
Demystifying the dark side of the middle: field study of middlebox failures in datacenters
The Internet Measurement Conference, 2013

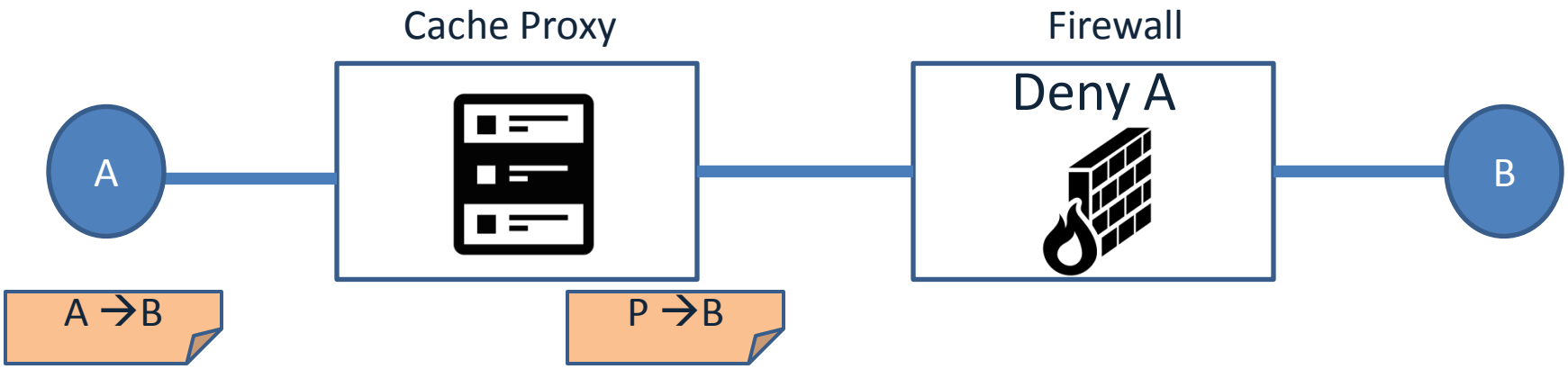
Safety of Computer Networks

- Show that something bad cannot happen
- Early detection of potential bugs
- Isolation:
  - A packet of type t sent from host A never reaches host B
    - Isolation between two universities
    - SSH packets from host A cannot reach B

Safety with middleboxes

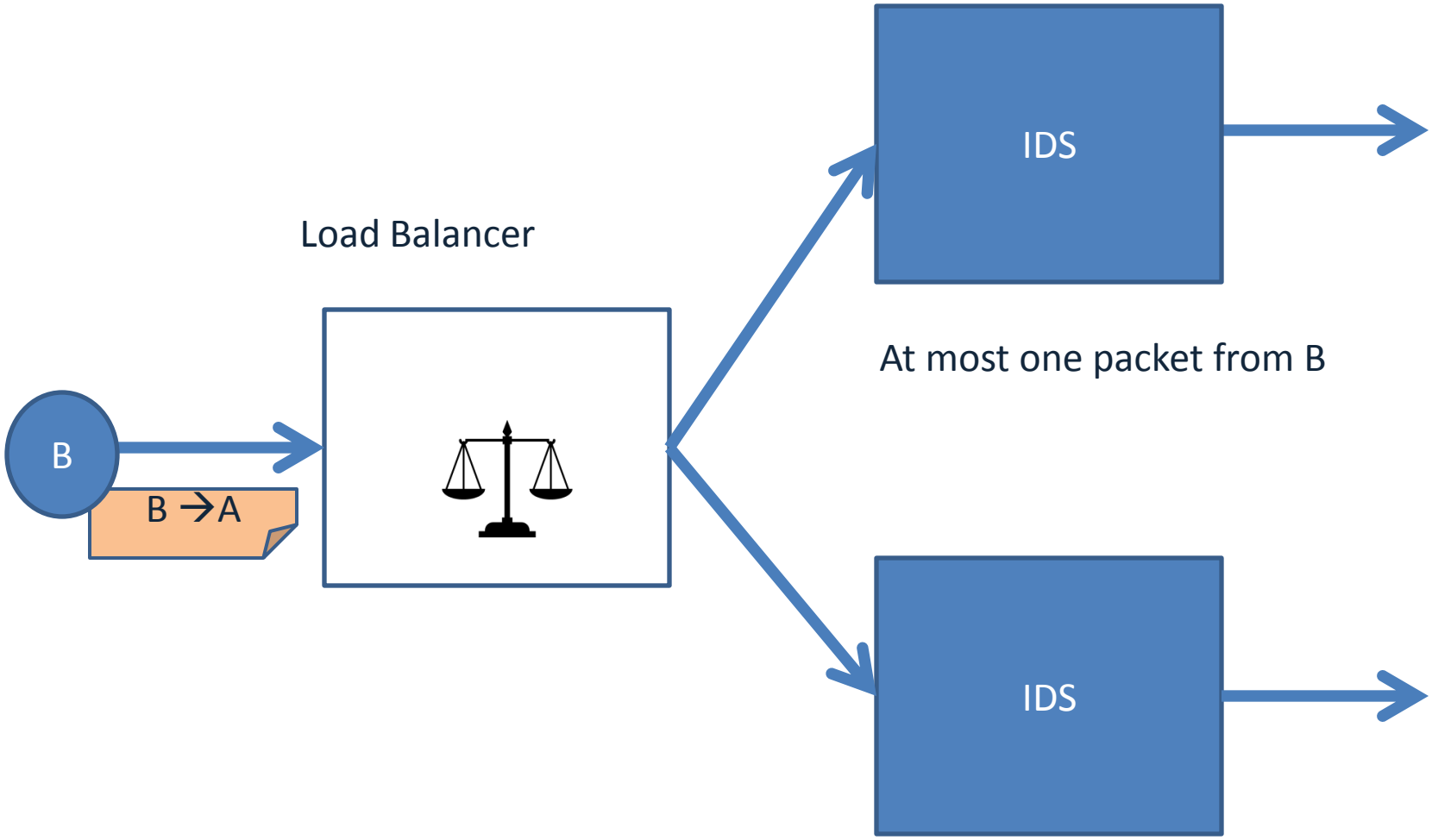
- Safety can be checked when the network only has switches with static routing rules
  - Trace the forwarding graph
- Middleboxes make everything harder
  - Arbitrary behavior – black box
  - Rewrite packet headers
  - Middlebox behave differently over time – need to reason about history
  - Composition may violate safety

Firewall Misconfiguration



A is isolated from B

Complex misconfiguration



Topology Assumptions

- Finite set of hosts H
- Fixed set of middleboxes M
  - Switches are degenerate middleboxes
- Fixed undirected topology
  $E \subseteq (H \times Pr \times M) \cup (M \times Pr \times Pr \times M)$

Packet Assumptions

- Finite set of packet types T
- Finite set of ports Pr per middlebox
- Finite set of packet headers
  $(t, src, dst, pr) \in P = T \times H \times H \times Pr$
- No bound on the number of packet sent
- Many packets may be sent before a safety violation occurs

Middlebox Abstract Semantics

- The abstract semantics of each middlebox is a function
  - m: $P^* \times P \rightarrow 2^P = P^* \rightarrow (P \rightarrow 2^P)$
  - Packet bodies are unchanged

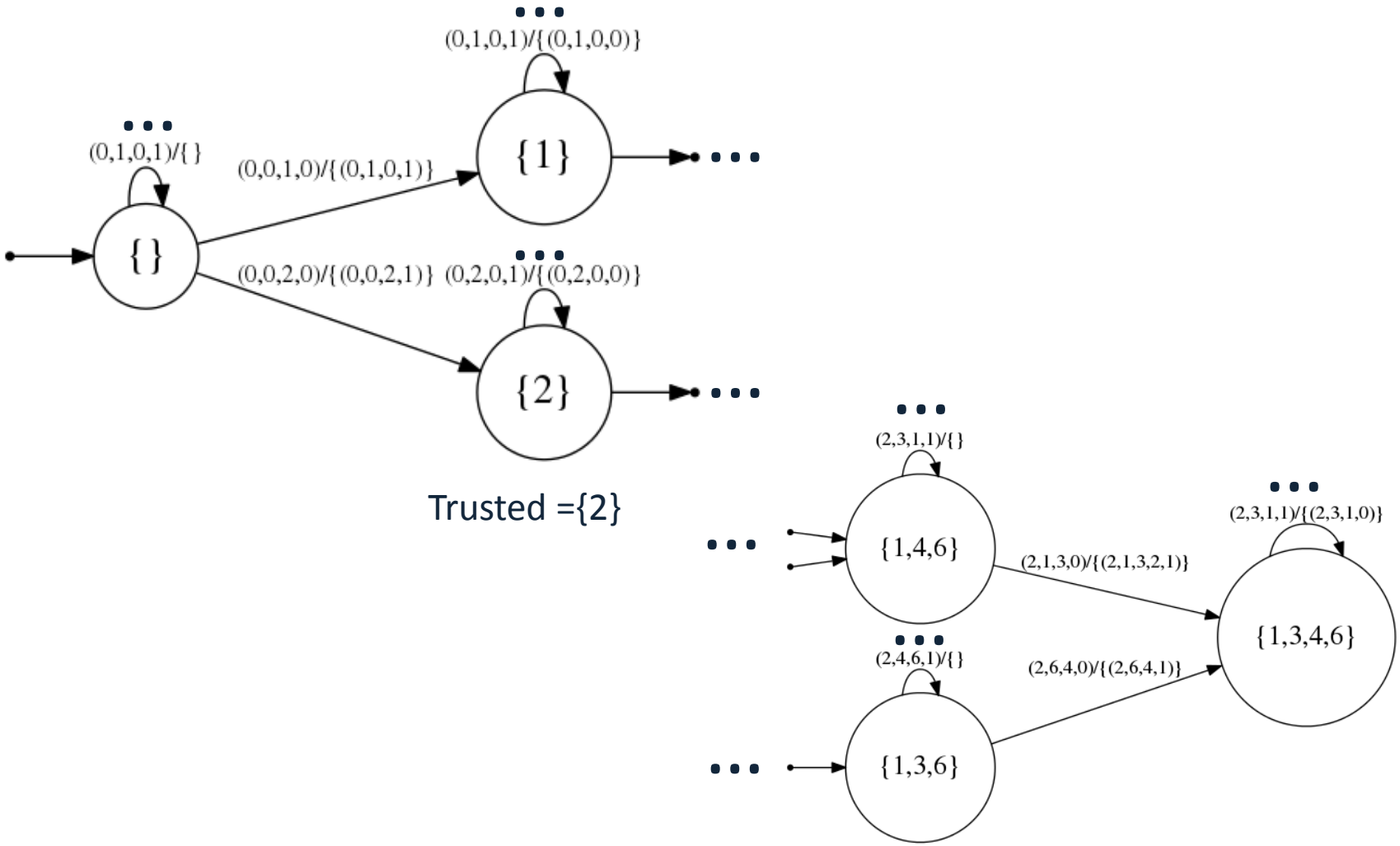
Common middleboxes

Middlebox	Function
Switch	$\lambda h, p = \{p[out \mapsto pr] \mid pr \in PR - p.ip\}$
Firewall	$\lambda h, p =$ if trusted(p, h) then $\{p[out \mapsto pr] \mid pr \in PR - p.ip\}$ // forward else {} // drop
Learning Switch	$\lambda h, p =$ if there exists $pr_0 \in Prt$ such that connected(p.dst, h, pr_0) then $\{p[out \mapsto pr_0]\}$ // forward else $\{p[out\} \mapsto pr : pr \in Prt, pr \neq p.ip\}$ // flood
IDS	$\lambda h, p =$ if trusted(p, h) then $\{p[out \mapsto pr] \mid pr \in PR - p.ip\}$ // forward else {} // drop
Cache Proxy	$\lambda h, p =$ if avail(p.body, h, response) then $\{p[src \mapsto me, dst \mapsto p.src, body \mapsto response]\}$ else $\{p[src \mapsto me]\}$

Modeling Middliboxes by FSMs

- A Transducer $\underline{m}$ =<S, s_0, P, f, δ> where
  - S are the states of the middleboxes
  - $s_0 \in S$ is the initial state
  - f: $S \times P \rightarrow 2^P$ is the current forwarding behavior
  - δ: $S \times P \rightarrow 2^S$ is the next state
  - Extend δ to histories
    - δ ([]) = {s_0}
    - δ (h . p) = δ (δ (h), p))
- $\underline{m}$ models m: $P^* \times P \rightarrow 2^P$ when for all h $\in P^*$ and P $\in$P:
  - $f(\delta(h), p) = m(h, p)$

Partial FSM for Firewall



Trusted ={2}

(Type, Source, Destination, Port)/{Forwarded Packets}

The Safety Problem

- Given a fixed topology of middleboxes
- A finite state transducer for each of the middleboxes
- Prove that there exists no scenario of packet transmissions leading to a bad state
- Identify such scenariors

Undecidability

- Checking safety properties such as isolation is undecidable even for finite state middleboxes
  - Cycles in the topology allows counting
  - Even in the absence of forwarding loops

Obtaining Decidability

- Show that if there is a scenario leading to a safety violation then there is also bounded one
- Reduction to a decision procedure

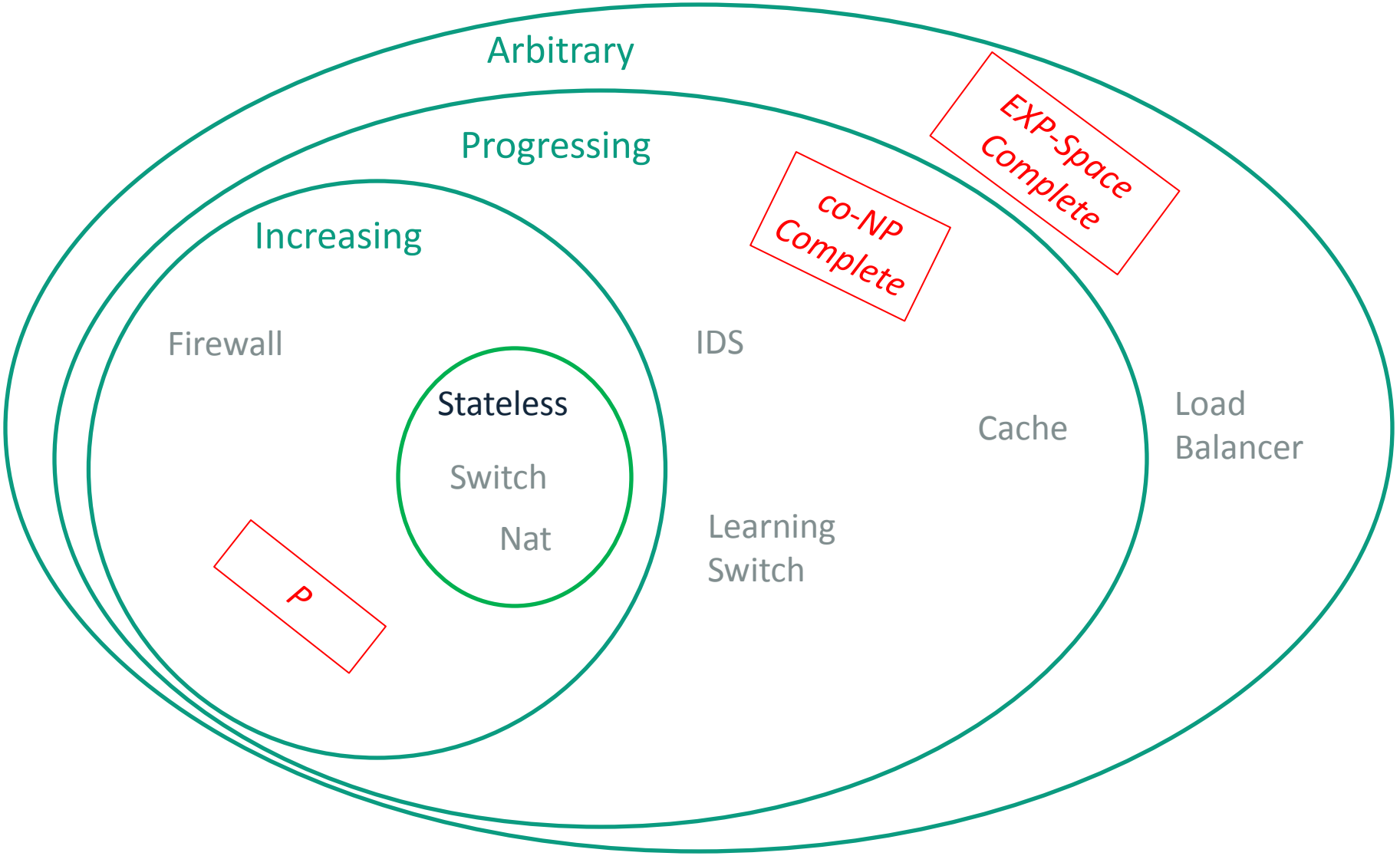
Non-Deterministic Packet Handling

- Assumes that order of packet processing is arbitrary
- It may be that a packet p arrives before q and yet the middlebox processes q first
- If a the network is safe under non-deterministic assumption it is also safe under FIFO assumption
- May lead to false alarms
  - Middlebox can impose orders based on acknowledgements

Decidability

- Under non-deterministic assumptions safety is decidable
- More packets per state means more forwarding options
  - Order is immaterial
  - Terminating backward reachabilty
- Well Quasi-Order on Packet Multisets
- Reduction to Coverability in Petri Net
  - But complexity is high
    - EXPSPACE-Complete

Middlebox classification



Stateless Middleboxs

- Behavior independent of the history
  - Can maintain configuration information
- For all h, h' $\in$ P*:
  - m(h) = m(h')
  - For all p $\in$ P: m(h, p) = m(h', p)
- Examples
  - Switches and Routers
  - ACL Firewall
  - Simple load-balancer

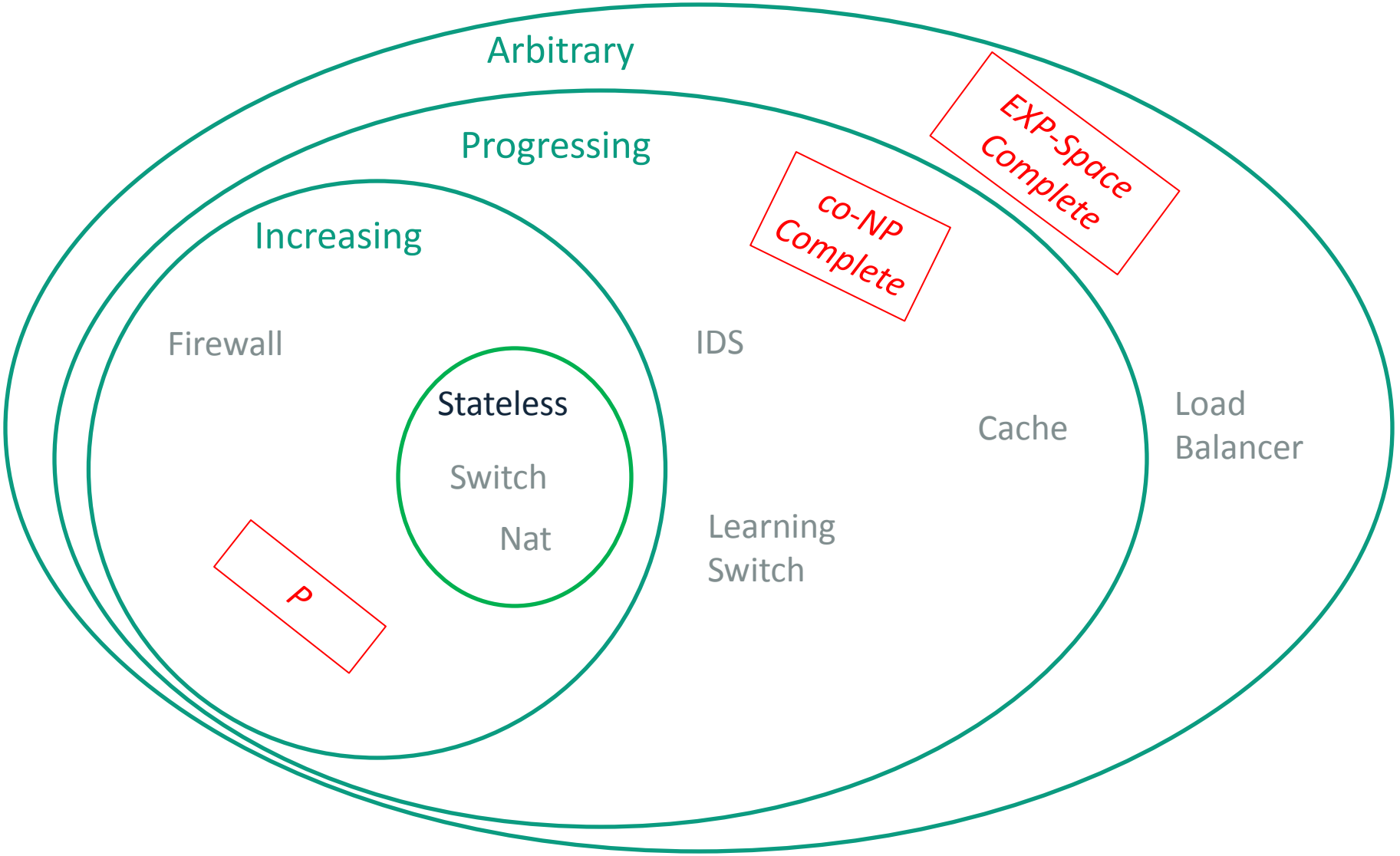
Increasing Middleboxs

- For every history, adding packets increase forwarding behavior
- For all h1, h2 $\in$ P* , p, p' $\in$ P:
  - m(h1:h2, p) $\subseteq$ m(h1:p':h2, p)
- Good examples
  - Stateless
  - Firewall
- Bad Examples
  - Learning Switch
  - Cache

Progressing Middleboxs

- No state reset
- No cycles in the automaton besides self-cycles
- Good examples
  - Learning switches
  - IDS?
  - Cache
- Bad Examples
  - Round-robin load balancer

Middlebox classification



Abstract **M**iddlebox **D**efinition **L**anguage

- Powerful enough to express the behavior of interesting middleboxes
- Succinct
  - Sometimes exponential state saving
- Simple enough for analysis
- Lends itself to classification of middleboxes
  - Same worst case complexity
  - But sometimes exponential saving

Firewall (AMDL)

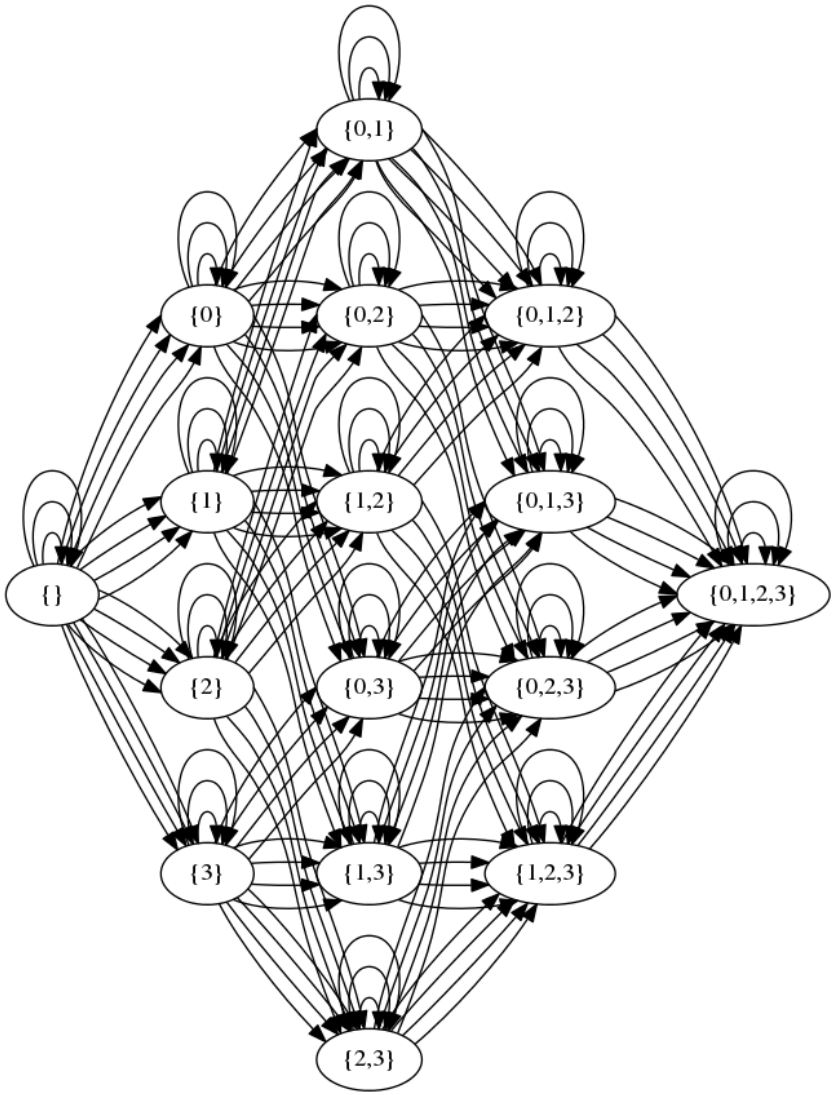
```
firewall(self) =
 receive(p, prt)⇒
  when prt = 1
     trusted_hosts.insert p.dst
     forward p to 2
   when prt = 2 and p.src ∈ trusted_hosts
     forward p to 1
```

Proxy (AMDL)

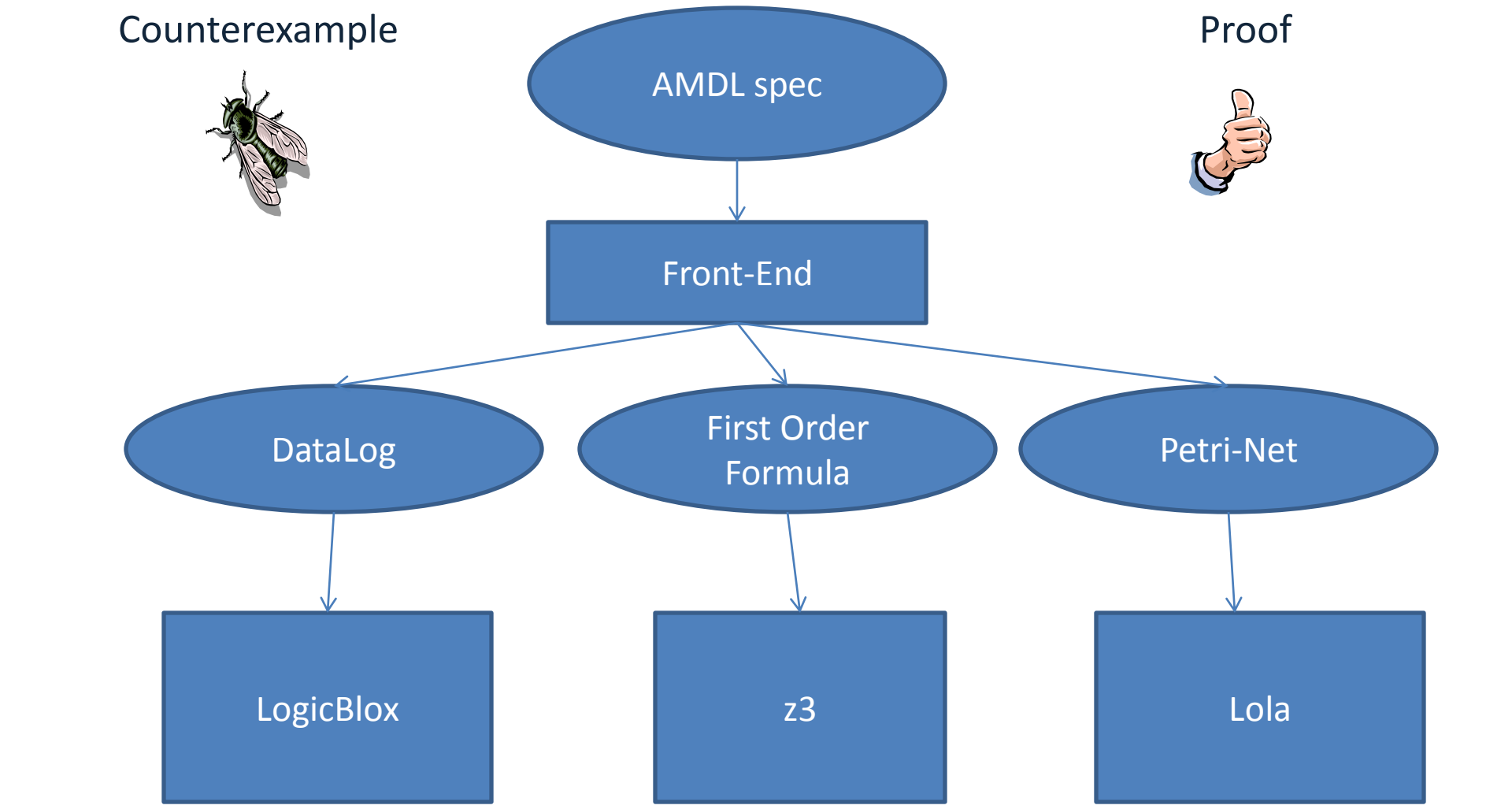
```
proxy(self) =
 receive(p, prt) ⇒
  when (p.type, response)∈ cache
     //stored response
     forward response[src=self.host] to prt
  when (p.type, p.src, p.dst,rport)∈requested
    // first response
    cache.insert (p.type, p);
    forward p[src = self.host] to port
  otherwise  // new message
   requested.insert (p.type, p.src, p.dst, prt);
   forward p[src = self.host] to oprt
     forall oprt ∈ AllPrt and oprt != pr
```

Firewall vs. FSM

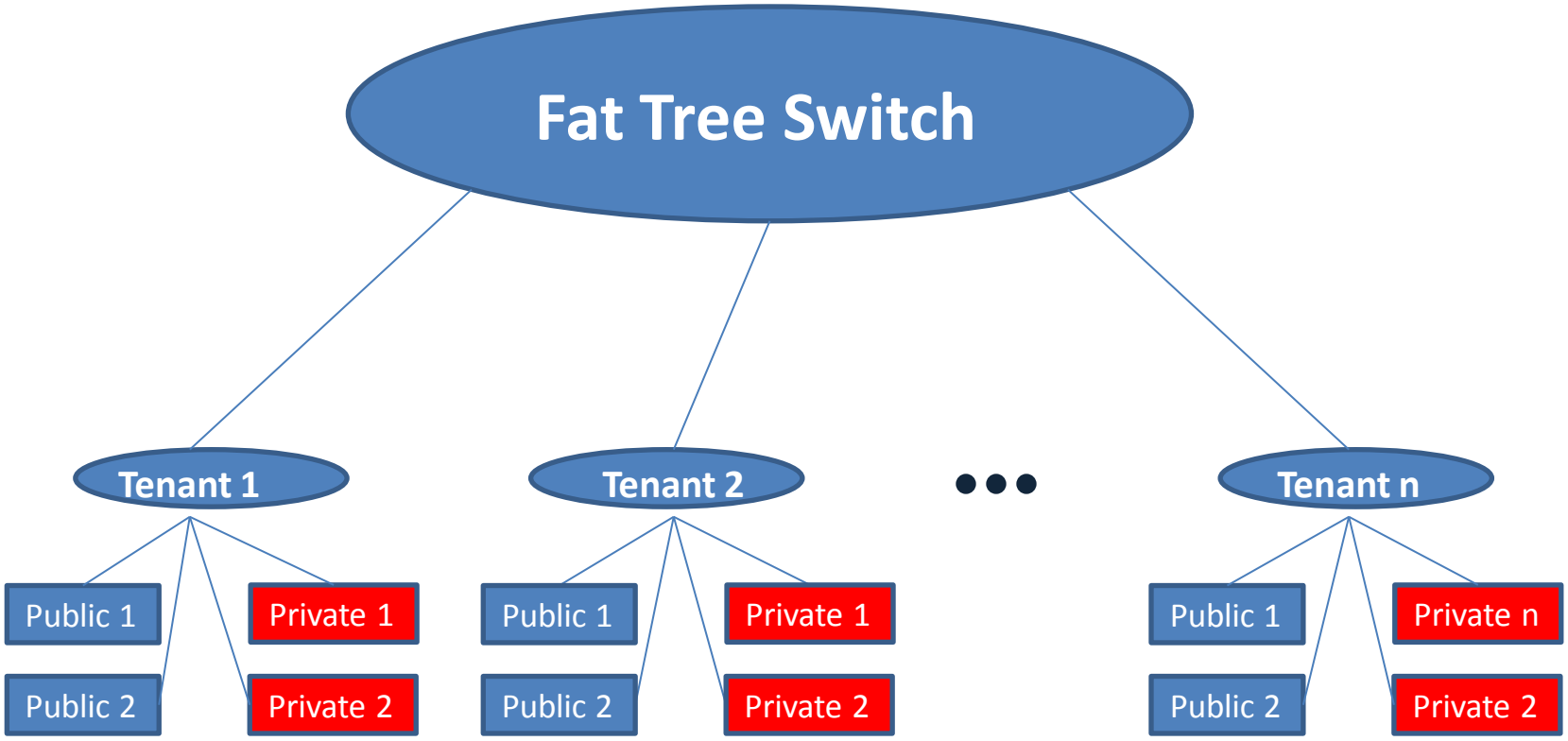
```
firewall(self) =
 receive(p, prt)⇒
  when prt = 1
     trusted_hosts.insert p.dst
     forward p to 2
  when prt=2 and
       p.src∈trusted_hosts
    forward p to 1
```



The MuteVer Toolset



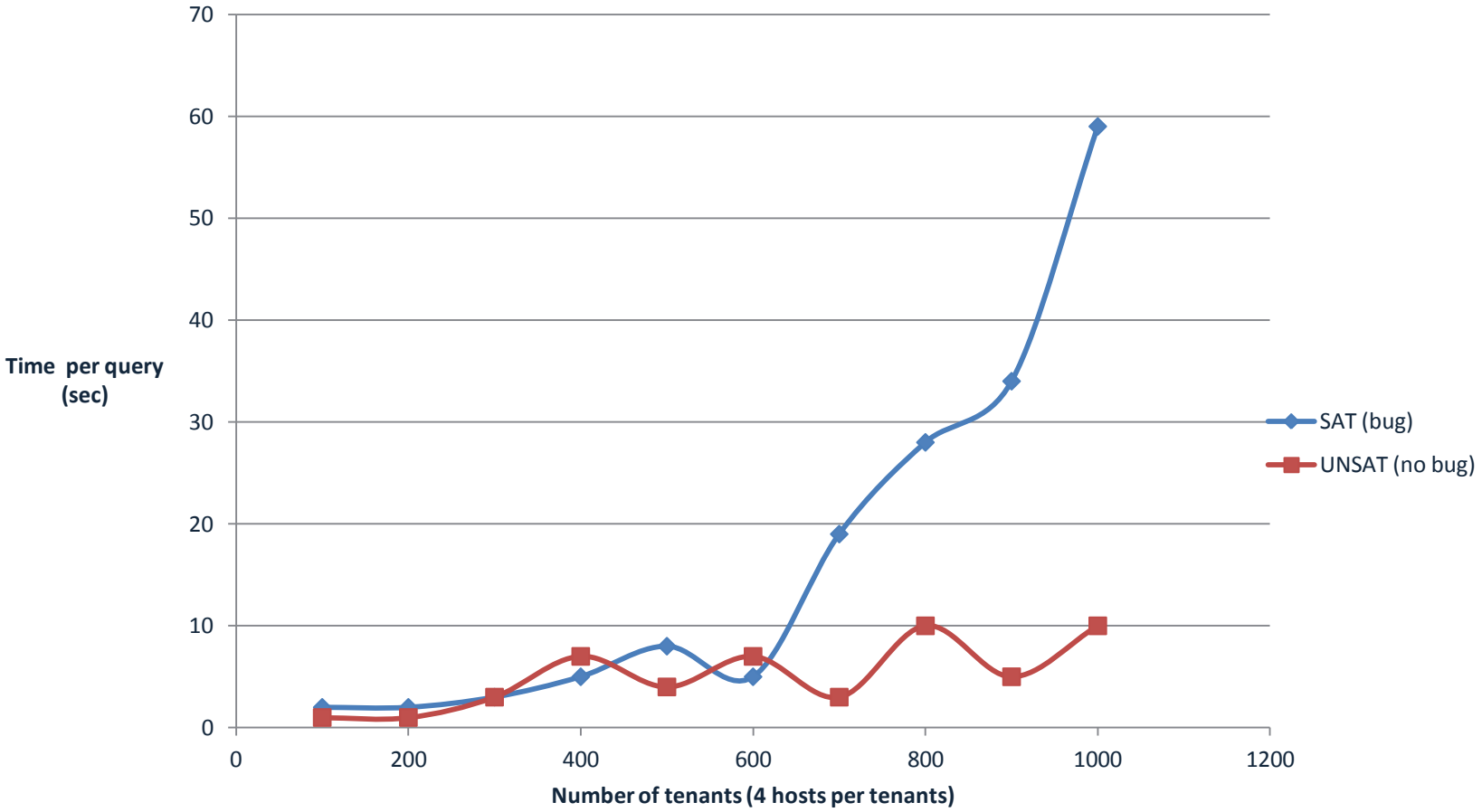
Amazon EC2 Security Groups model



Query

- Q1: can a packet arrive from tenant 7 to private host of faulty tenant, provided that the private host never sent a packet to tenant 7? (YES)
- Q2: can a packet arrive from tenant 7 to private host at tenant 2 (not faulty), provided that the private host never sent a packet to tenant 7? (NO)

Results (muZ)


70
60
50
40
30
20
10
0
Time per query (sec)
0
200
400
600
800
1000
1200
Number of tenants (4 hosts per tenants)
SAT (bug)
UNSAT (no bug)

(Some) Related Work

Dynamic

- Veriflow
  - Online verification
  - Handles dynamic networks pretty well
- Header Space Analysis
  - Offline and online verification

Static

- Firewall Verification
  - Margrave
- SDN
  - Netkat
  - Vericon
- Reductions to Datalog
  - Badfish
  - Checking Beliefs

Summary

- Middlebox classification
- Complexity results
- Initial toolset

Acknowledgments

- The Noun Project
- Nate Foster, Michael Freedman, and Jane Rexford