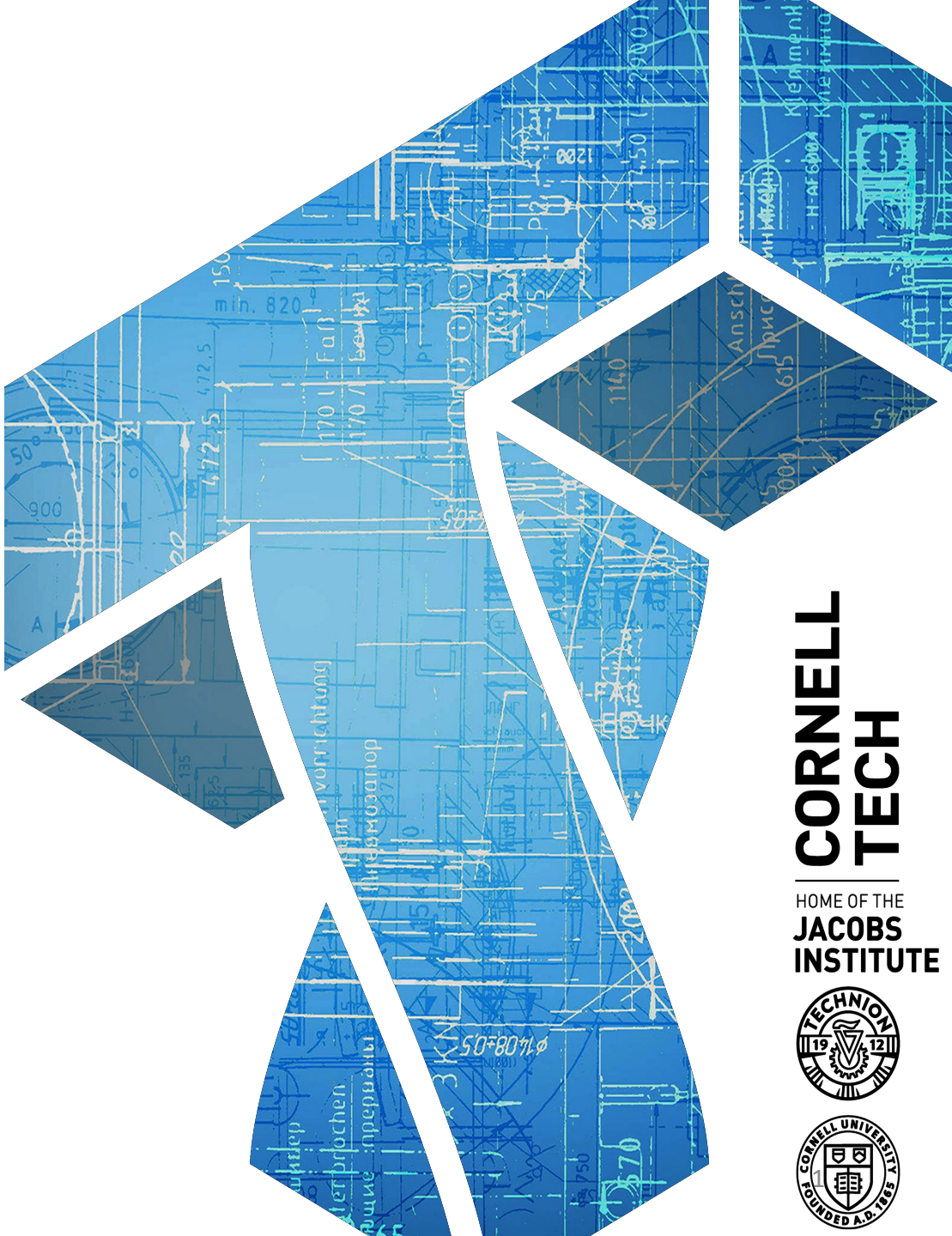
CS 5435: Network security part 2

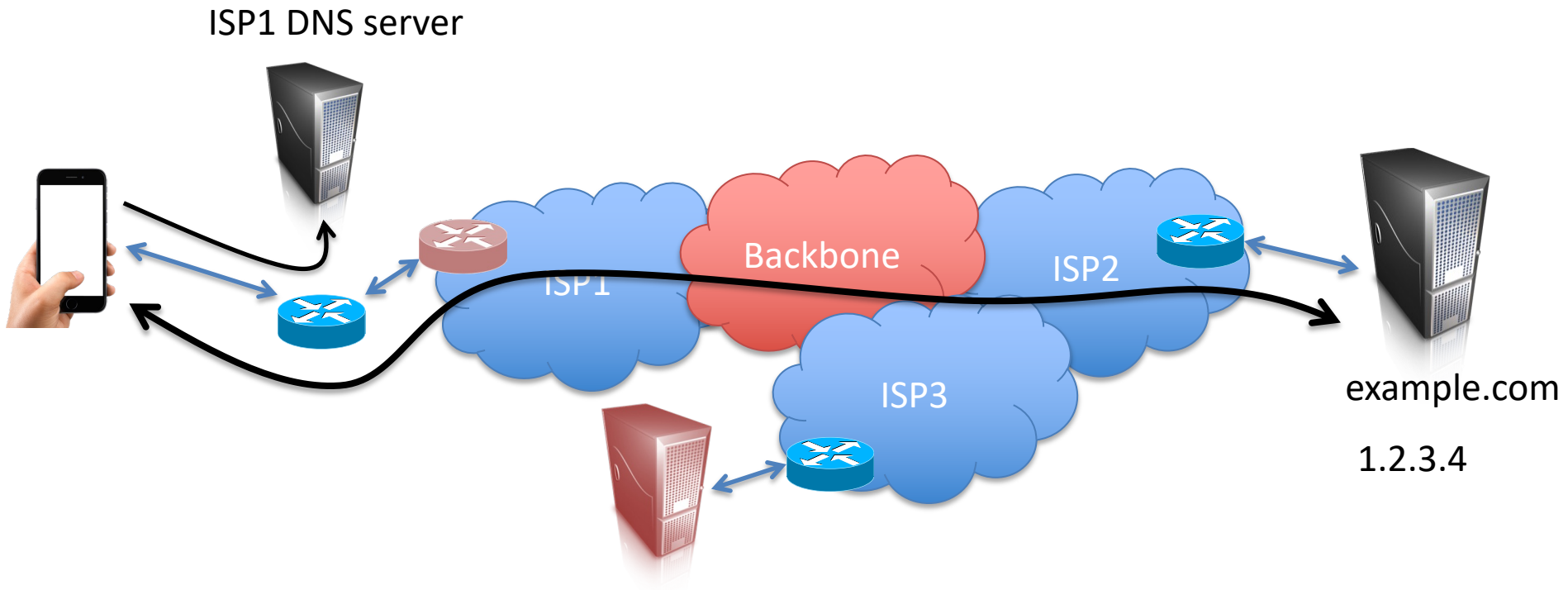
Instructor: Tom Ristenpart

https://github.com/tomrist/cs5435-fall2024



CORNELL TECH
HOME OF THE JACOBS INSTITUTE

Network threat models



(1) Malicious hosts } Off-path attackers

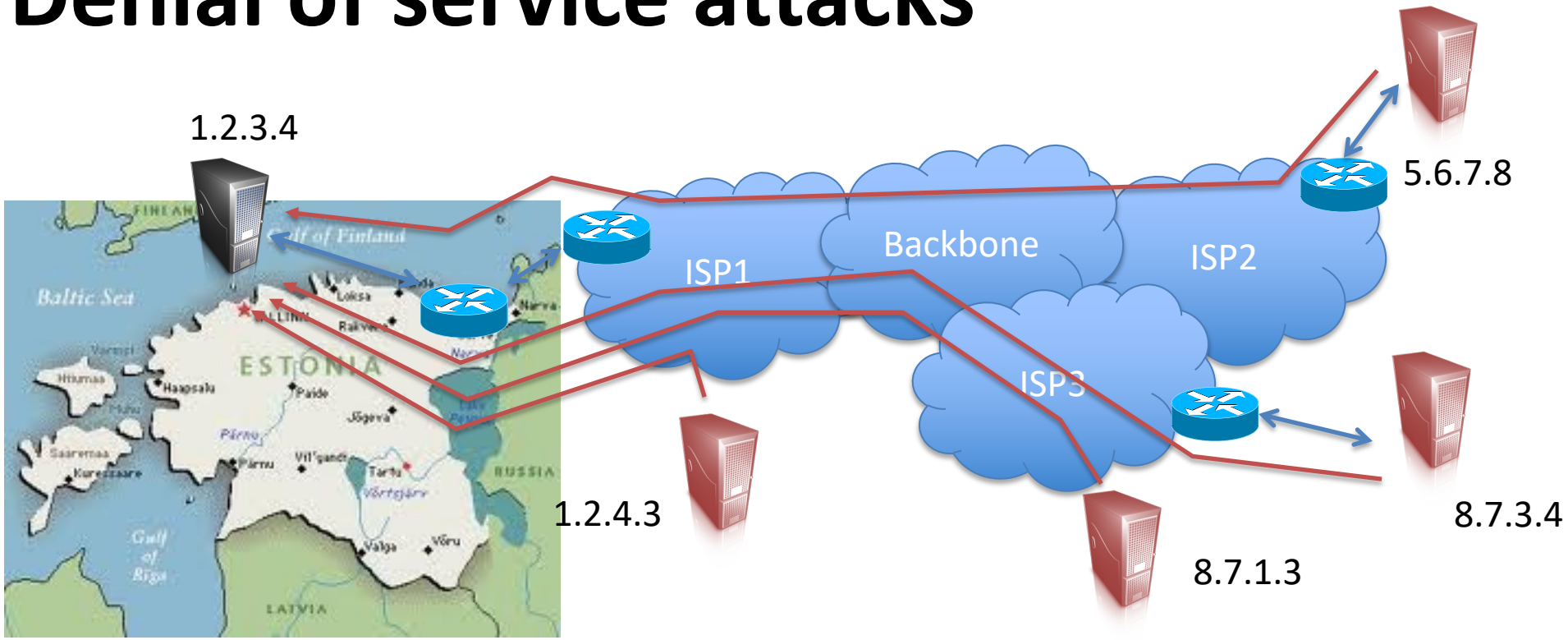
On-path attackers {
(2) Subverted routers or links
(3) Malicious ISPs or backbone

Some attacks we'll cover

- BGP IP hijacking — Defenses: advertisement filtering, BGPsec, RPKI
- DNS cache poisoning — Defenses: Randomize query ID & SRC port, TLS, DNSsec

- Denial of service attacks
- Off-path TCP injection

Denial of service attacks



April 27, 2007

Continued for weeks, with varying levels of intensity

Government, banking, news, university websites

Government shut down international Internet connections

Telegram blames China for 'powerful DDoS attack' during Hong Kong protests

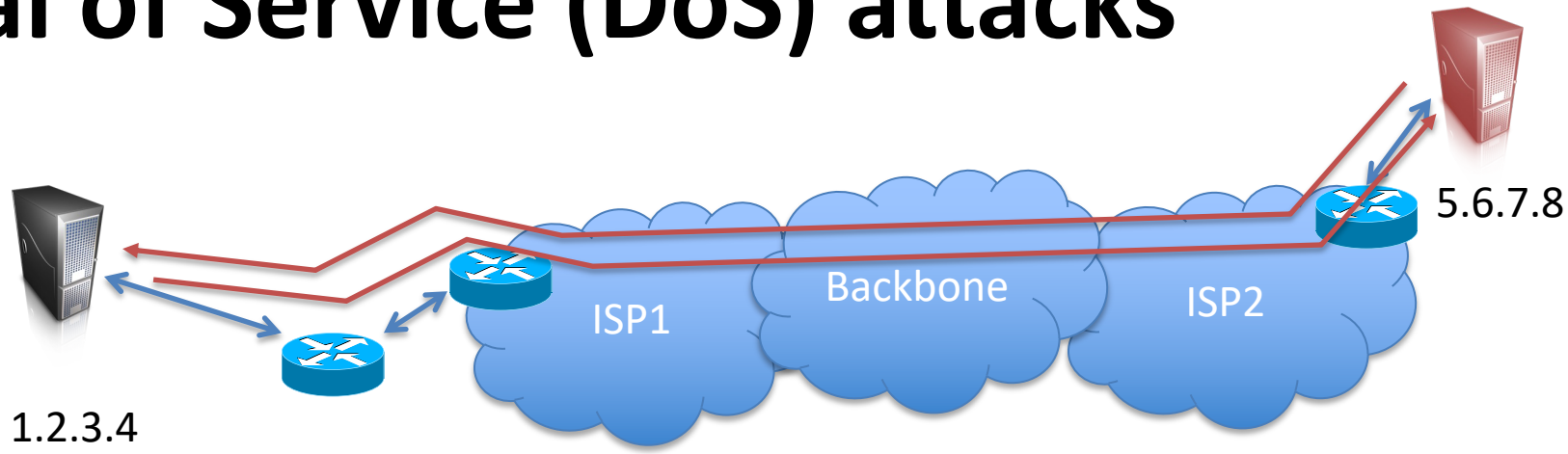
Telegram CEO says 'IP addresses coming mostly from China' were to blame

By Jon Porter | @JonPorty | Jun 13, 2019, 4:21am EDT

"Writing on Twitter, the founder called it a 'state actor-sized DDoS' which came mainly from IP addresses located in China. Durov noted that the attack coincided with the ongoing protests in Hong Kong, where people are using encrypted messaging apps like Telegram to avoid detection while coordinating their protests."

https://www.theverge.com/2019/6/13/18677282/telegram-ddos-attack-china-hong-kong-protest-pavel-durov-state-actor-sized-cyberattack

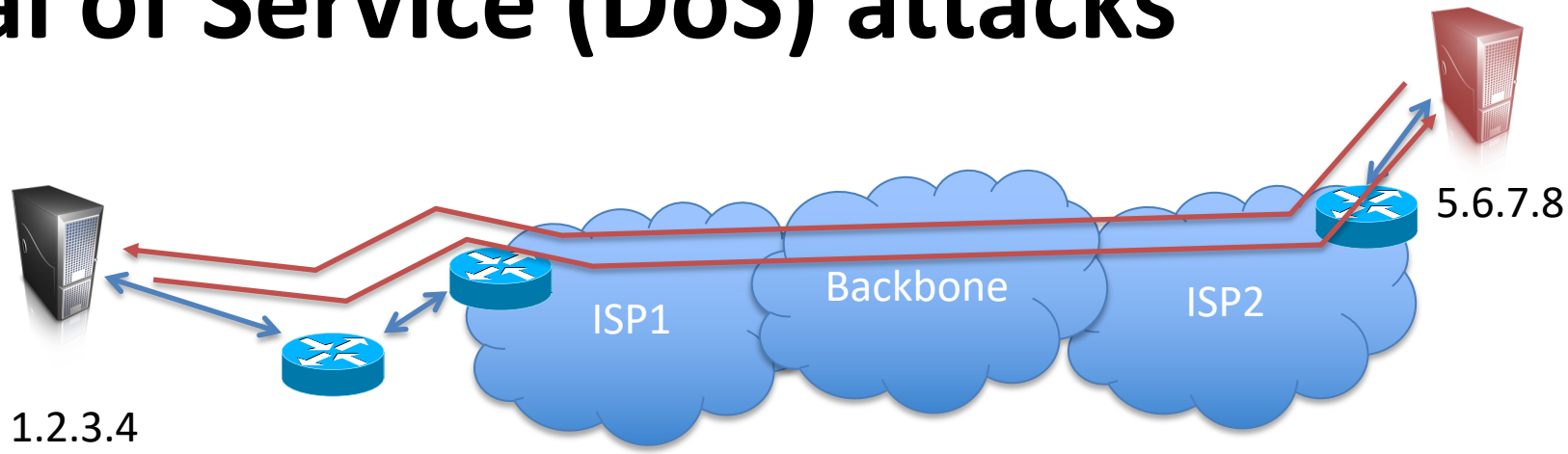
Denial of Service (DoS) attacks



Goal: prevent legitimate users from accessing victim (1.2.3.4)

ICMP ping flood

Denial of Service (DoS) attacks

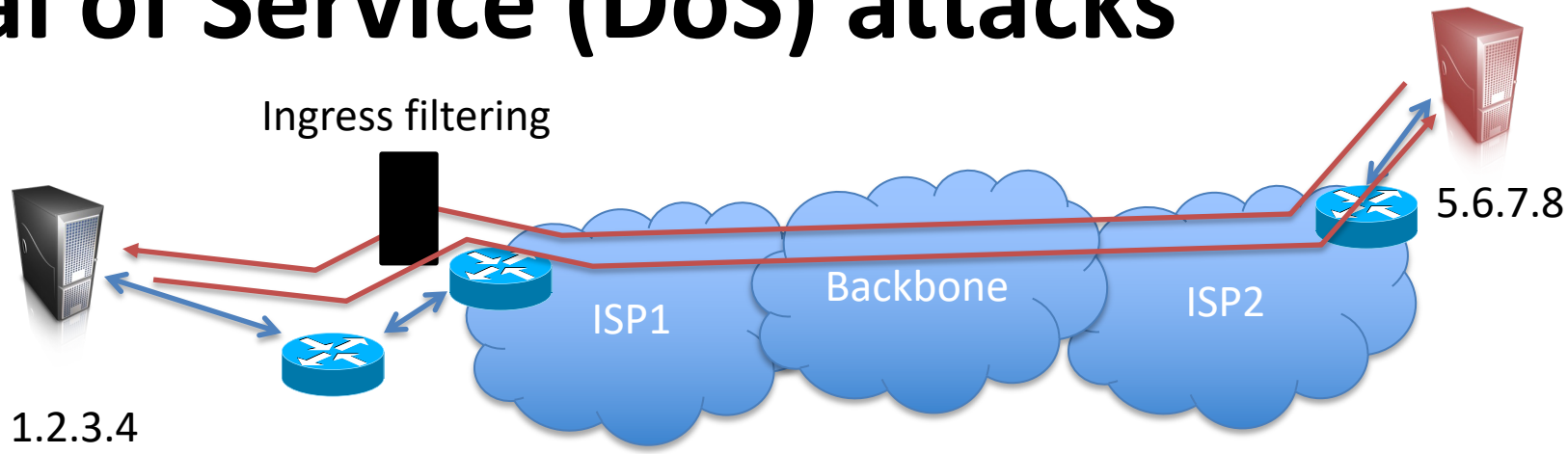


Goal: prevent legitimate users from accessing victim (1.2.3.4)

ICMP ping flood

- Attacker sends ICMP pings as fast as possible to victim
- When will this work as a DoS? Attacker resources > victim's
- How can this be prevented? Ingress filtering of attacker IP addresses near victim once attack identified

Denial of Service (DoS) attacks

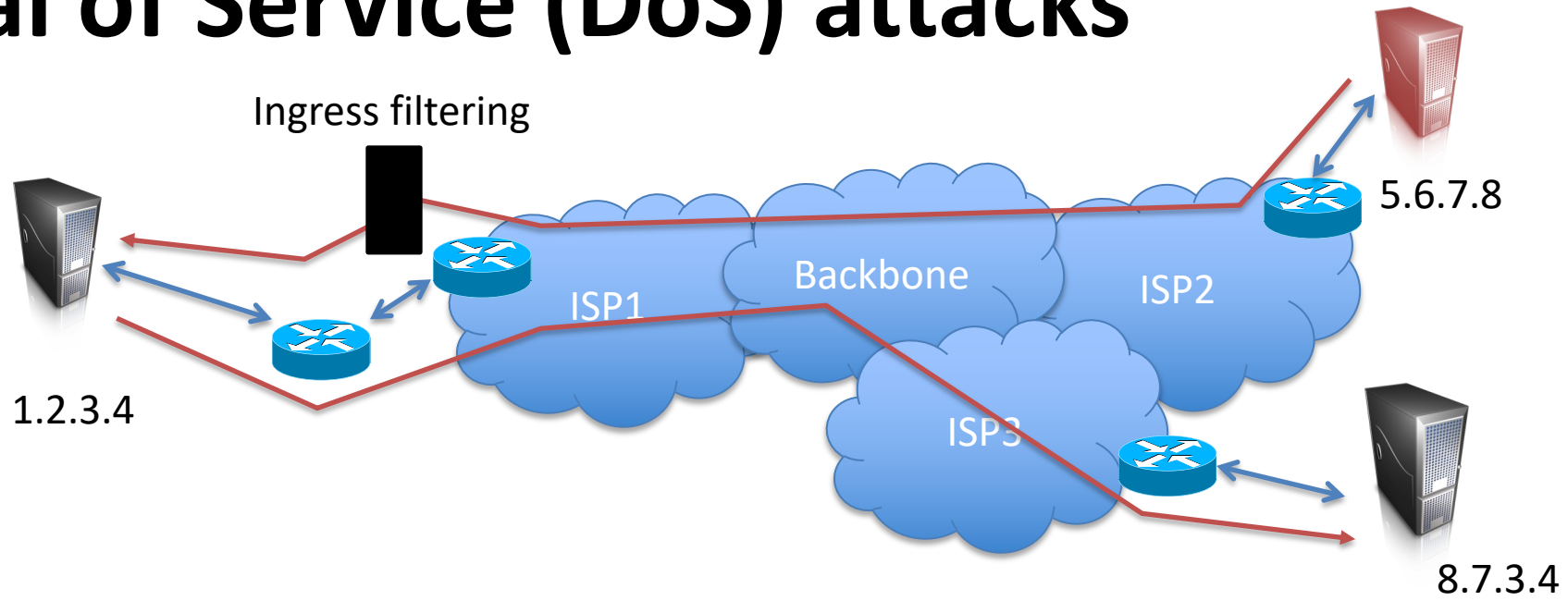


Goal: prevent legitimate users from accessing victim (1.2.3.4)

ICMP ping flood

- Attacker sends ICMP pings as fast as possible to victim
- When will this work as a DoS? Attacker resources > victim's
- How can this be prevented? Ingress filtering of attacker IP addresses near victim once attack identified

Denial of Service (DoS) attacks



How can attacker avoid ingress filtering?

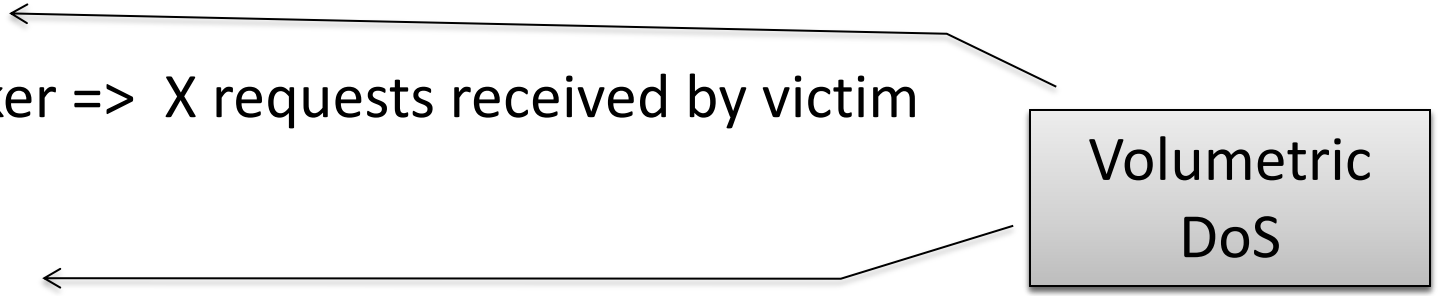
1. Attacker can send packet with fake source IP "spoofed" packet. Packet will get routed correctly. Replies will not

   Send IP packet with source: 8.7.3.4 dest: 1.2.3.4 from 5.6.7.8

2. Distribute attack across many IP addresses

Amplification in DoS attacks

- ***Resource amplification*** (attacker resources > victim's)
  - Number of requests
    - 1 request sent by attacker => X requests received by victim
    - Smurf attack
  - Bandwidth
    - 1 byte sent by attacker => X bytes received by victim
    - ~65 byte DNS request (spoofed SRC) => NS sends ~512 bytes to target
  - Computational time
  - Logical resources on target server



13.02.2024 – 17:09

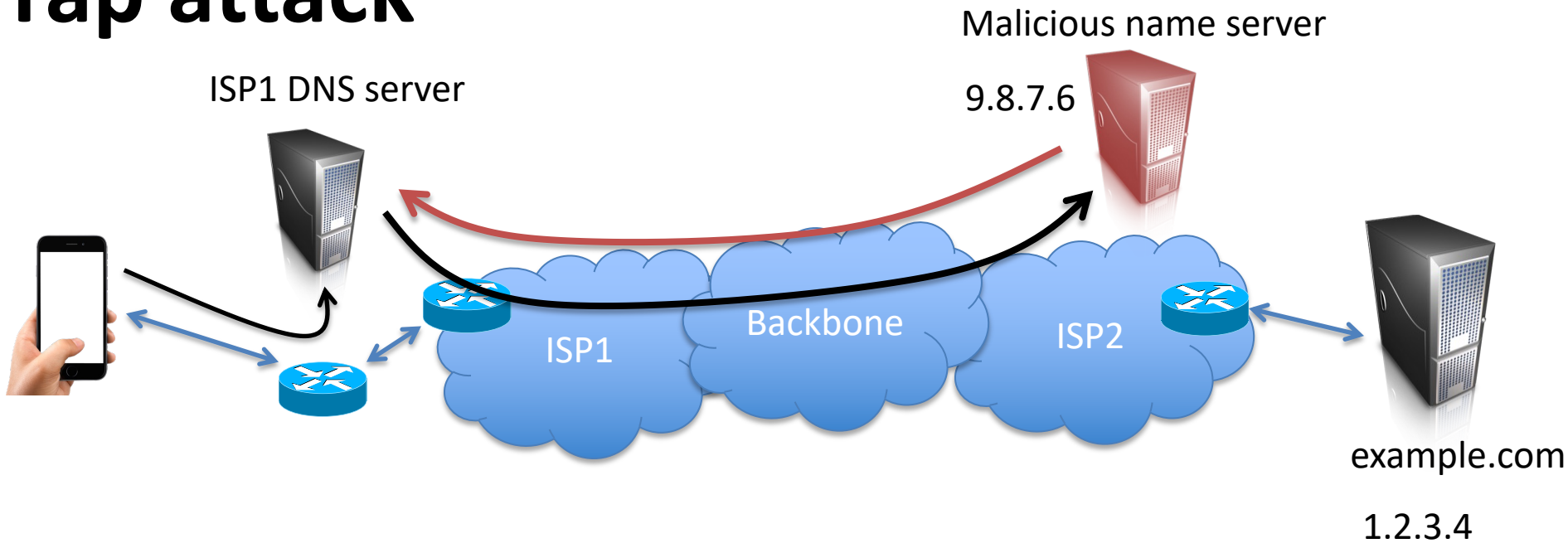
National Research Center for Applied Cybersecurity ATHENE

Serious Vulnerability in the Internet Infrastructure Fundamental design flaw in DNSSEC discovered

KeyTrap DoS algorithmic complexity attack

- Just recently announced attack against DNSSEC
- DNSSEC in brief:
  - Domain owners digitally sign DNS records (RRSIG record)
    - RRSIG record includes key id field
  - DNSKEY record specifies public key for verifying signature
  - DNS resolvers should try all ways to validate signatures:
    - DNS resolvers run signature verification with every DNSKEY record that **matches each** RRSIG record by checking (owner name, algorithm, key id)
    - Multiple DNSKEY records can match an RRSIG record. If two DNSKEY records match an RRSIG record, we call them ***colliding keys***

KeyTrap attack

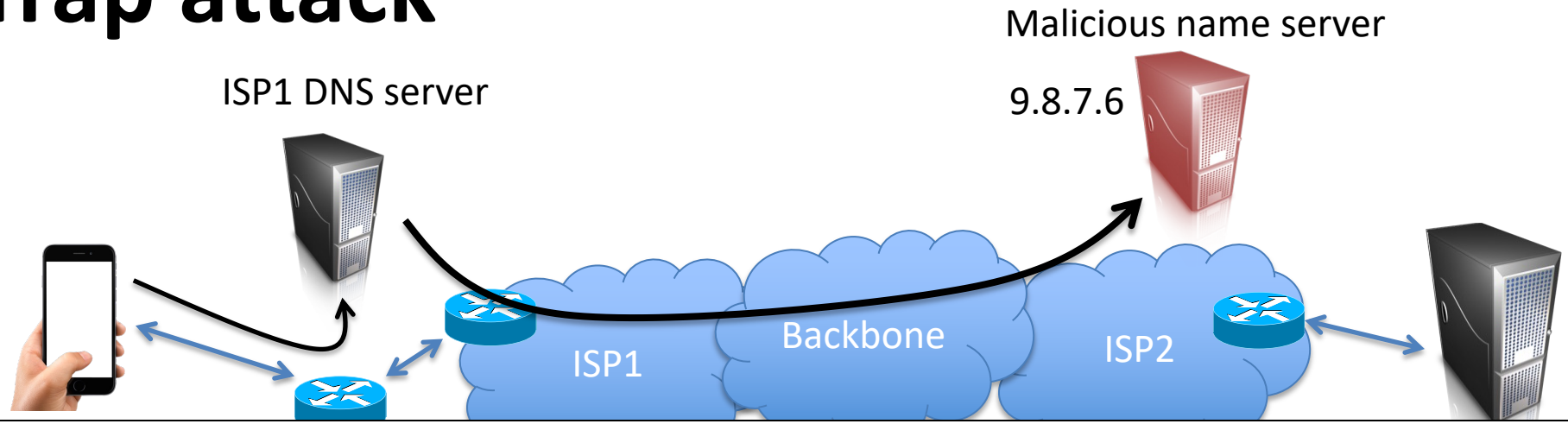


Malicious name server sends specially crafted UDP DNS response:

- 582 colliding DNSSEC keys
- 340 signatures
- Requires 582 x 340 = 197,880 signature verifications

Amplification: O(n) work by malicious name server causes $O(n^2)$ work at target resolver

KeyTrap attack



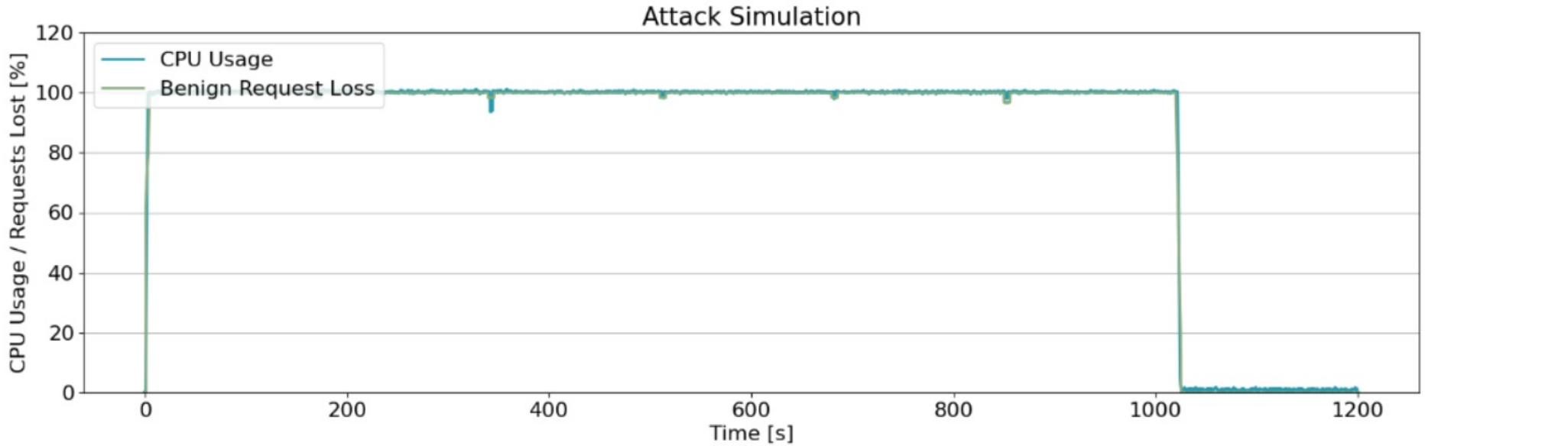
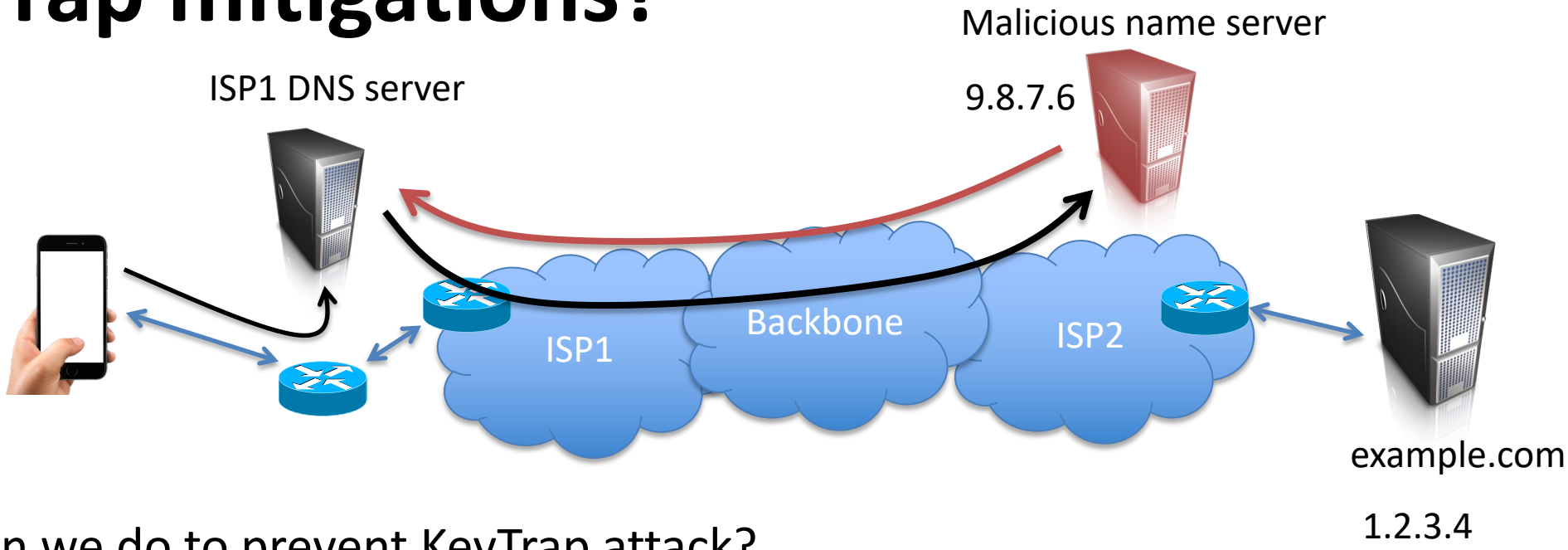


Fig. 1: KeySigTrap attack on Unbound with single request.

From: https://www.athene-center.de/fileadmin/content/PDF/Keytrap_2401.pdf

KeyTrap mitigations?



What can we do to prevent KeyTrap attack?

Limit total number of validations per request to to 8 (Akamai patch)

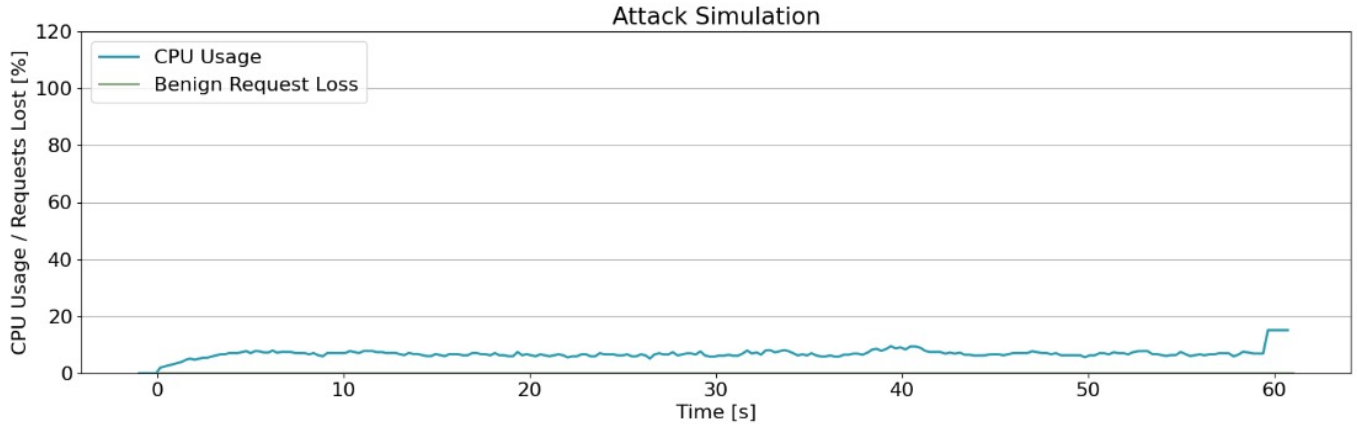
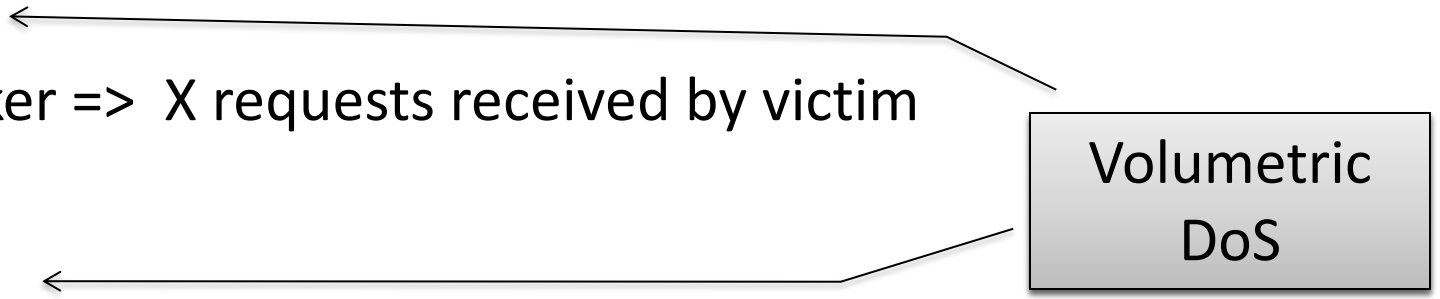


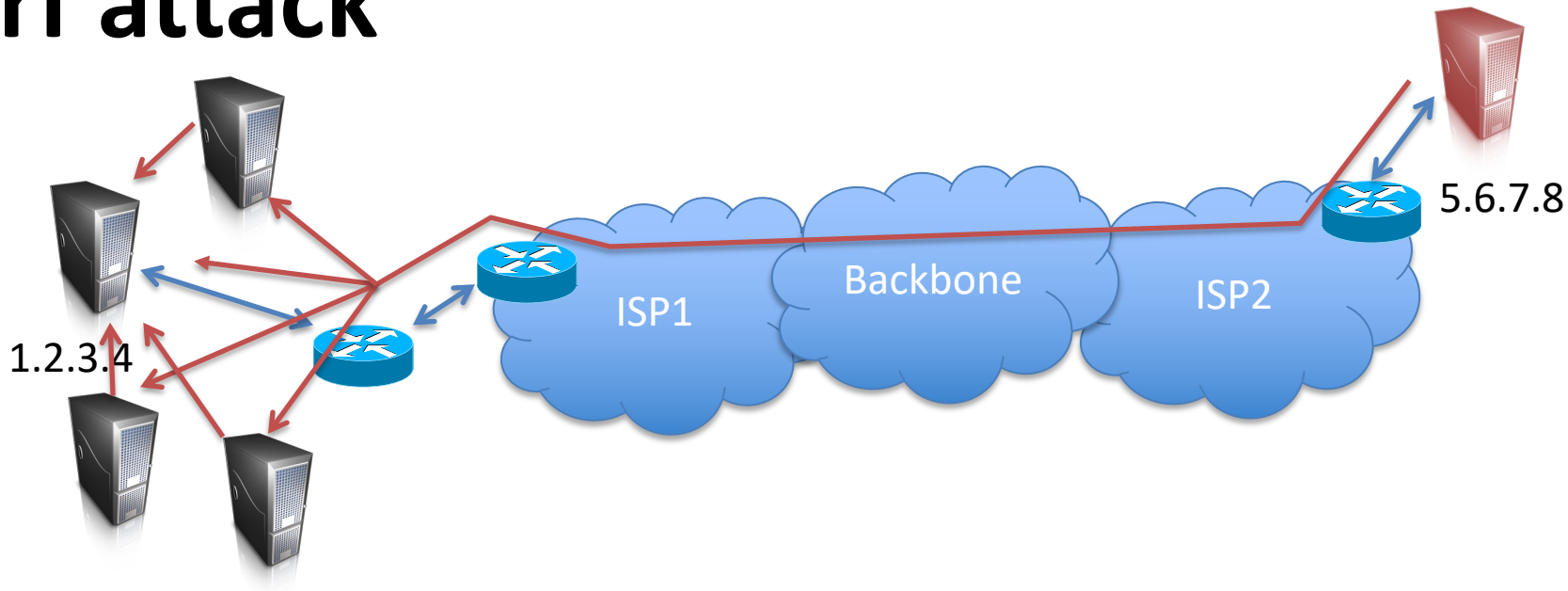
Fig. 12: ANY type attack with 10req/s on patched Akamai.

Amplification in DoS attacks

- ***Resource amplification***
  - Number of requests
    - 1 request sent by attacker => X requests received by victim
    - Smurf attack
  - Bandwidth
    - 1 byte sent by attacker => X bytes received by victim
    - ~65 byte DNS request (spoofed SRC) => NS sends ~512 bytes to target
  - Computational time
    - Small number of requests cause significant CPU usage
  - Logical resources on target server



Smurf attack



Abuse of IP broadcast addresses

- Attacker sends broadcast ICMP ping to vulnerable network
- Spoof SRC address to be 1.2.3.4

Mitigated in now new routers (don’t respond to ICMP broadcast pings)

ICMP sometimes blocked on networks defensively

Memcached DDoS attacks

- Memcached is popular in-memory data store
- Supports UDP requests
- Attack:
  - Insert large data item into Memcached server
  - Send 10 UDP requests / second to Memcached server
    - with SRC IP = GitHub
- Relies on vulnerable memcached servers
  - Default configuration: accept UDP requests from anywhere

https://github.blog/2018-03-01-ddos-incident-report/

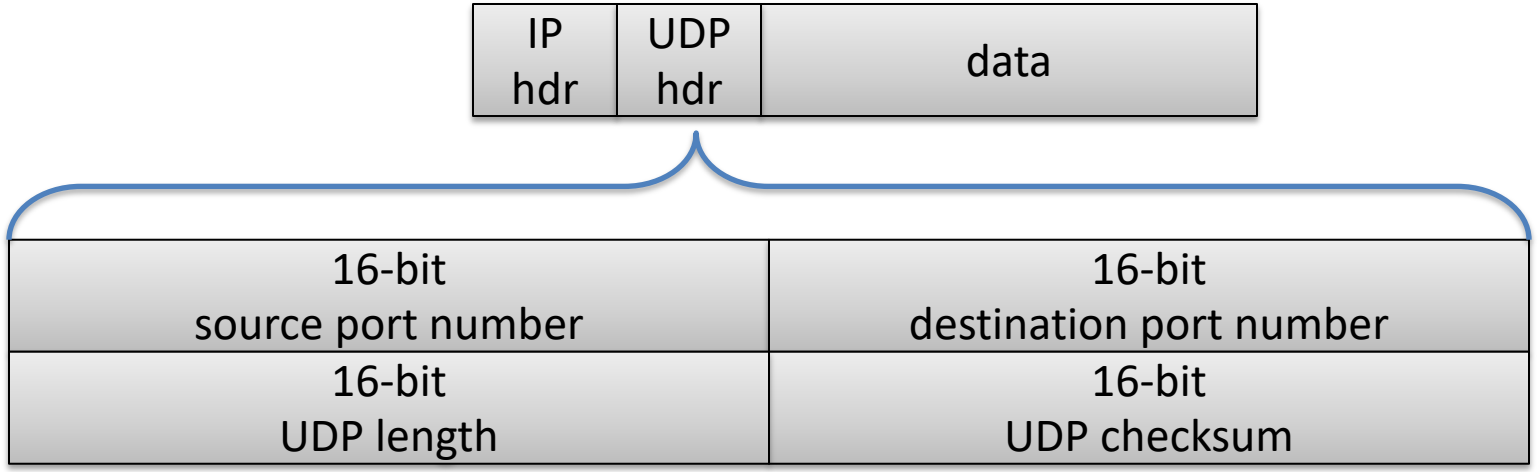
Memcached DDoS attacks

- Feb 2018 attack against GitHub
  - **51,000x** bandwidth magnification
  - 1.3 TB/s of traffic to GitHub, using 126 million packets per second
  - Over 1,000 AS's originate traffic
  - GitHub offline for 5 minutes
- Response?
  - Use BGP announcement to route GitHub traffic through Akamai
  - Akamai gives more capacity + helps filter out bogus requests
  - Turn off UDP support in memcached (default now is off)

Why UDP in DoS reflection attacks?

- DNS, memcached, … application-layer protocols running on UDP often exploited in DoS attacks
- Single packet to victim service yields response, so spoofing works

Application	HTTP, DNS, FTP, SMTP, SSH, etc.
Transport	TCP, UDP
Network	IP, ICMP, …
Link	802x (802.11, Ethernet)

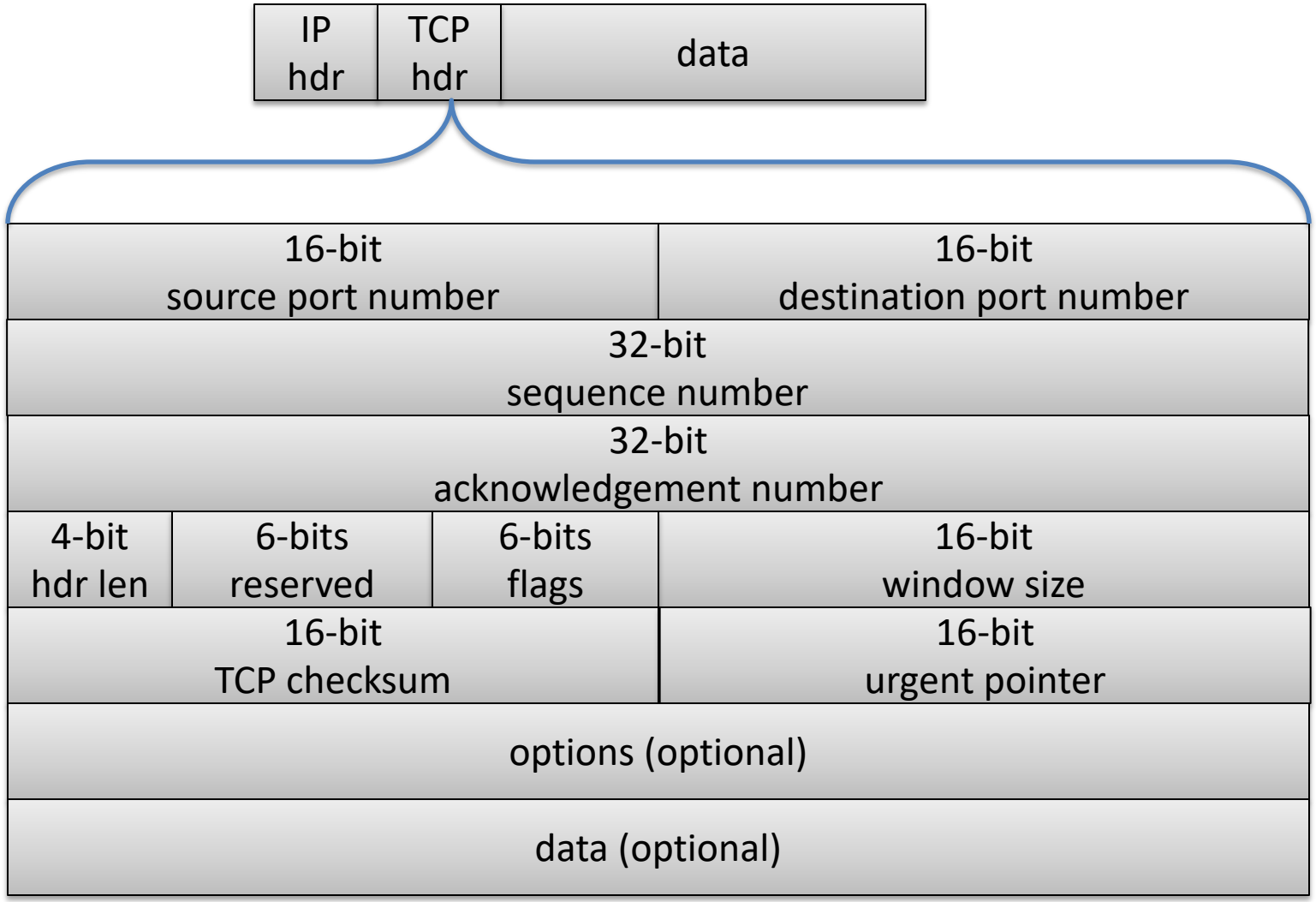


length = header len + data len

TCP (transport control protocol)

- Connection-oriented
  - state initialized during handshake and maintained
- Reliability is a goal
  - generates segments
  - timeout segments that aren’t ack’d
  - checksums headers
  - reorders received segments if necessary
  - flow control to avoid congestion

TCP (transport control protocol)

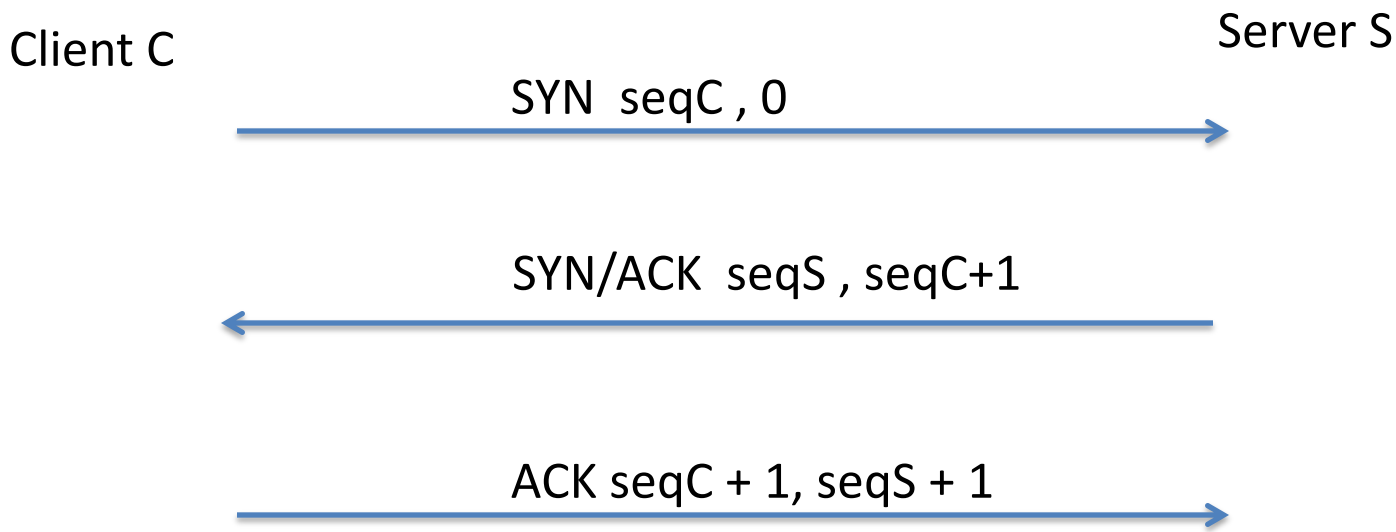


TCP (transport control protocol)

TCP flags:

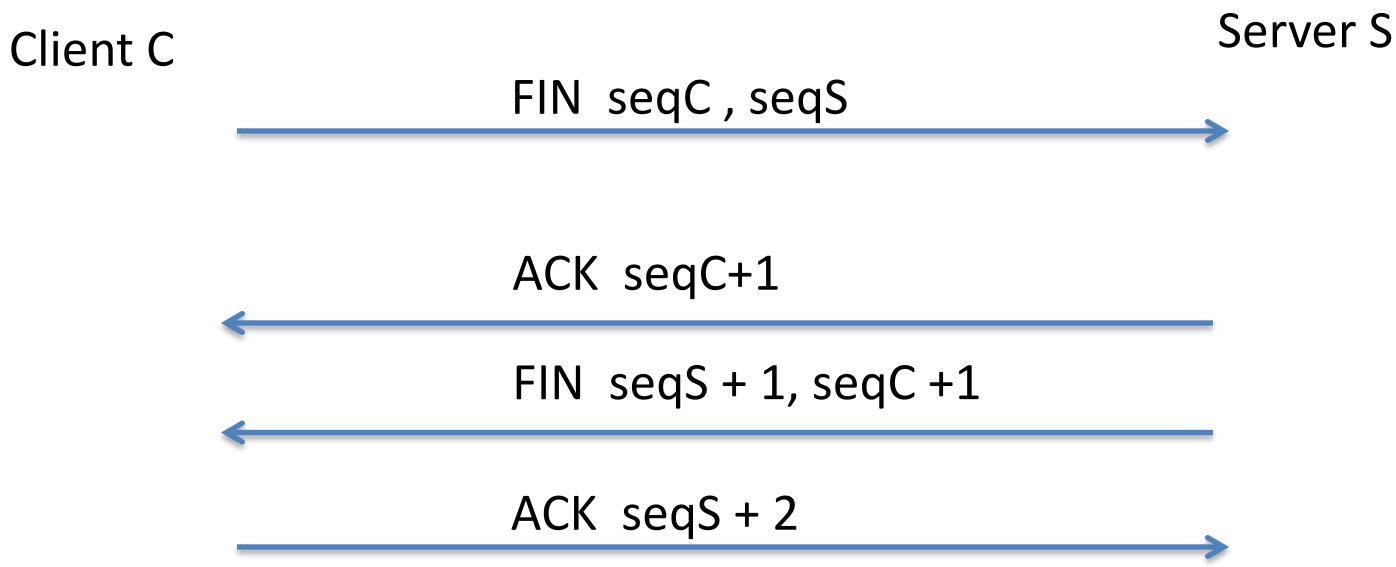
URG	urgent pointer valid
ACK	acknowledgement number valid
PSH	pass data to app ASAP
RST	reset connection
SYN	synchronize sequence #’s
FIN	finished sending data

TCP handshake



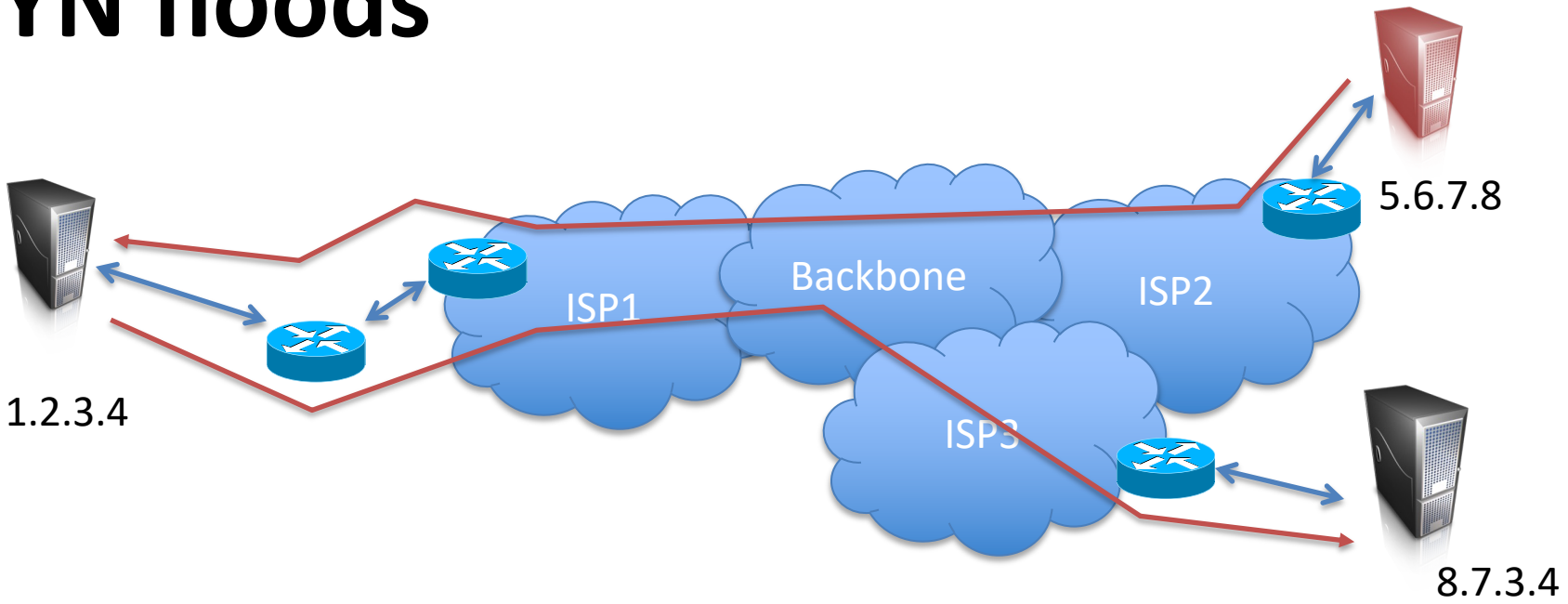
SYN = syn flag set
ACK = ack flag set
seqC,seqS = initial client and server sequence numbers

TCP connection termination



FIN = fin flag set
ACK = ack flag set
seqC,seqS = current client and server sequence numbers

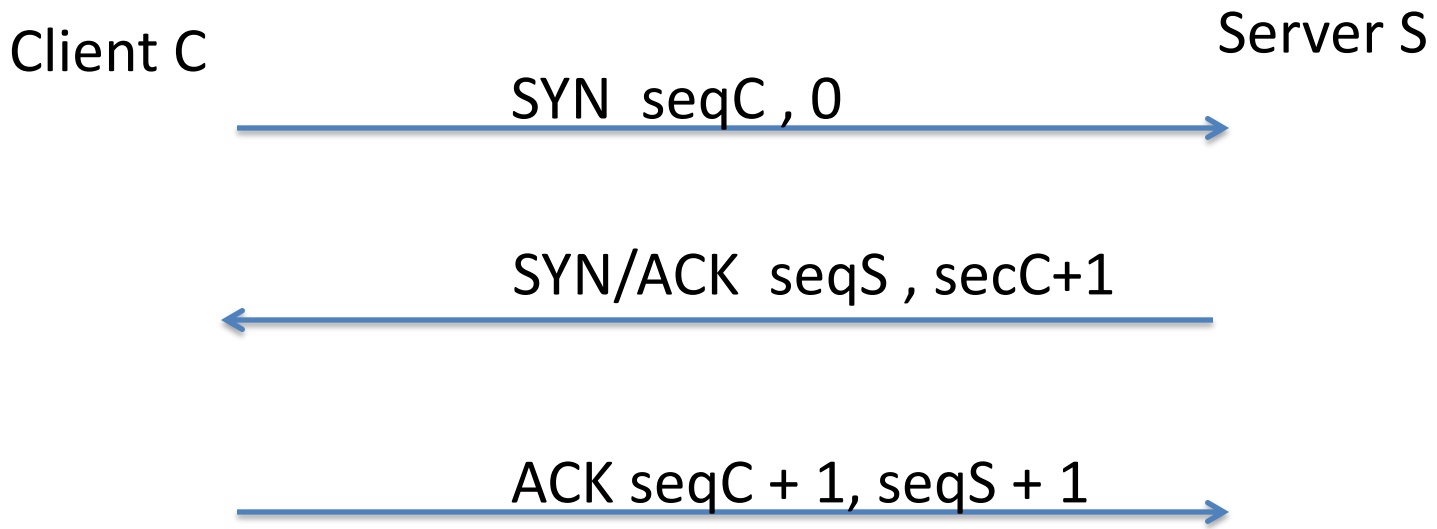
TCP SYN floods



5.6.7.8 sends TCP SYN packets to 1.2.3.4

- 1.2.3.4 maintains state for each SYN packet for some window of time
- If 5.6.7.8 sets SRC IP to be 8.7.3.4, what does 8.7.3.4 receive?

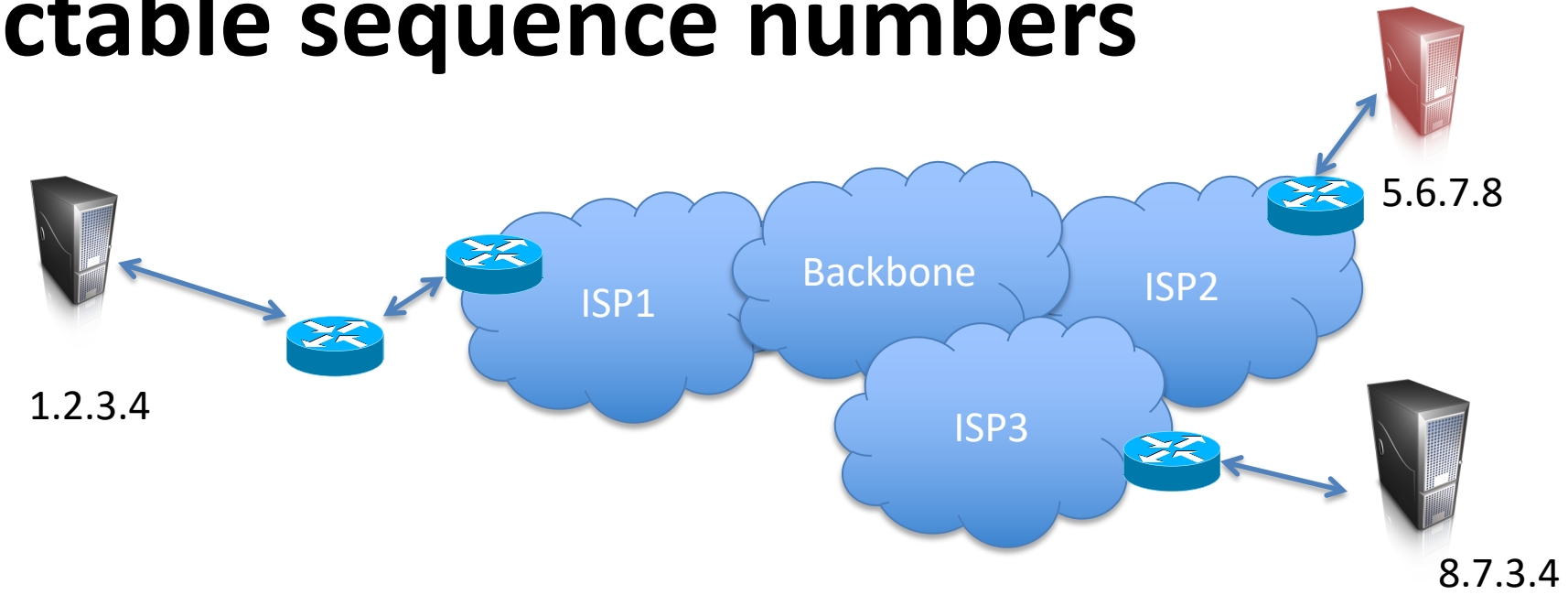
TCP handshake



How are seqC and seqS selected?

Initial sequence numbers must vary over time so that different connections don't get confused

Predictable sequence numbers

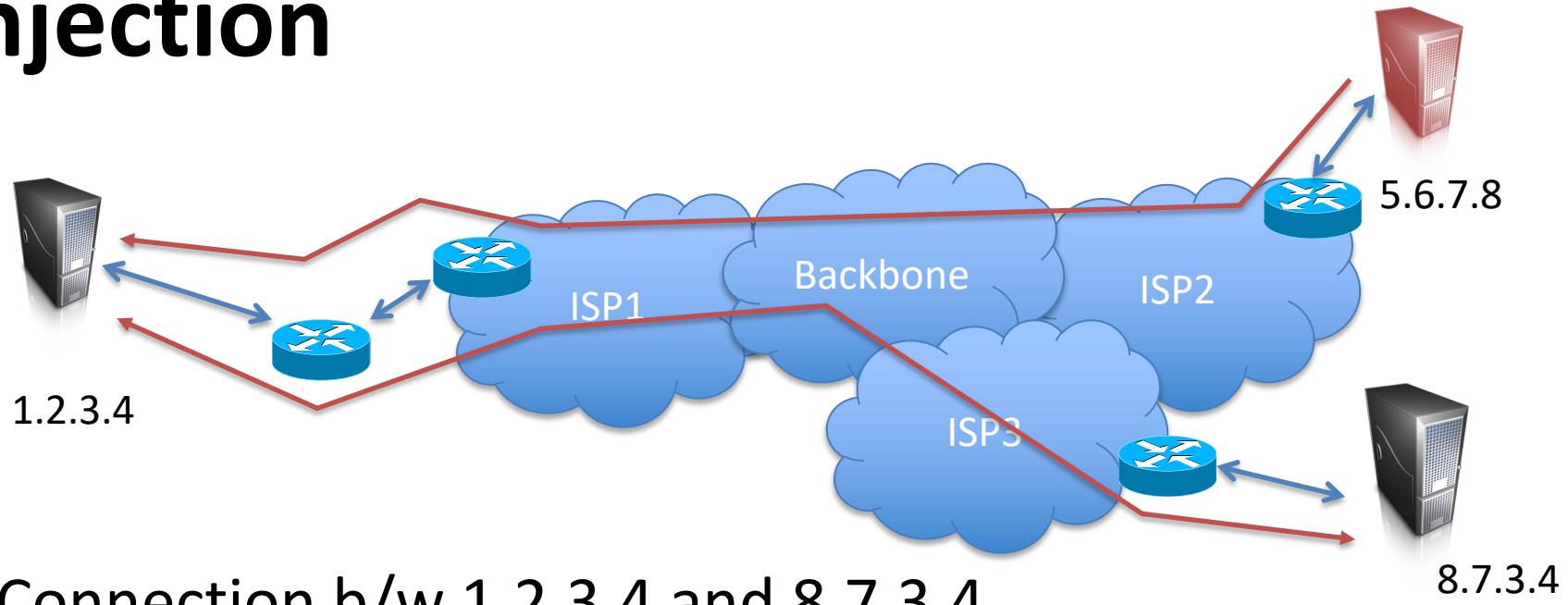


4.4BSD used predictable initial sequence numbers (ISNs)

- At system initialization, set ISN to 1
- Increment ISN by 64,000 every half-second

What can a clever attacker do?
(assume spoofing possible)

TCP injection



Connection b/w 1.2.3.4 and 8.7.3.4

Forge a FIN packet from 8.7.3.4 to 1.2.3.4

src: 8.7.3.4
dst: 1.2.3.4

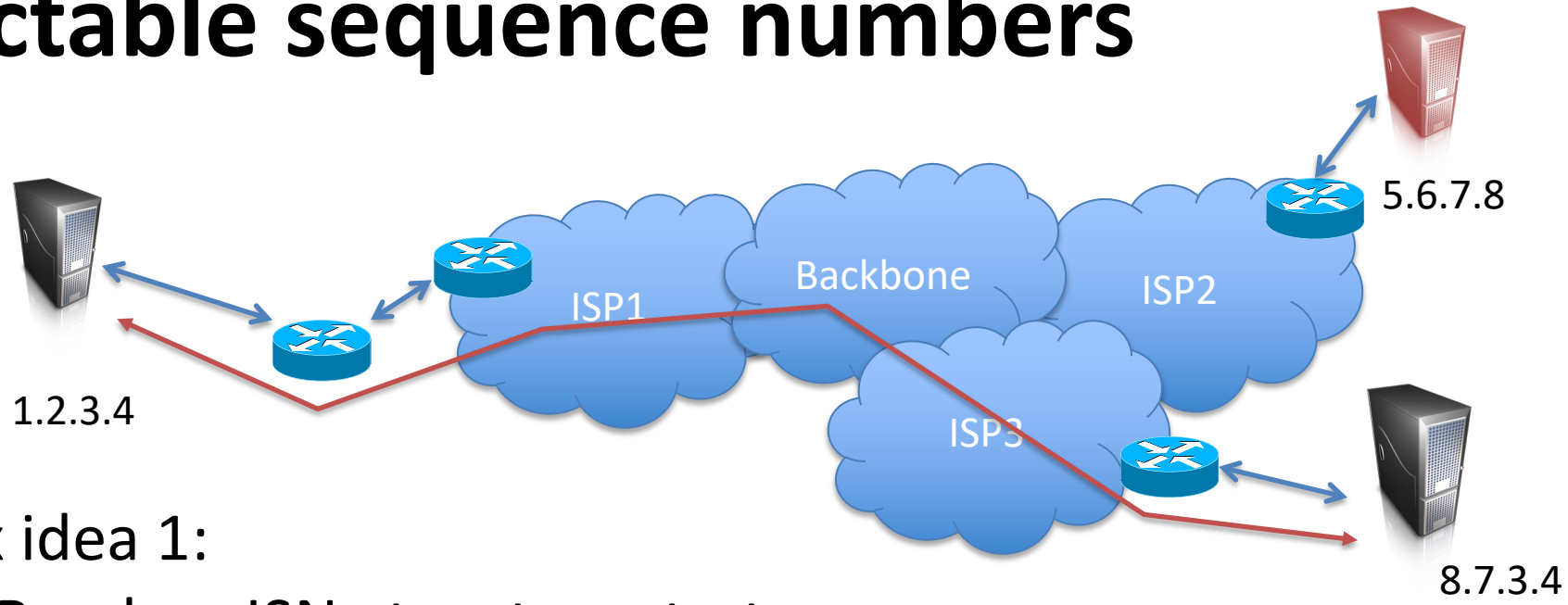
seq#(8.7.3.4)
FIN

Forge some application-layer packet from 8.7.3.4 to 1.2.3.4

src: 8.7.3.4
dst: 1.2.3.4

seq#(8.7.3.4)
“rsh rm –rf /”

Predictable sequence numbers



Fix idea 1:

- Random ISN at system startup
- Increment by 64,000 each half second

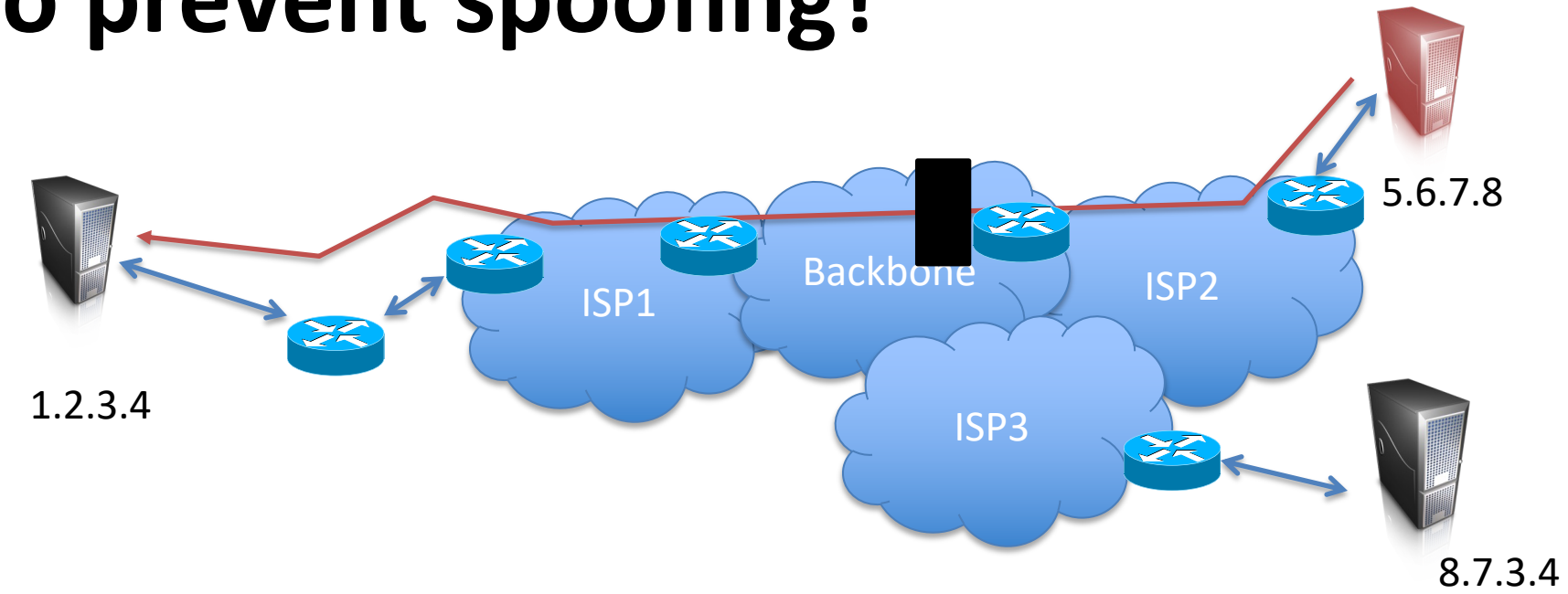
Better fix:

- Random ISN for every connection

Remains an issue in some cases:

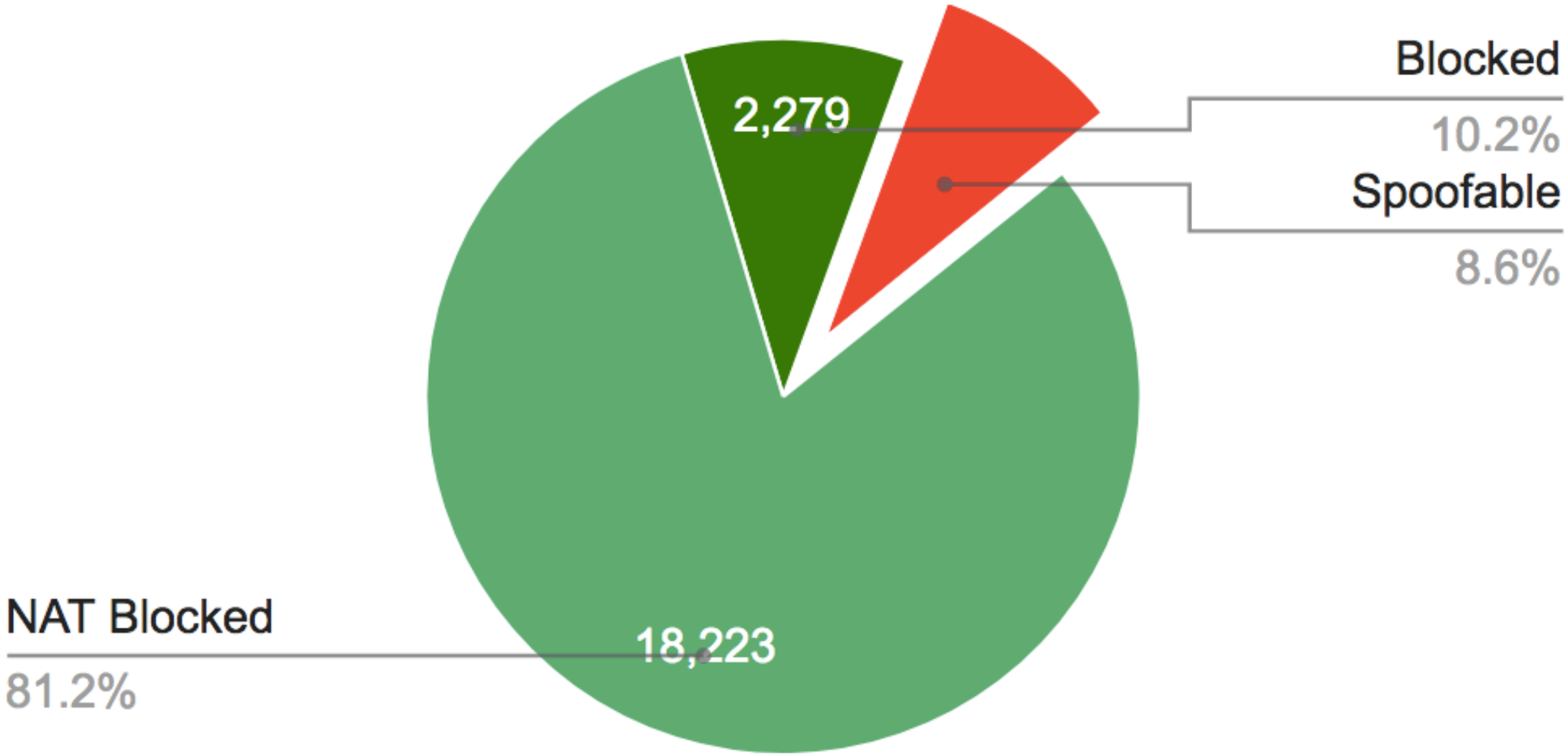
- Any FIN accepted with seq# in receive window: 2^{17} attempts
- Side-channel attacks to infer seq#

How to prevent spoofing?



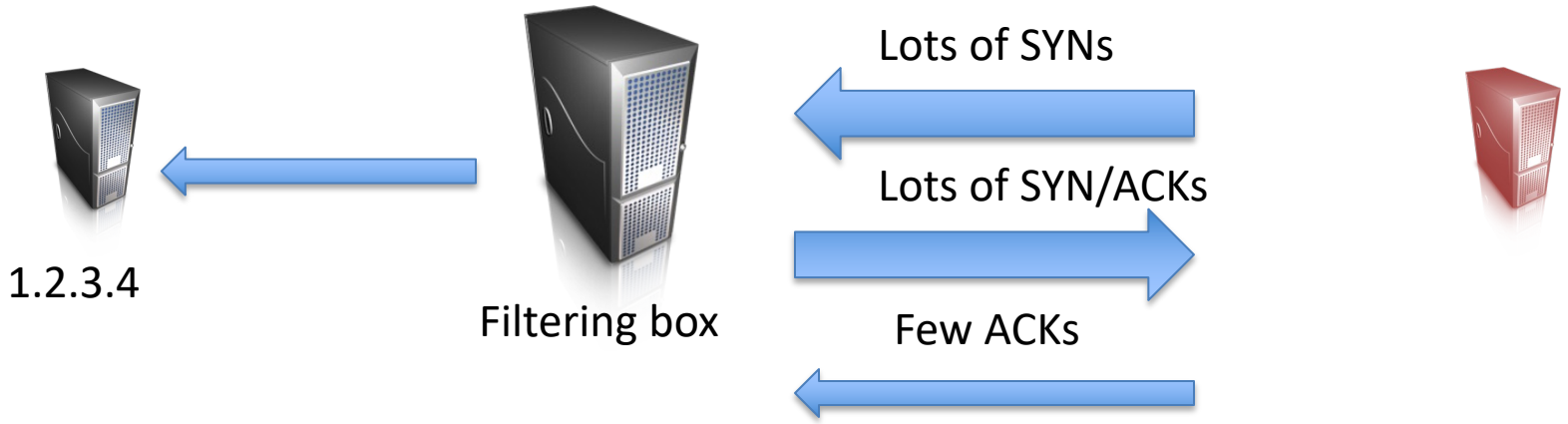
- IP traceback: techniques for inferring actual source of a (spoofed) packet
- BCP 38 (RFC 2827): upstream ingress filtering to drop spoofed packets
  - Ideally, all network traffic providers would perform ingress filtering

IPv4 blocks (including NAT)



https://spoofer.caida.org/summary.php

Preventing DoS example: Prolexic approach



Filter out (“scrub”) requests

Prolexic purchased by Akamai in 2014

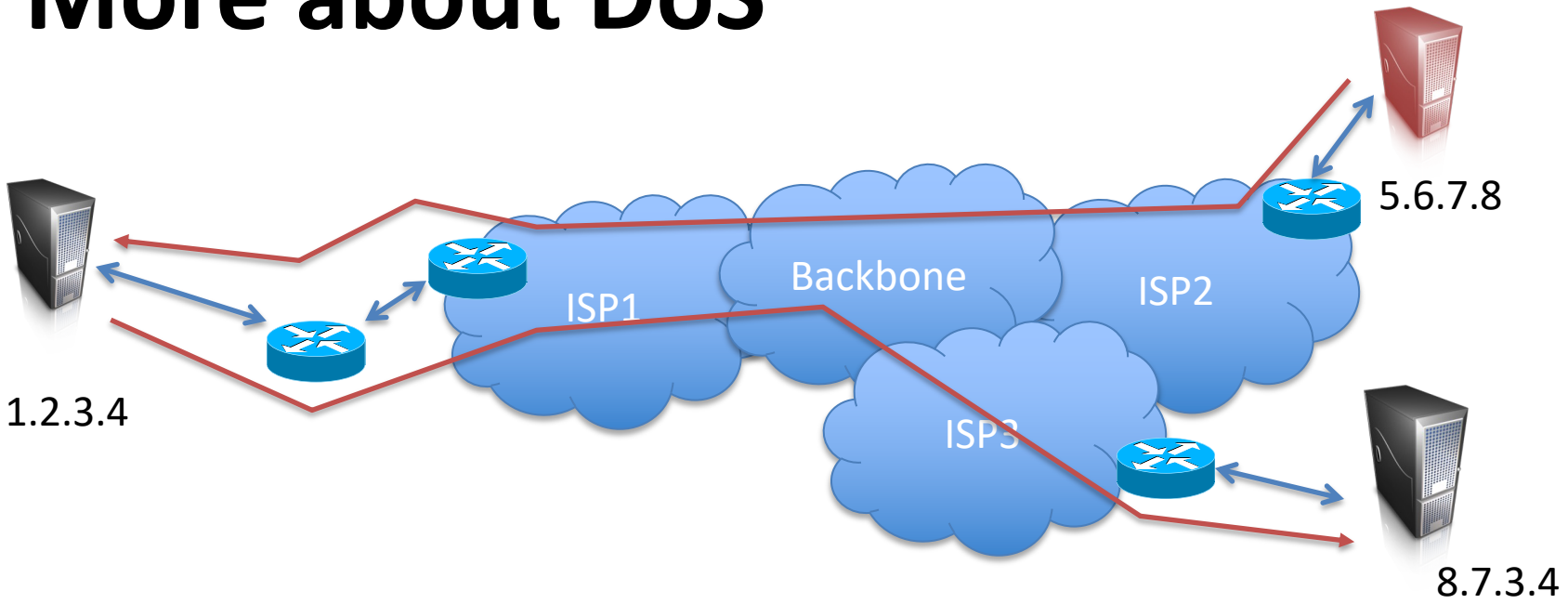
Many companies: Cloudflare, Imperva, Arbor Networks, …

Network security recap

- Internet protocols not designed with security in mind
  - No confidentiality, integrity mechanisms. No authentication
- On-path attackers
  - Traffic plaintext by default, can be observed & manipulated
- Off-path attackers can exploit protocol/implementation weaknesses to cause problems
  - Reflection attacks, UDP / TCP injection
- New security protocols tricky to deploy

DNSsec, BGPsec, IPsec all having slow uptake

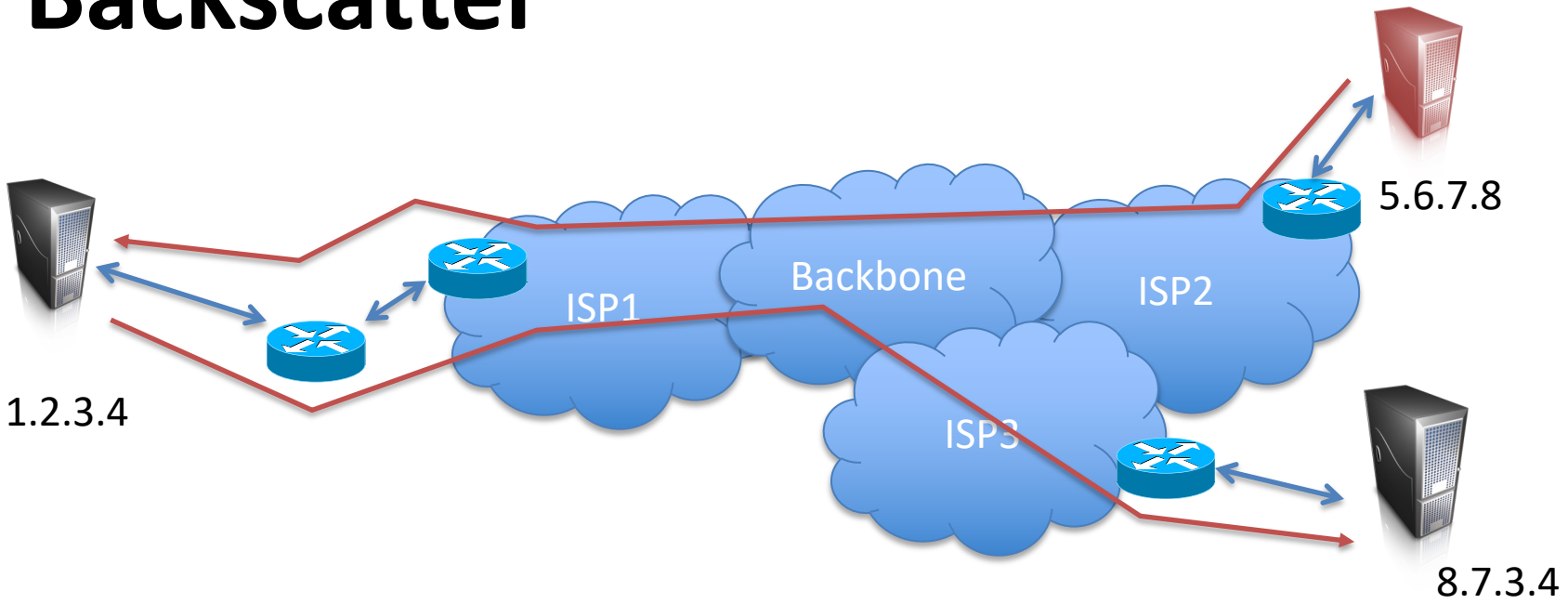
More about DoS



DoS is still a big problem

How big?

Backscatter



Can we measure the level of DoS attacks on Internet?

- If we can measure spurious packets at 8.7.3.4, we might infer something about DoS at 1.2.3.4

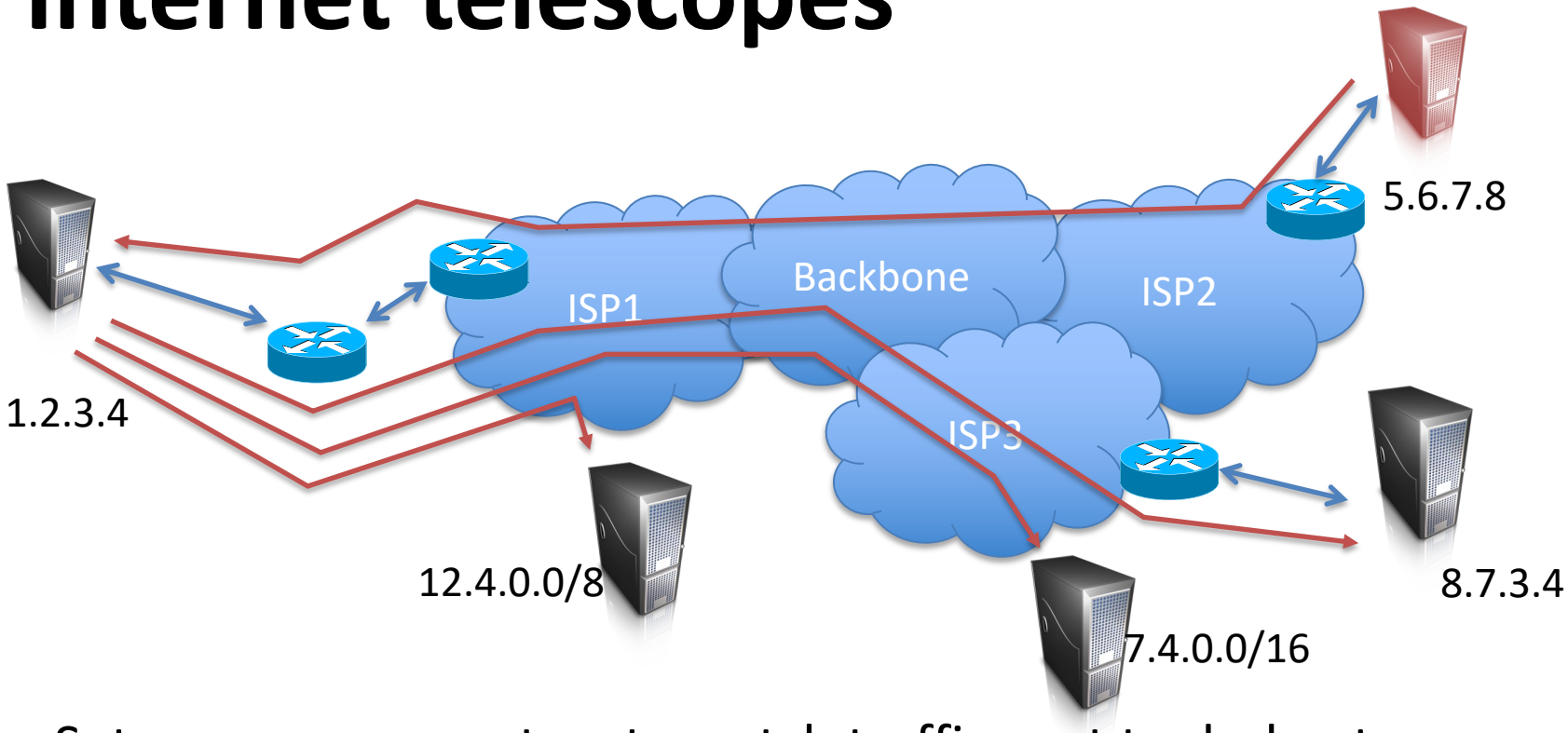
Types of responses to floods

Packet sent	Response from victim
TCP SYN (to open port)	TCP SYN/ACK
TCP SYN (to closed port)	TCP RST (ACK)
TCP ACK	TCP RST (ACK)
TCP DATA	TCP RST (ACK)
TCP RST	no response
TCP NULL	TCP RST (ACK)
ICMP ECHO Request	ICMP Echo Reply
ICMP TS Request	ICMP TS Reply
UDP pkt (to open port)	protocol dependent
UDP pkt (to closed port)	ICMP Port Unreach
...	...

Table 1: A sample of victim responses to typical attacks.

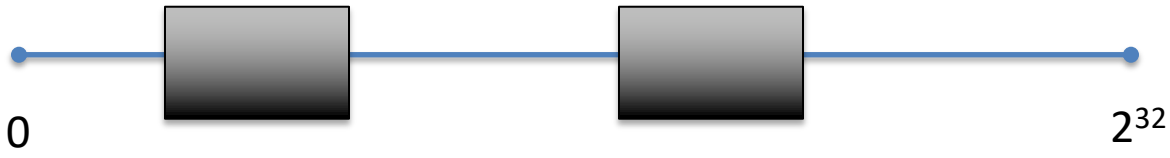
From Moore et al., “Inferring Internet Denial-of-Service Activity”

Internet telescopes



Setup some computers to watch traffic sent to darknets

- Darknet = unused routable space



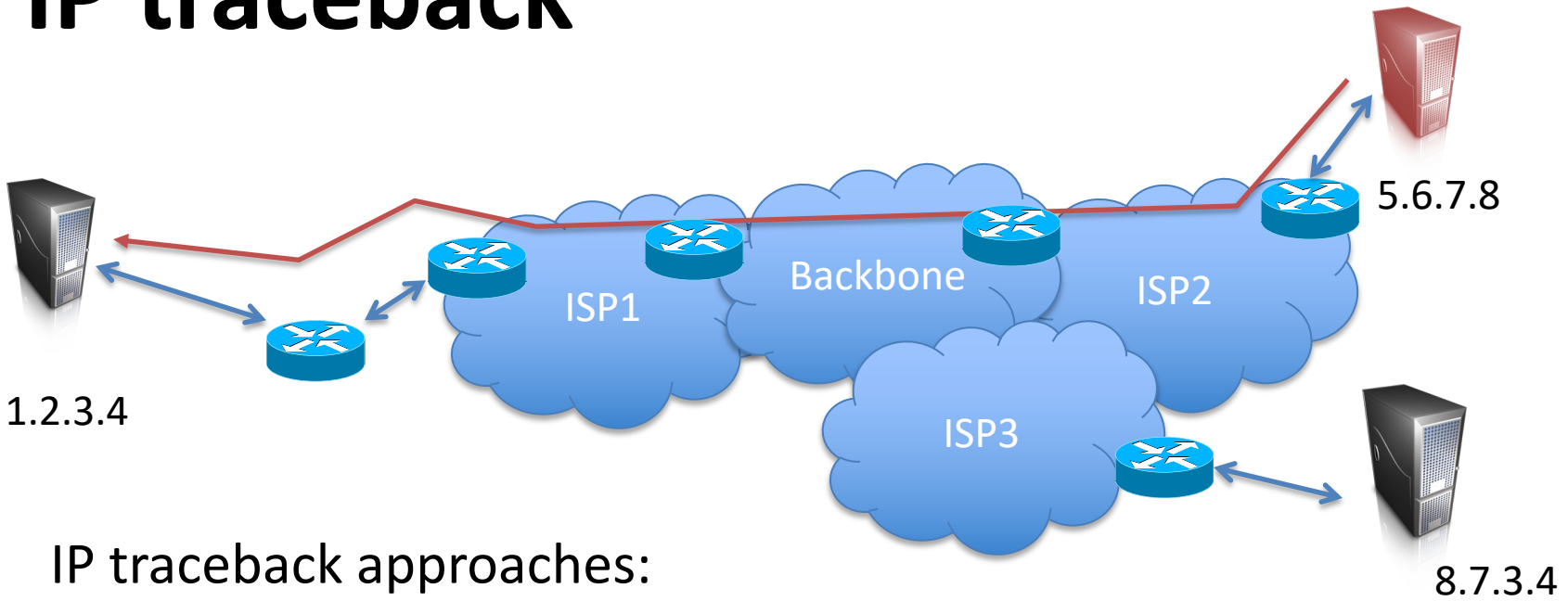
2001: 400 SYN attacks per week

2008: 4425 SYN attacks per 24 hours

Dealing with spoofing: IP traceback

- Spoofed IPs means we cannot know where packets came from
- IP traceback is problem of determining the origination of one or more packets

IP traceback



IP traceback approaches:

- Logging – each router keeps logs of packets going by
- Input debugging – feature of routers allowing filtering egress port traffic based on ingress port. Associate egress with ingress
- Controlled flooding – mount your own DoS on links selectively to see how it affects malicious flood
- Marking – router probabilistically marks packets with info
- ICMP traceback – router probabilistically sends ICMP packet with info to destination

IPv4 fragmenting

Ethernet frame containing IP datagram

IP allows datagrams of size from

20 bytes up to 65535 bytes

Some link layers only allow MTU of 1500 bytes

IP figures out MTU of next link, and fragments packet if necessary into smaller chunk

IPv4 fragmenting

Ethernet frame containing IP datagram

4-bit version	4-bit hdr len	8-bit type of service	16-bit total length (in bytes)	
16-bit identification			3-bit flags	13-bit fragmentation offset
8-bit time to live (TTL)		8-bit protocol	16-bit header checksum	
32-bit source IP address				
32-bit destination IP address				
options (optional)				

IPv4 fragmenting

Ethernet frame containing IP datagram

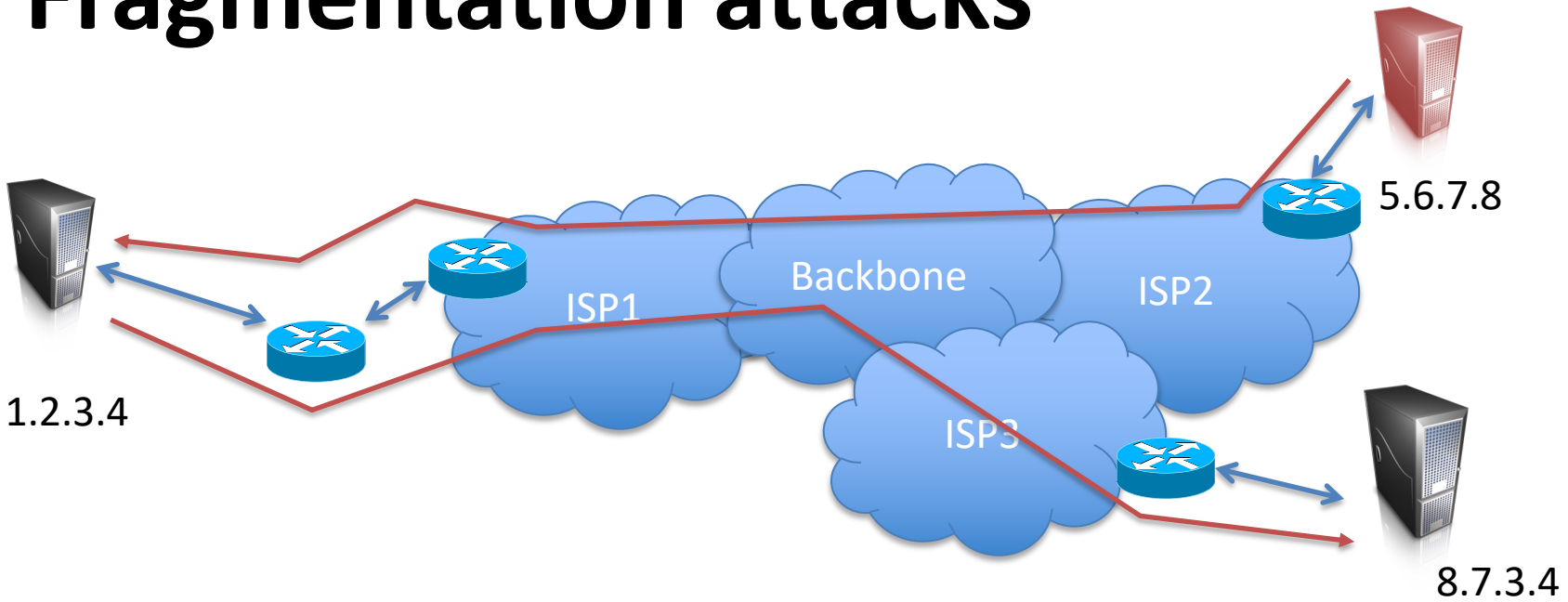
Source-specified “unique” number identifying datagram

Fragment offset in 8-byte units

Flags:
0 b1 b2

where b1 = May Fragment (0) / Don’t Fragment (1)
where b1 = Last Fragment (0) / More Fragments (1)

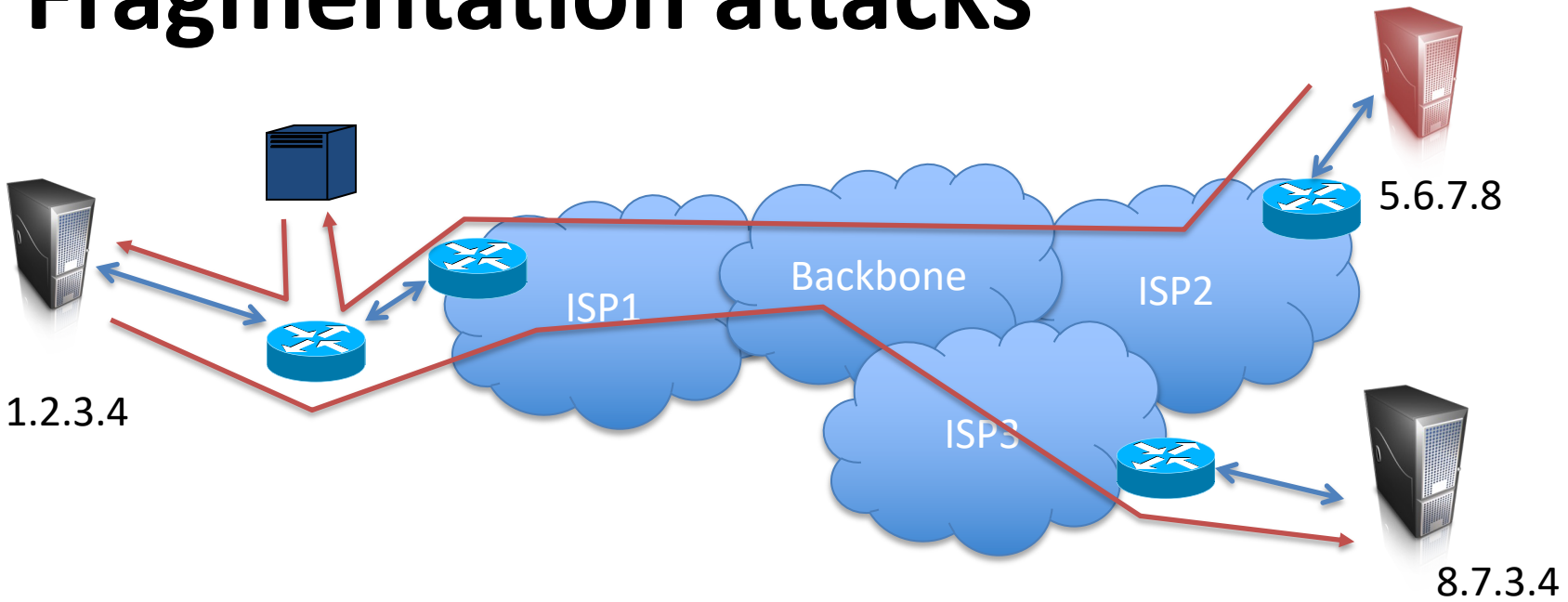
Fragmentation attacks



Fragmentation abused in lots of vulnerabilities:

- Ping of death: allows sending 65,536 byte packet, overflows buffer.
- Teardrop DoS: mangled fragmentation crashes reconstruction code (Set offsets so that two packets have overlapping data)

Fragmentation attacks



Fragmentation abused in lots of vulnerabilities:

- Ping of death: allows sending 65,536 byte packet, overflows buffer.
- Teardrop DoS: mangled fragmentation crashes reconstruction code (Set offsets so that two packets have overlapping data)
- Avoiding IDS systems: IDS scans packets for exploit strings; add random data into packets, overwrite later during reconstruction due to overlapping fragments