An Extensible Platform for Evaluating Security Protocols

Seny Kamara

joint with
L. Ballard, R. Caudy, D.Davis, F. Monrose

Outline

- Objectives
- High-level architecture
- Plugin architecture
- Case studies

Objectives

- Security
  - DDoS, VPNs, worm propagation, cryptographic protocols
- Ease of use
- Fast prototyping
- Research
- Education

Objectives

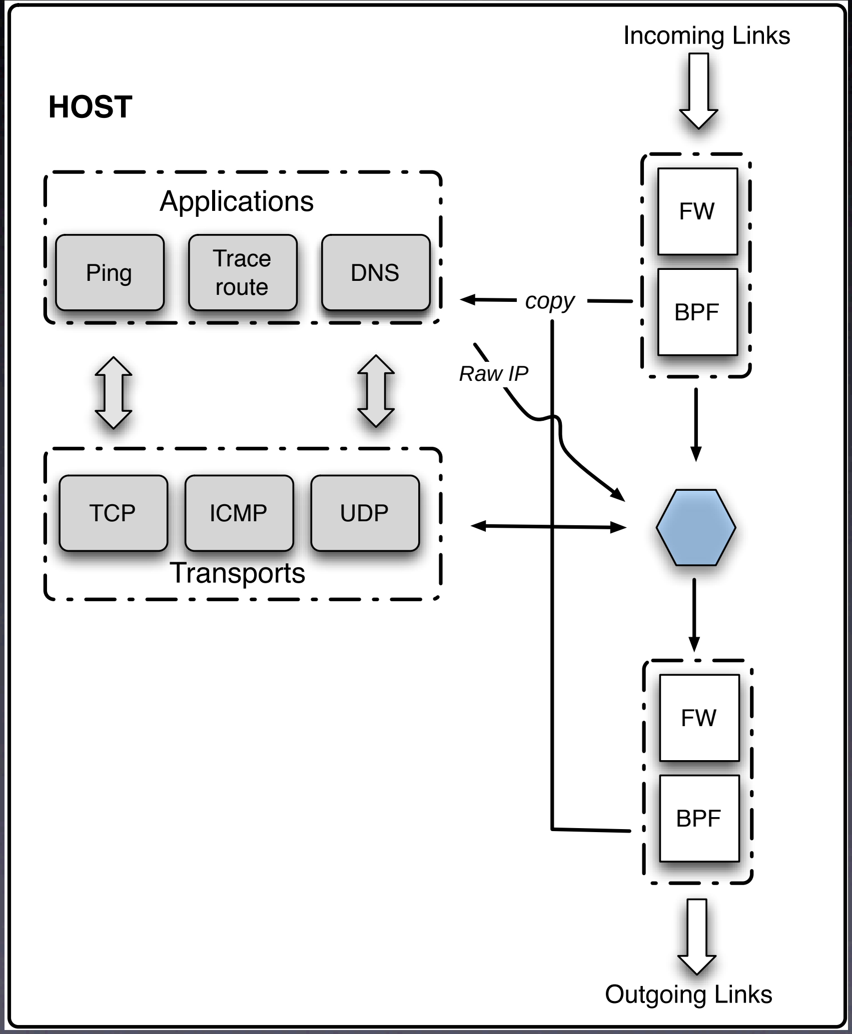
- Modularity
  - plugin architecture
- Portability
  - Java
  - Java networking API
- Dynamic customization
  - Java dynamic class loading

System Architecture

- Topology parser
  - Otter [CAIDA] file format (Brite)
  - Extended to handle real IPs
  - Routers can serve network prefixes
- User interface (interactive and scripts)
- Simulator (hosts, routers, links)

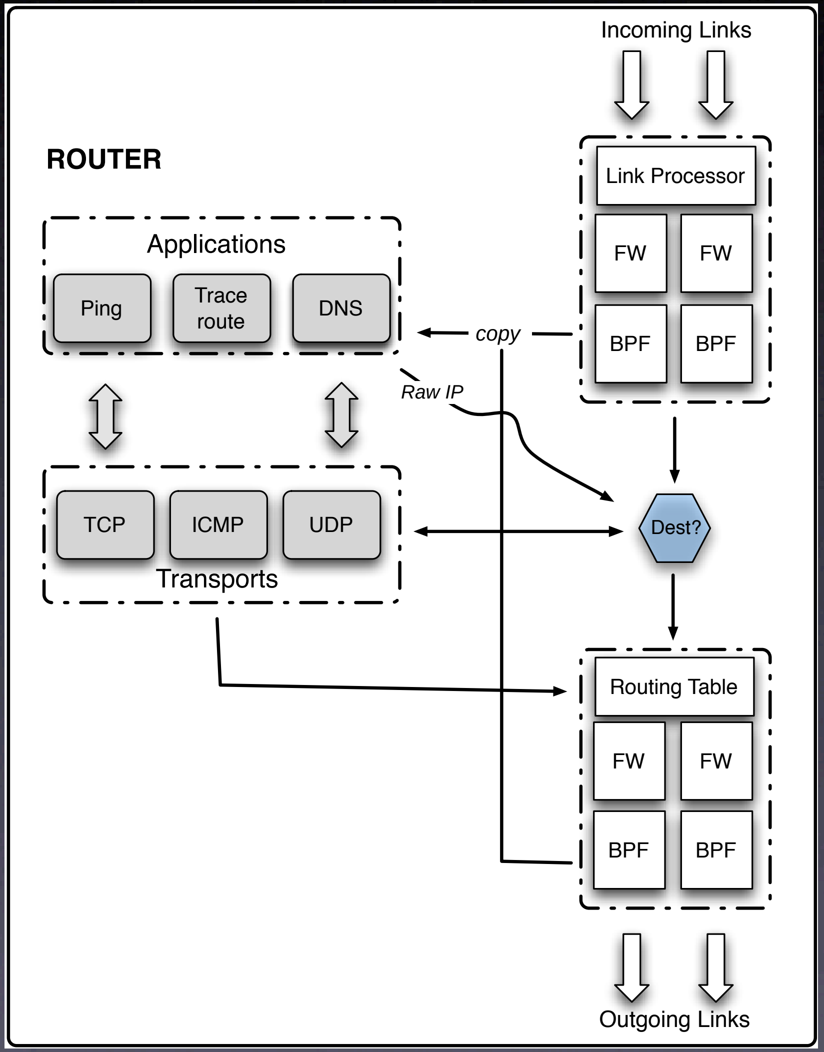
Host Architecture

- Incoming packet filter (FW)
- Incoming Berkeley Packet Filter (BPF)
- Transports
- Applications
- Transports
- Outgoing packet filter (FW)
- Outgoing Berkeley packet filter (BPF)



Router Architecture

- Link Processor
- Incoming packet filters (FW)
  - Incoming Berkeley Packet Filters (BPF)
- Transports
- Applications
- Transports
- Routing Table
- Outgoing packet filters (FW)
- Outgoing Berkeley packet filters (BPF)

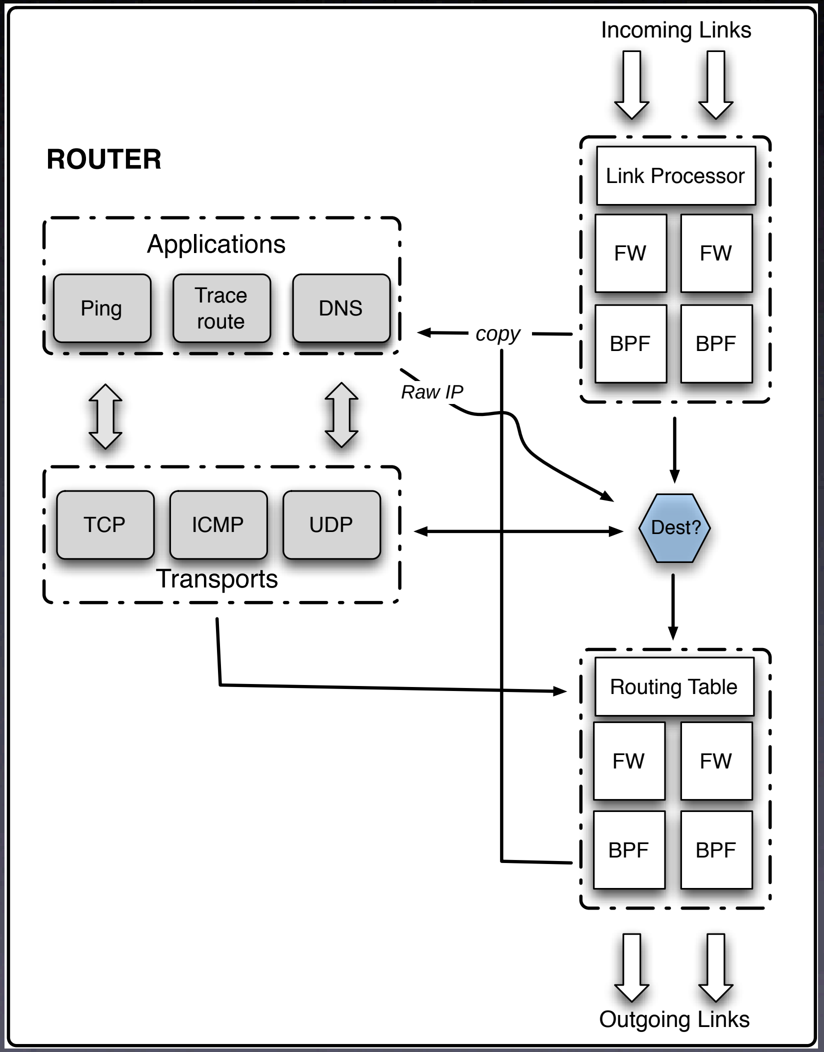


Plugin Architecture

- Modularity
- Transparency to user
- Dynamic Customization
  - Correctness and interoperability testing
  - Cryptographic protocols, TCP implementations, DDoS mitigation etc...

Plugin Architecture

- Transparent
  - *plugin [IP|all] ICMP*
  - *plugin [IP|all] Ping*
  - *select src-IP*
  - *ping dest-IP*
- Dynamic (Java's dynamic class loading)



Plugin Architecture

- Event notification (i.e. applications need to know if TCP stack is being replaced)
- Before plugin
  - Objects can register as listeners for particular plugins

Plugin Architecture

- Before plug out:
  - plugin's pre-plugout method is called and given replacing object
    - transfer state (i.e. firewall rules)
  - listeners are notified of plugout operation

Plugin Architecture

- Simnet plugins:
  - Topology parser
  - User interface
  - Hosts
  - Routers
  - Link processor

Plugin Architecture

- Simnet plugins:
  - Packet filters
  - Berkeley Packet Filters (BPFs)
  - Routing tables
  - Transports
  - Applications

Case Studies

- *Scalability*: Worm Propagation
- *Modularity*: DNSSEC

Experimental Setup

- Dual-processor 1.3 GHz XServe G4
- 1024 MB RAM
- Mac OS 10.2.6

Worm Propagation

- Zero-day worms
  - Nimda, Code Red I, Code Red II
- Compare effectiveness of various worm target selection algorithms

Worm Propagation

- Naive worms
  - Uniform selection
- Nimda
  - Biased towards own class B
- Code Red II
  - Biased towards own class A

Worm Propagation

- Requires
  - Topologies on the order of millions
  - Simnet only supports topologies on the order of hundreds (full packet-level simulation)
  - Trade simulation detail for scalability

Worm Propagation

- Aggregate Router plugin
- Simulate entire Class B networks
- Parameters:
  - percentage of reachable class C nets.
  - percentage of allocated IPs (in each class C)

Worm Propagation

- Worm Modeler Plugin
- Simulates propagation characteristics
- Parameters:
  - percentage of reachable hosts that are vulnerable
  - probing rate per infected host per second
  - target selection probs. for Class B, A, I

Worm Propagation

- Given scope of simulation we want to reduce total simulation time
- “Compress” time by only sending probes to vulnerable hosts
- And assigning a time cost to each probe according to a geometric distribution on the probability of choosing a vulnerable host

Worm Propagation

- 192 Agg. Routers chosen from AS level topology from Router Views project
- Yields about 2 million hosts

Worm Propagation

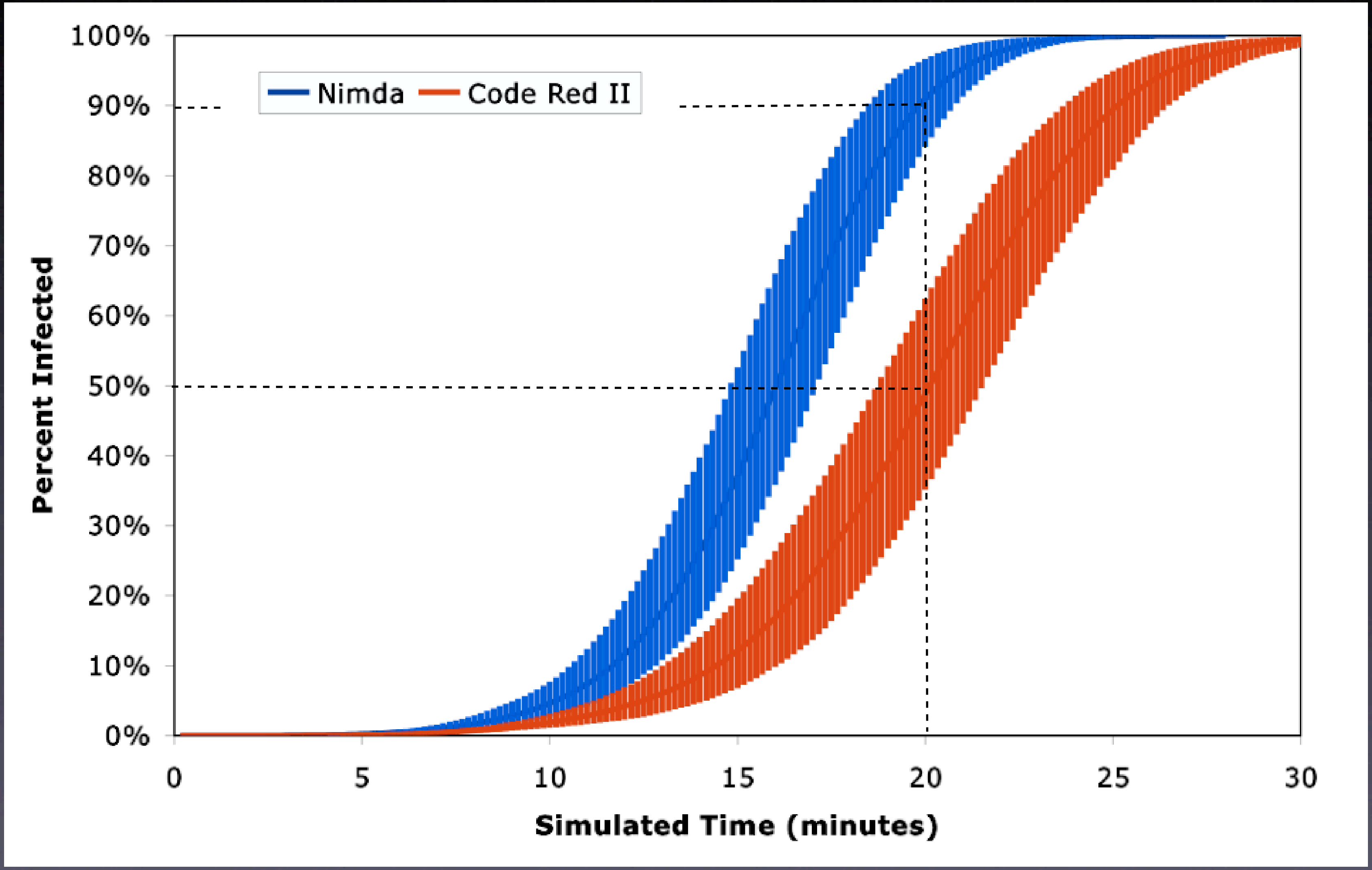
- 500,000 vulnerable hosts
- 0.5 probes per infected host per second
- Target selection:

	B	A	I
Naive	0.3	0.3	0.3
Nimda	0.5	0.25	0.25
Code Red II	0.375	0.5	0.125

Worm Propagation

- Assumptions
  - Vulnerable hosts infected after 1 UDP probe (SQLSlammer)
  - Once infected host remains infected

Worm Propagation



Pushback

- Aggregate-based Congestion Control (ACC) [MBF+01]
  - DDoS mitigation
  - Rate limits flows that match certain characteristics
  - If necessary propagates rate limiting upstream

Pushback

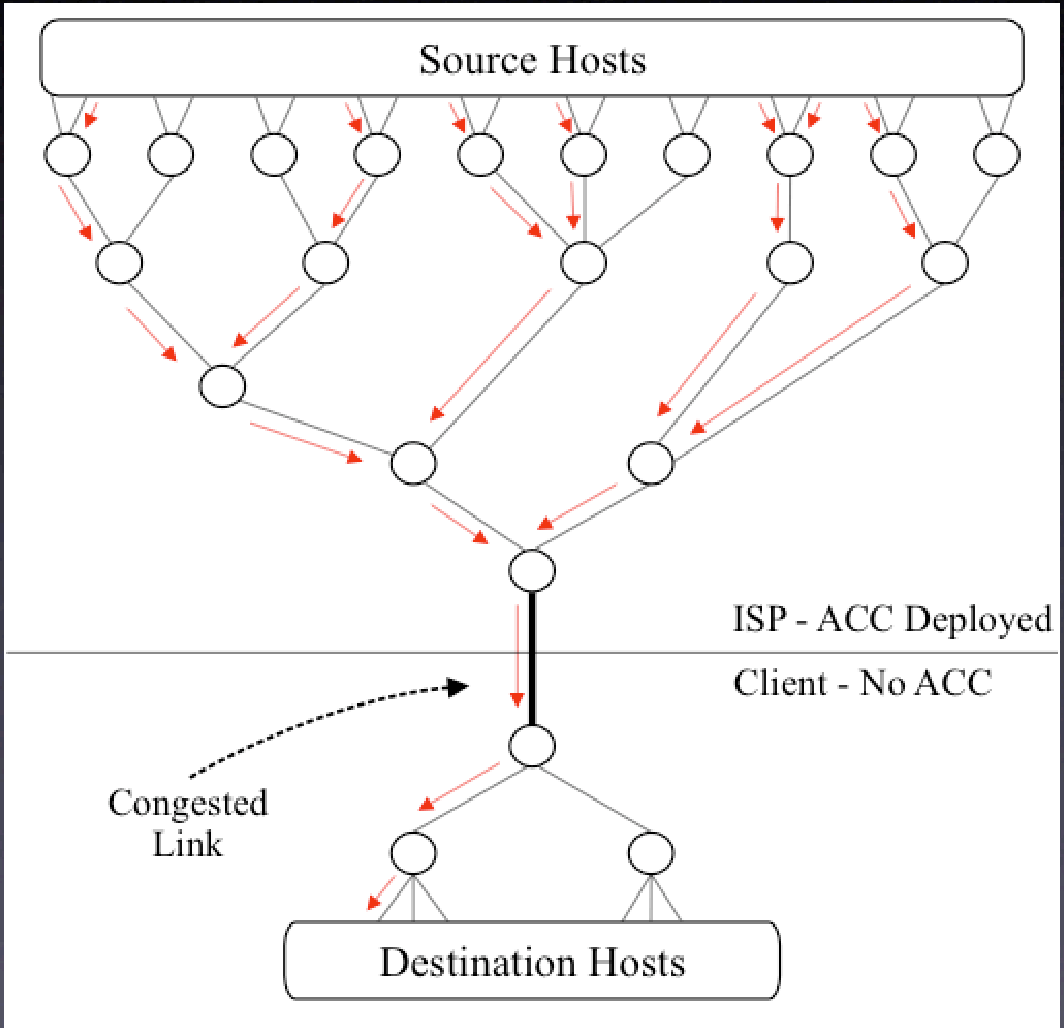
- Am I congested?
  - monitor packet drop rate
- Can I identify the offending flow
  - Sample high volume traffic (dropped packets from RED)
- How much should I rate limit offending flow?
- When do I stop rate limiting

Pushback

- Compare effectiveness of various ACC mechanisms against DDoS attacks
- Requires
  - Accurate bandwidth and latency modeling

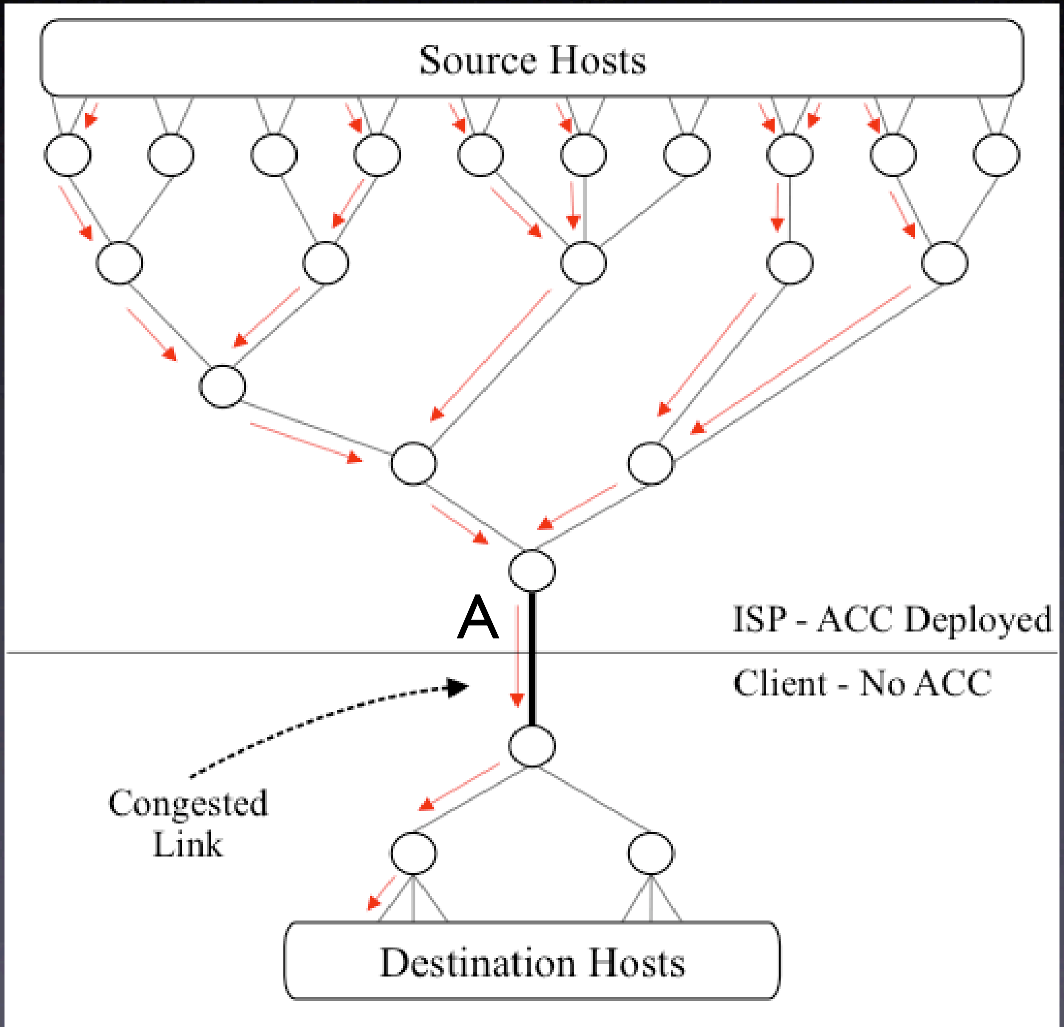
Pushback

- Pushback variants:
  - Pushback
  - Direct pushback (unpublished)
  - On/Off pushback (unpublished)

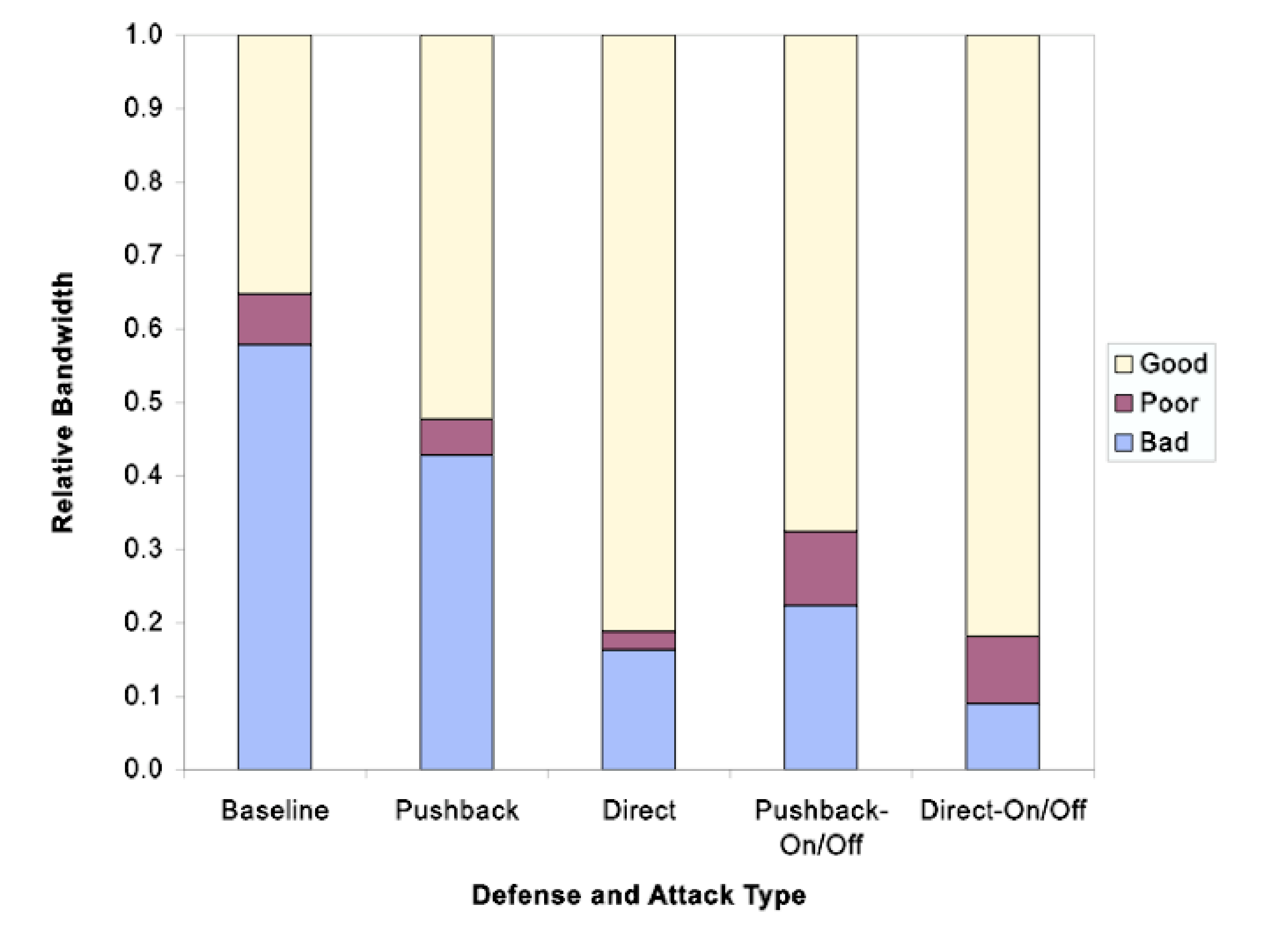


Pushback

- Link A has 3/4 cap. and 2/3 queue size
- Attack traffic from 7 (/20) hosts @ 25 pkts. per sec. toward victim
- Good/poor traffic from 13 (/20) hosts @ 10 pkts. per sec toward 1/6 dests. (including victim)
- 10 min. experiments



Pushback



1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Relative Bandwidth
Good
Poor
Bad
Baseline
Pushback
Direct
Pushback-On/Off
Direct-On/Off
Defense and Attack Type

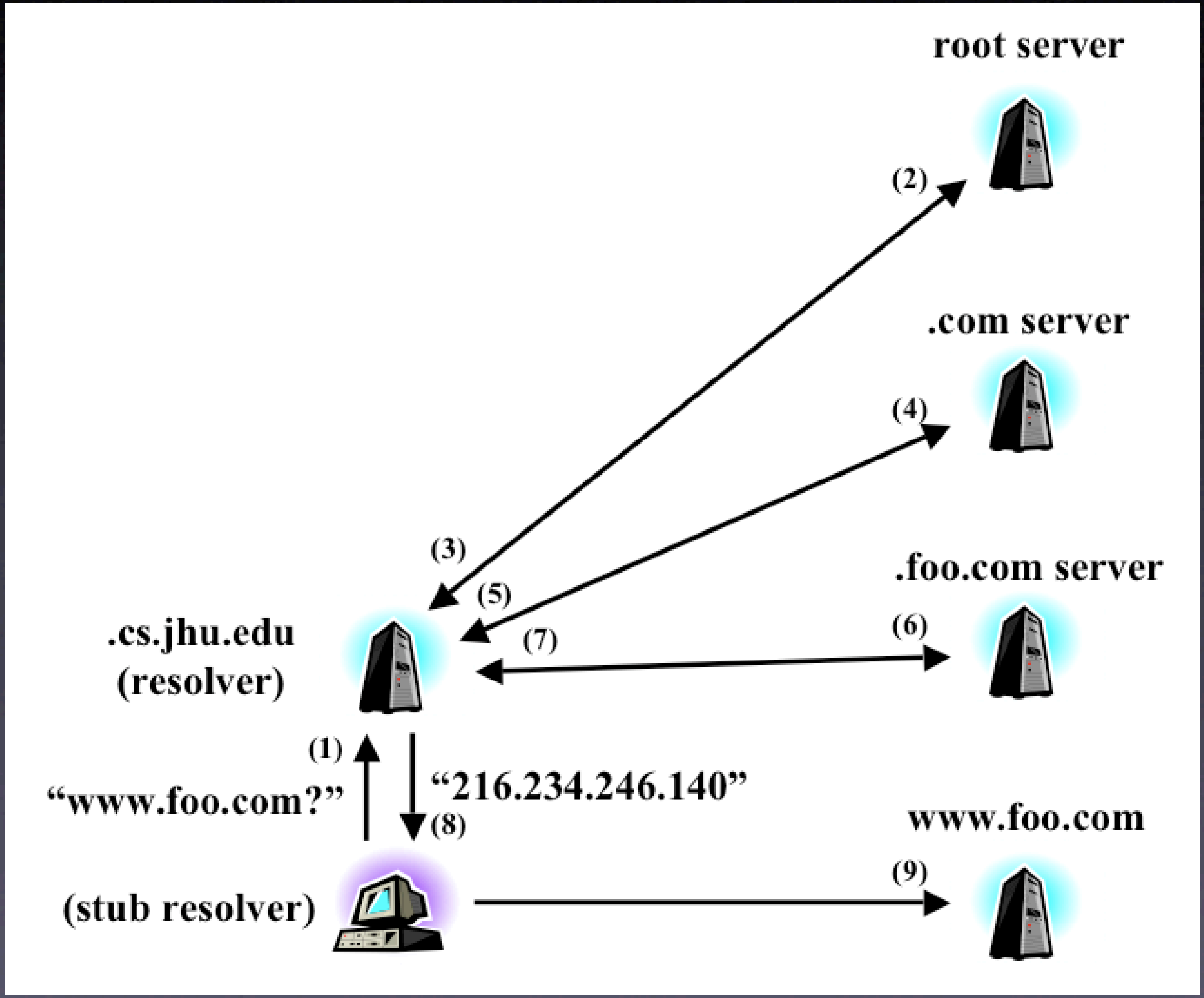
DNSSEC

- Public-key DNSSEC
  - Mitigates DNS spoofing, cache poisoning etc...
  - authenticates RRs

DNSSEC

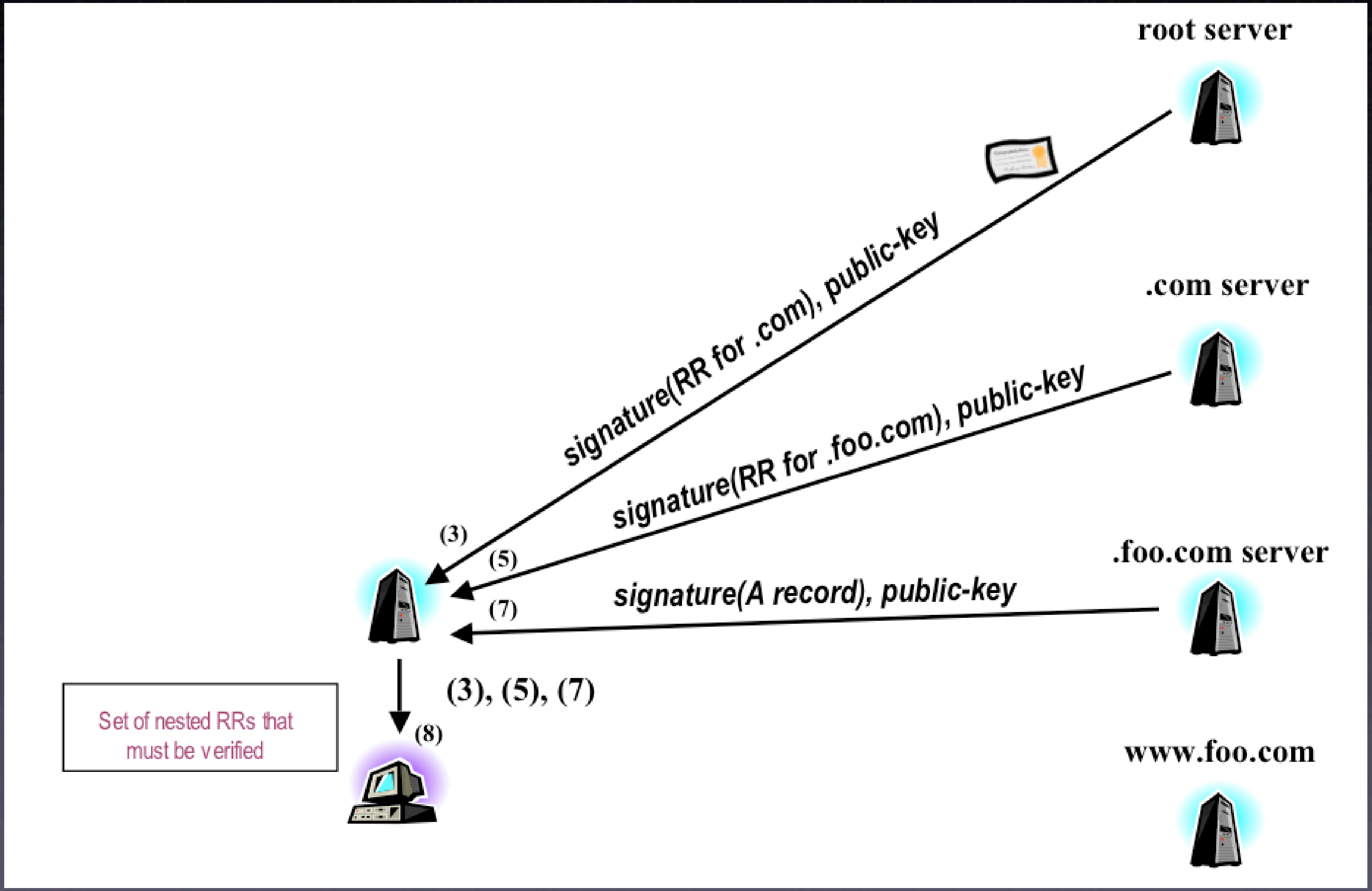
- Overhead in processing time and traffic (no experimental results have ever appeared)
- Requires
  - Modularity
  - Cryptography

DNSSEC



root server
(2)
.com server
(4)
(3)
.foo.com server
(5)
.cs.jhu.edu
(resolver)
(7)
(6)
(1)
"www.foo.com?"
"216.234.246.140"
(8)
www.foo.com
(9)
(stub resolver)

DNSSEC


root server
.com server
.foo.com server
www.foo.com
signature(RR for .com), public-key
signature(RR for .foo.com), public-key
signature(A record), public-key
(3)
(5)
(7)
(3), (5), (7)
(8)
Set of nested RRs that must be verified

DNSSEC

- 40 nodes in .com and .edu domains
- 16 clients (Application level plugins) making
  - type A and NS requests
  - bogus requests
  - domain distribution
  - all according to published results

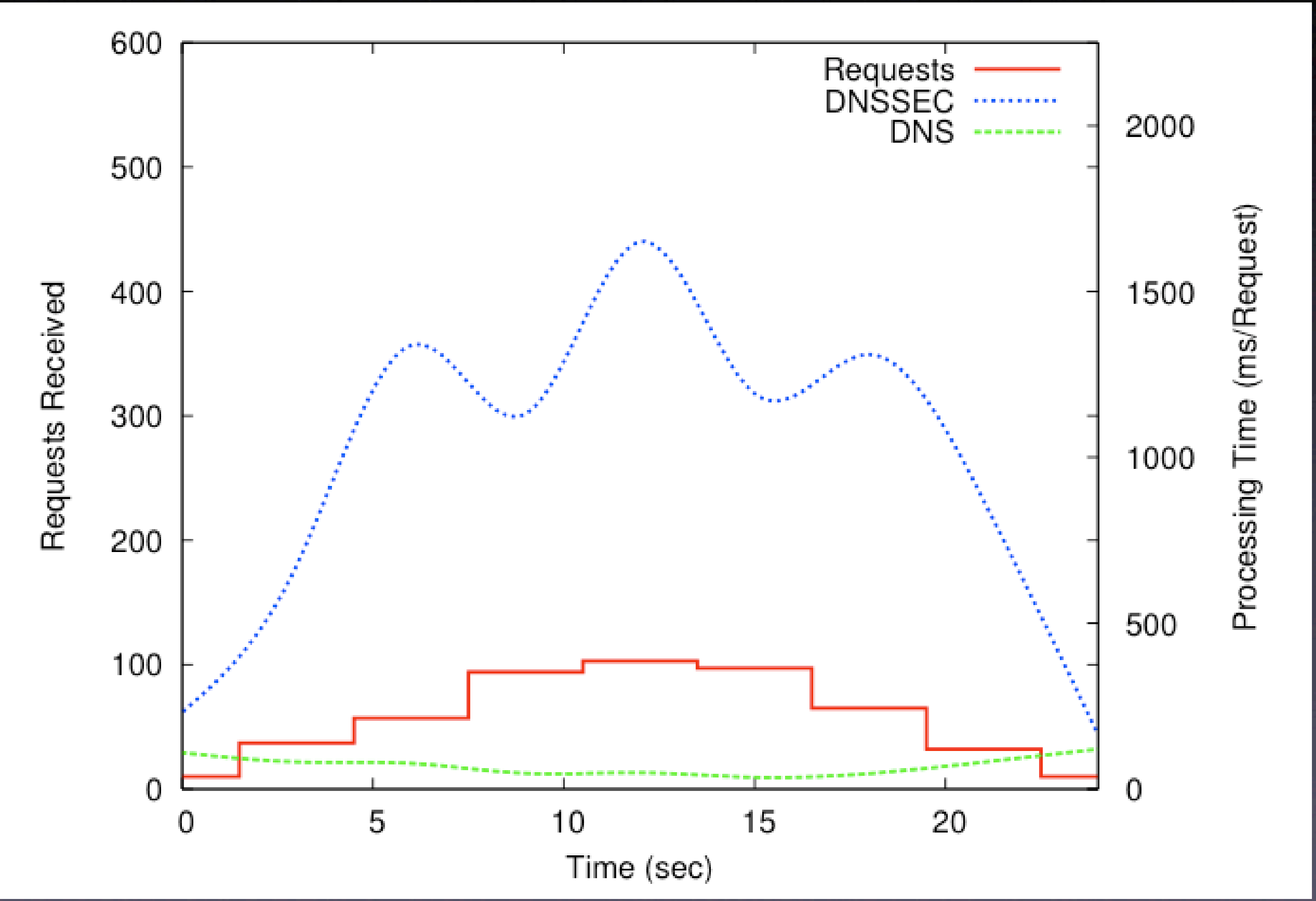
DNSSEC

- 3 second cache duration
- zones resigned every 6 seconds
- 3 second request timeouts
- Cryptographic primitives
  - Signatures: DSA
  - PK encryption: RSA
  - TSIGs: HMAC-MD5

DNSSEC

- Local resolver servicing 3 stub resolvers

DNSSEC



Requests
DNSSEC
DNS
Requests Received
Processing Time (ms/Request)
Time (sec)
600
500
400
300
200
100
0
2000
1500
1000
500
0
0
5
10
15
20

DNSSEC

Node	DNS *Pkts*	DNSSEC *Pkts*	increase (%) *Pkts*	increase (%) *Bytes / Pkts Rcvd*	increase (%) *Bytes / Pkts Sent*
ROOT	124	541	336.2	1.9	505.8
EDU	685	872	23.9	140.1	327.2
COM	393	487	27.2	123.7	358.1
Local (3 Clients)	763	917	20.2	207.5	262.9

- Increase in packets due to public key requests
- Increase in packet size due to signatures, RR sets etc...

Conclusions

- Simnet was designed with security protocols in mind
- Simnet is not meant to replace ns

Conclusions

- Low learning curve
- Highly modular
- Scalable
- Accurate modeling

Implementations

- Network protocols
  - IP, ICMP, UDP, TCP
  - Ping, Traceroute, DNS, NAT

Implementations

- DDoS mitigation protocols
  - Pushback
    - Direct Pushback
  - Synkill

Implementations

- IP traceback schemes
  - PPM
  - SPIE
  - Authenticated and Advanced Marking Schemes

Implementations

- Cryptographic protocols
  - SSL
  - PK-DNSSEC
  - Kerberos
  - Onion routing

Questions?

- Simnet v1.0 available at:

http://simnet.isi.jhu.edu