TCP/IP security

CS642: Computer Security

Professor Ristenpart

http://www.cs.wisc.edu/~rist/

rist at cs dot wisc dot edu

Moving up the network stack

Internet protocol and ICMP

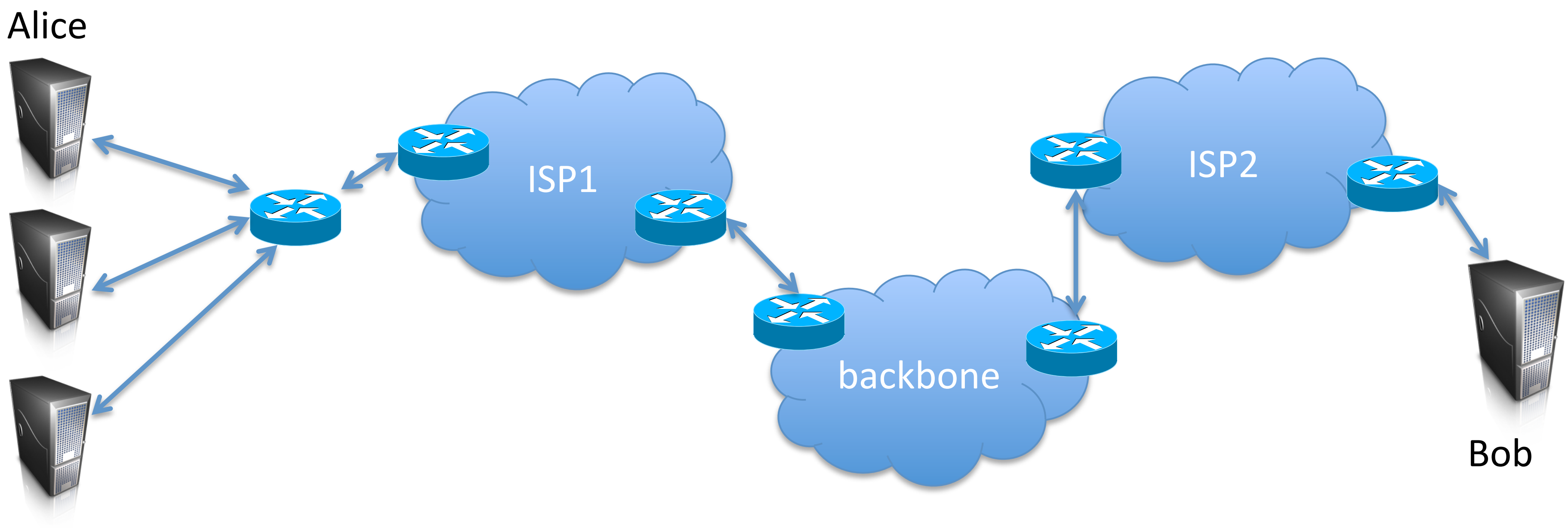
IP spoofing, fragmentation

TCP

Denial of Service

IP traceback, filtering

Internet



Local area network (LAN)

Ethernet

802.11

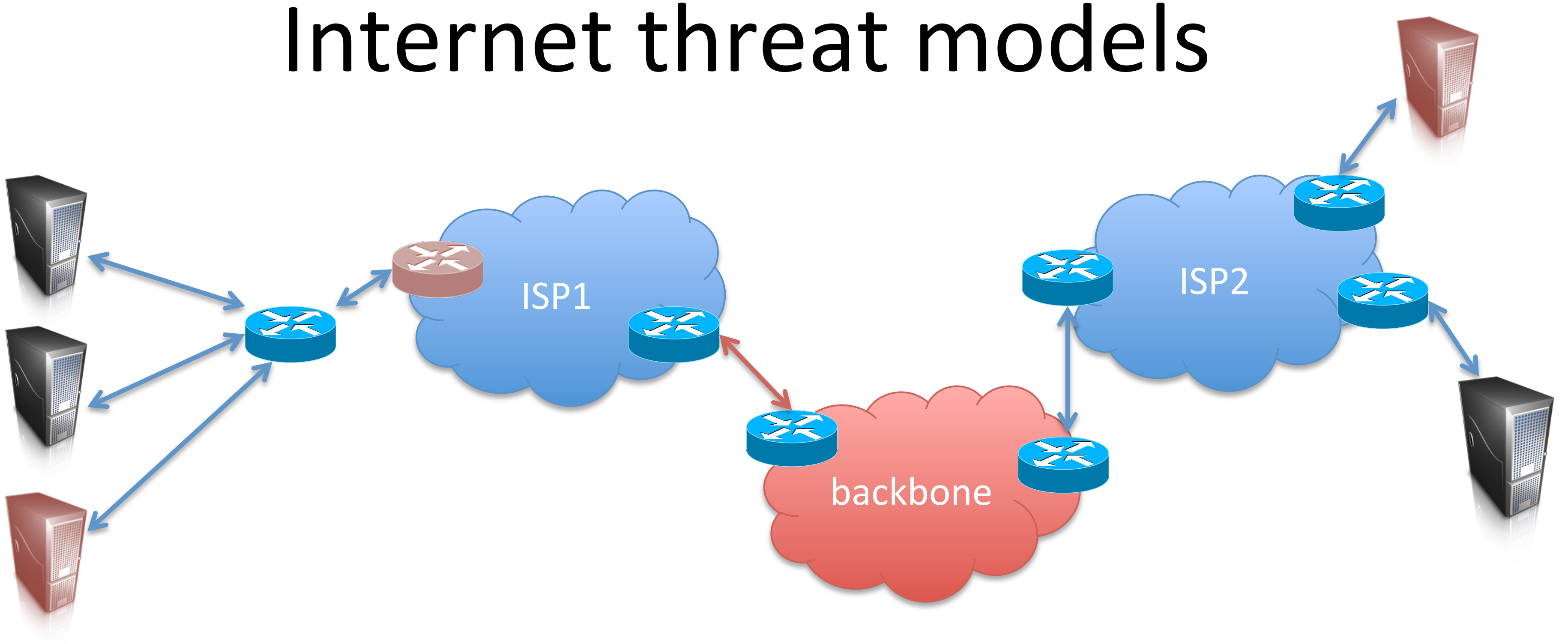
Internet

TCP/IP

BGP (border gateway protocol)

DNS (domain name system)

Internet threat models



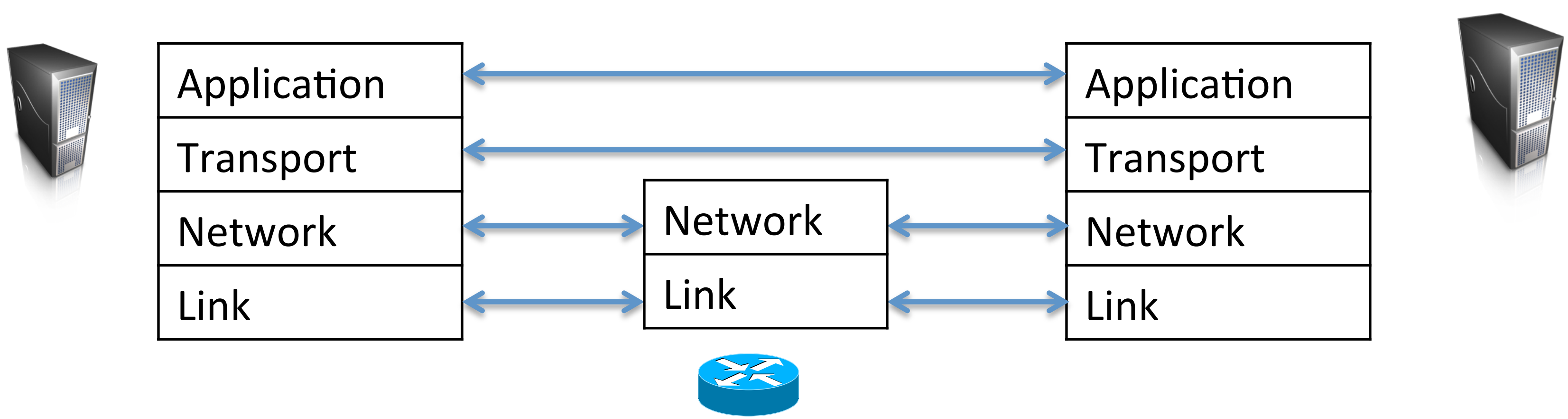
(1) Malicious hosts

(2) Subverted routers or links

(3) Malicious ISPs or backbone

Internet protocol stack

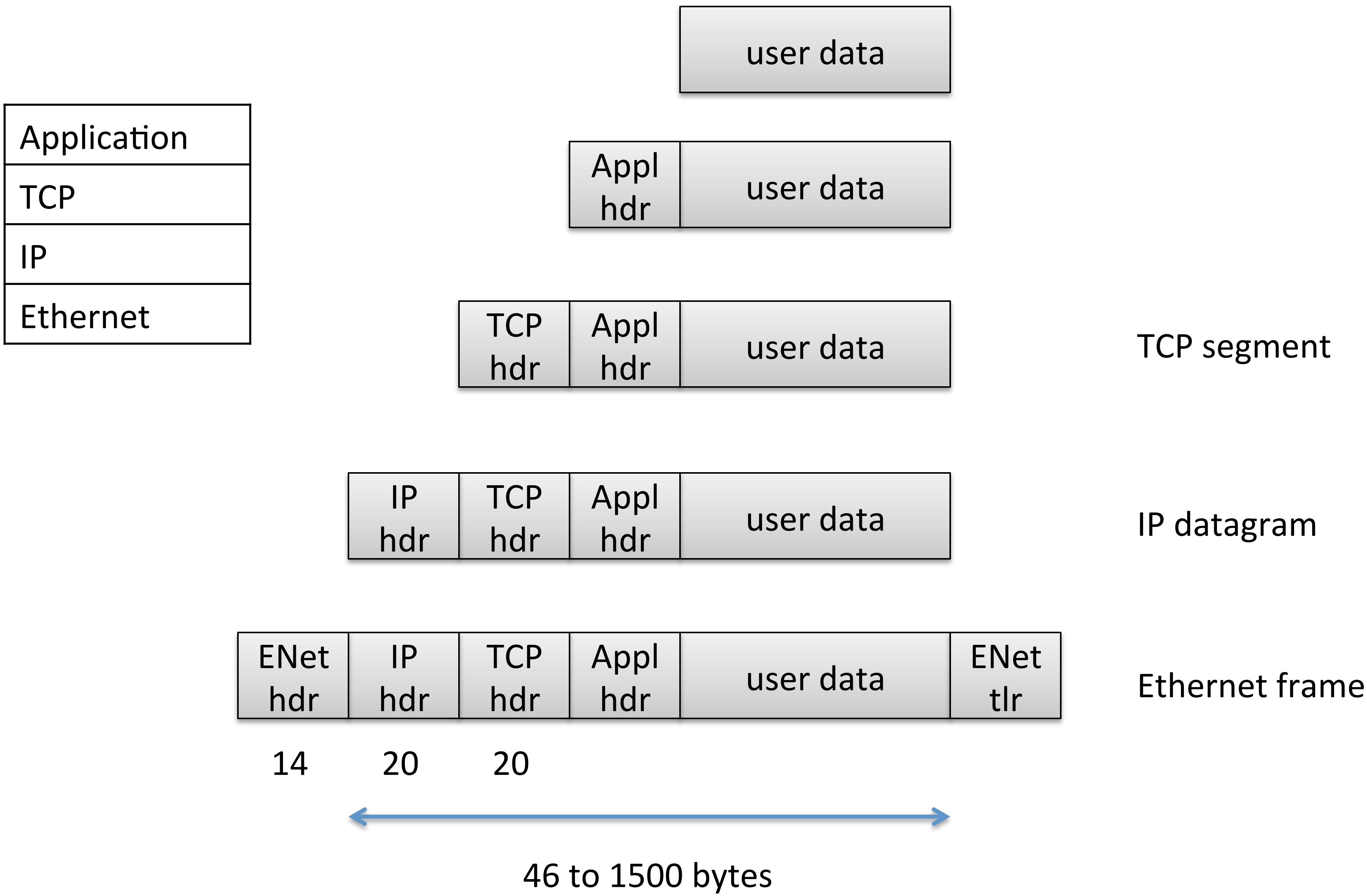
Application	HTTP, FTP, SMTP, SSH, etc.
Transport	TCP, UDP
Network	IP, ICMP, IGMP
Link	802x (802.11, Ethernet)



IP protocol (IPv4)

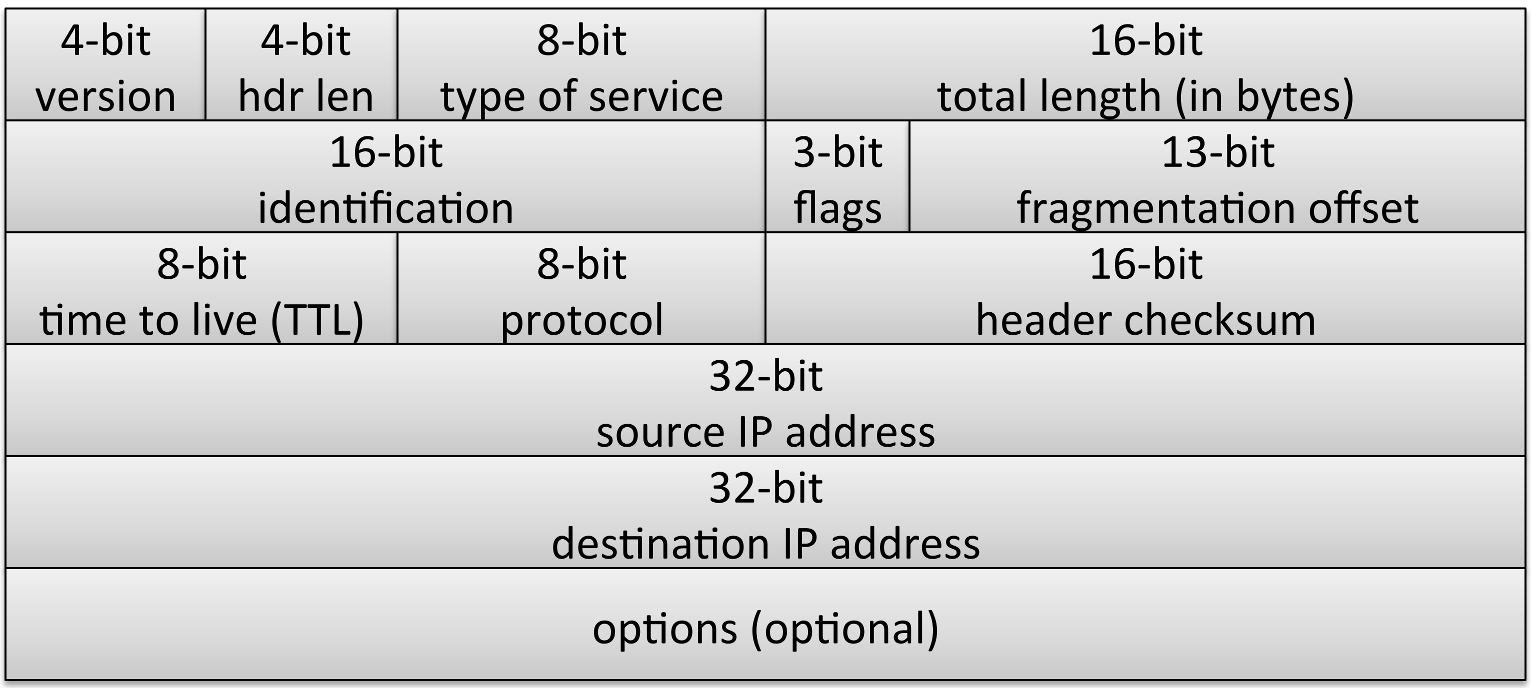
- Connectionless
  - no state
- Unreliable
  - no guarantees
- ICMP (Internet Control Message Protocol)
  - error messages, etc.
  - often used by tools such as ping, traceroute

Internet protocol stack



IPv4

Ethernet frame containing IP datagram



Classless Inter-Domain routing (CIDR)

128.168.0.0/16

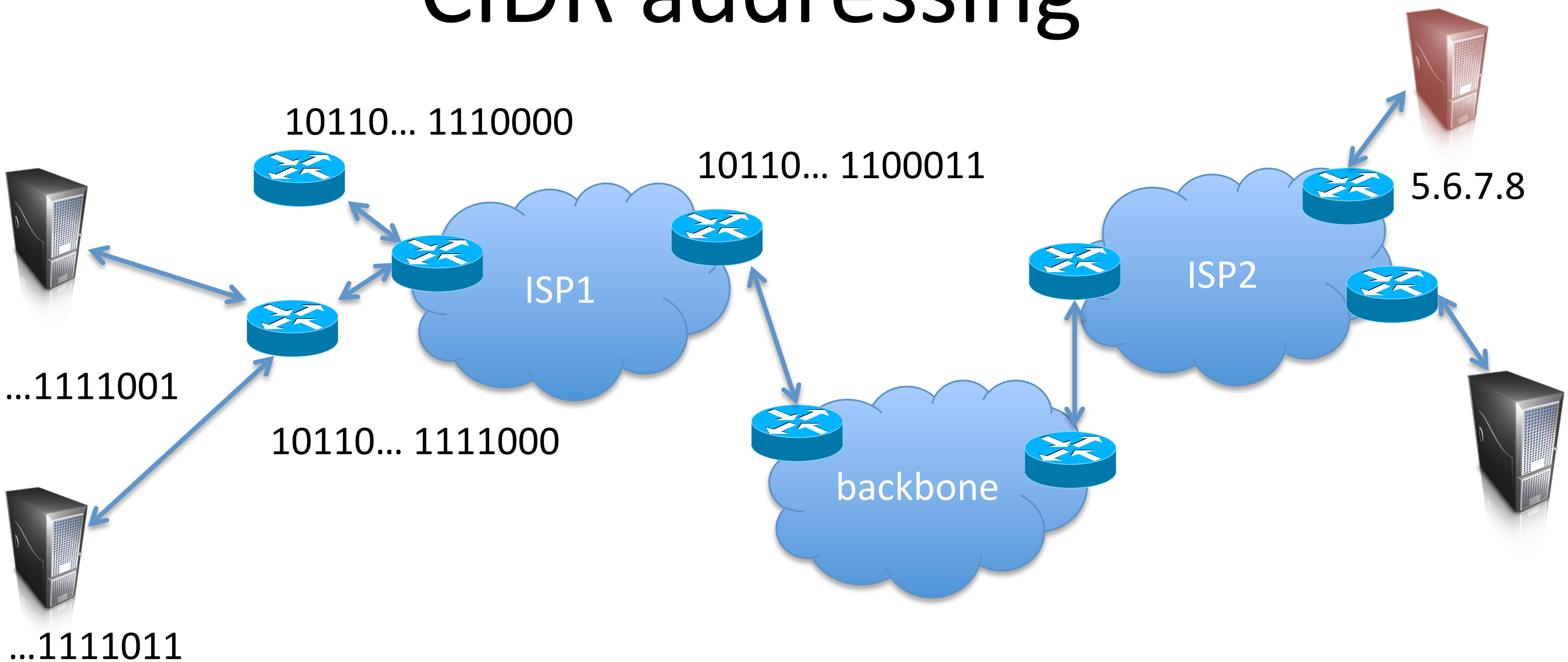
a.b.c.d / x

x indicates number of bits used for a routing prefix
IP addresses with same /x prefix share some portion of route

CIDR address block	Description	Reference
0.0.0.0/8	Current network (only valid as source address)	RFC 1700
10.0.0.0/8	Private network	RFC 1918
127.0.0.0/8	Loopback	RFC 5735
169.254.0.0/16	Link-Local	RFC 3927
172.16.0.0/12	Private network	RFC 1918
192.0.0.0/24	Reserved (IANA)	RFC 5735
192.0.2.0/24	TEST-NET-1, Documentation and example code	RFC 5735
192.88.99.0/24	IPv6 to IPv4 relay	RFC 3068
192.168.0.0/16	Private network	RFC 1918
198.18.0.0/15	Network benchmark tests	RFC 2544
198.51.100.0/24	TEST-NET-2, Documentation and examples	RFC 5737
203.0.113.0/24	TEST-NET-3, Documentation and examples	RFC 5737
224.0.0.0/4	Multicasts (former Class D network)	RFC 3171
240.0.0.0/4	Reserved (former Class E network)	RFC 1700
255.255.255.255	Broadcast	RFC 919

From http://en.wikipedia.org/wiki/IPv4

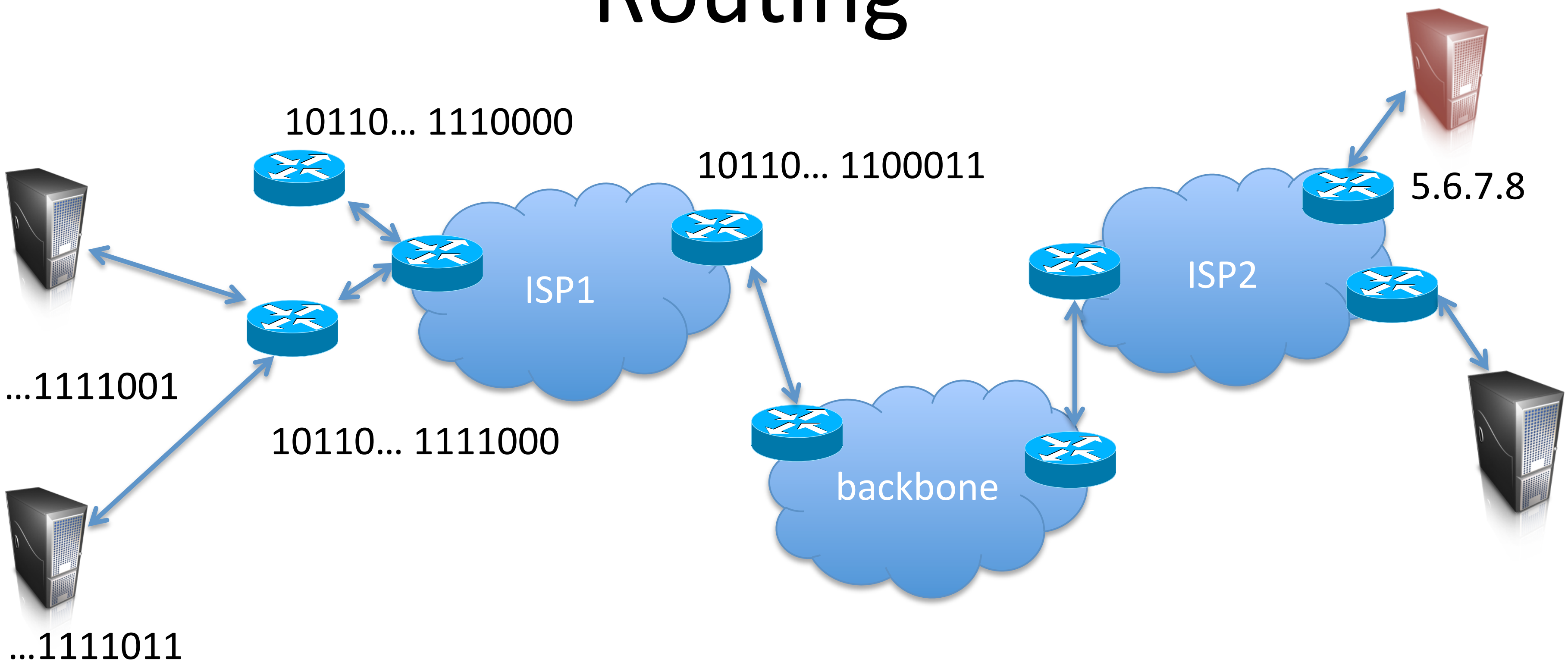
CIDR addressing



Prefixes used to setup hierarchical routing:

- An organization assigned   a.b.c.d/x
- It manages addresses prefixed by a.b.c.d/x

Routing



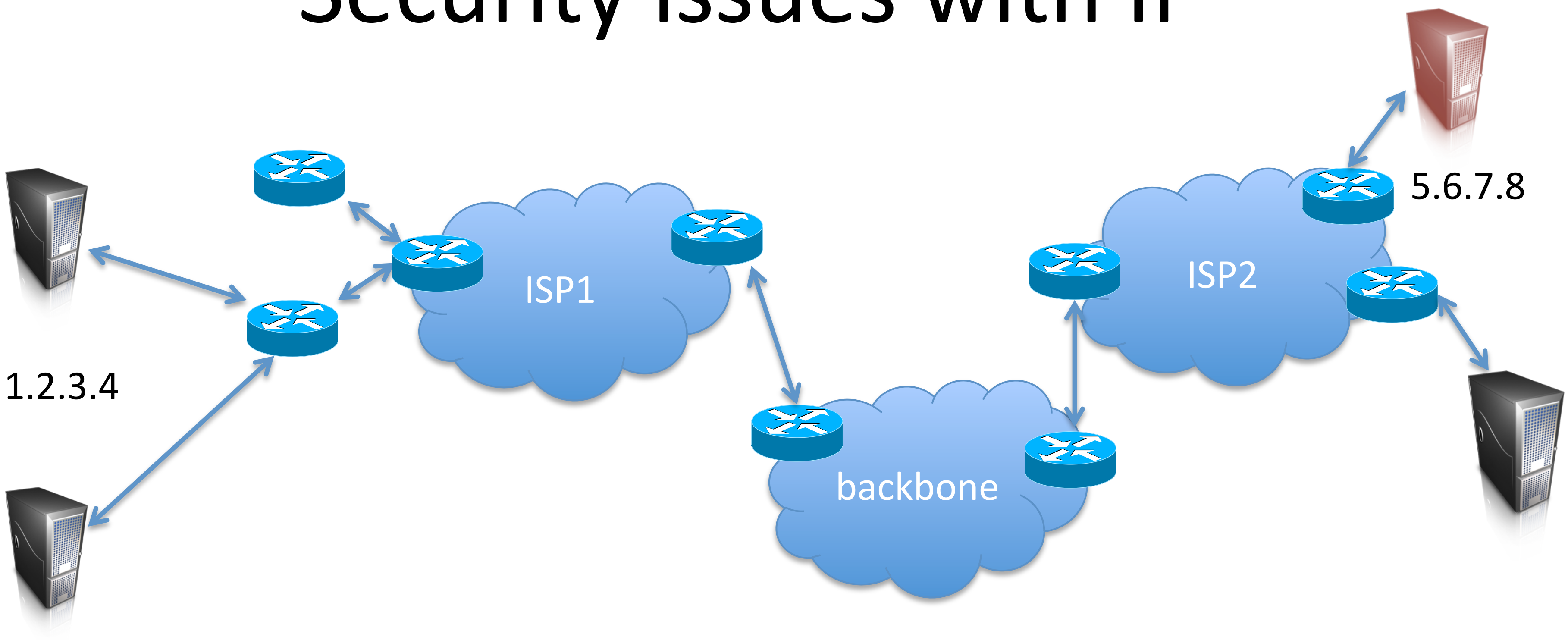
Autonomous systems (AS) are organizational building blocks

- Collection of IP prefixes under single routing policy
- wisc.edu

Within AS, might use RIP (Routing Information Protocol)

Between AS, use BGP (Border Gateway Protocol)

Security issues with IP

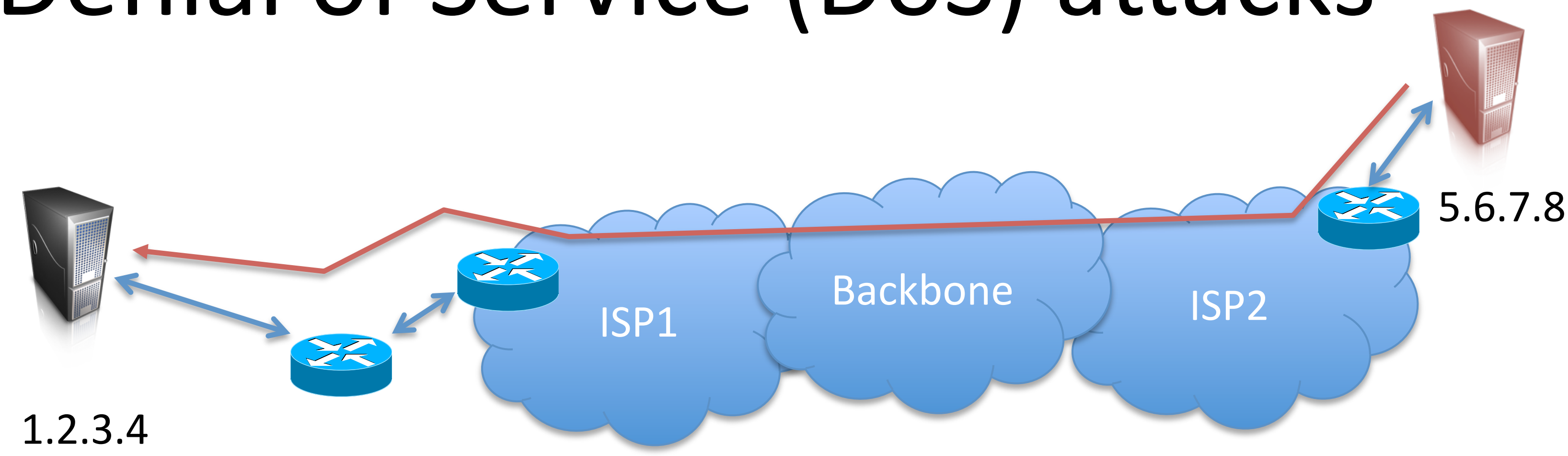


Routing has issues, we'll get to that later

What else?

- Anyone can talk to anyone
- No source address authentication in general

Denial of Service (DoS) attacks



Goal: prevent legitimate users from accessing victim (1.2.3.4)

ICMP ping flood

ICMP
(Internet Control Message Protocol)

IP hdr	ICMP hdr	ICMP message

8-bit type	8-bit code	16-bit checksum
4-byte more of header (depends on type)		
message …		

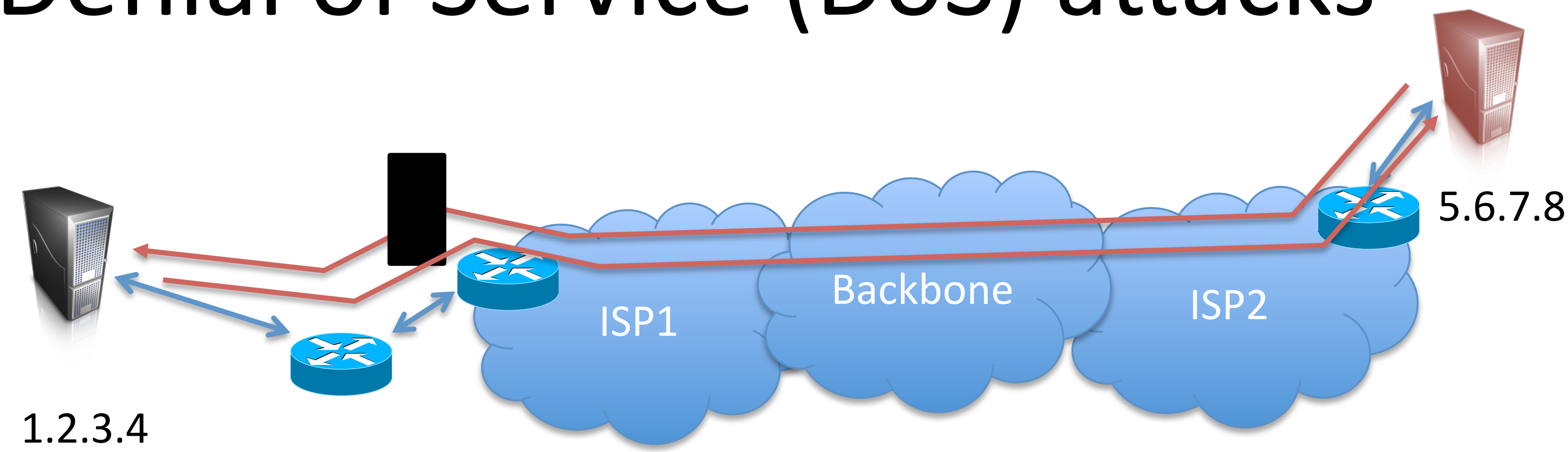
ICMP
(Internet Control Message Protocol)

IP hdr	ICMP hdr	ICMP message

8-bit type (0 or 8)	8-bit code = 0	16-bit checksum
16-bit identifier		16-bit sequence number
optional data		

Echo request (used by ping)

Denial of Service (DoS) attacks

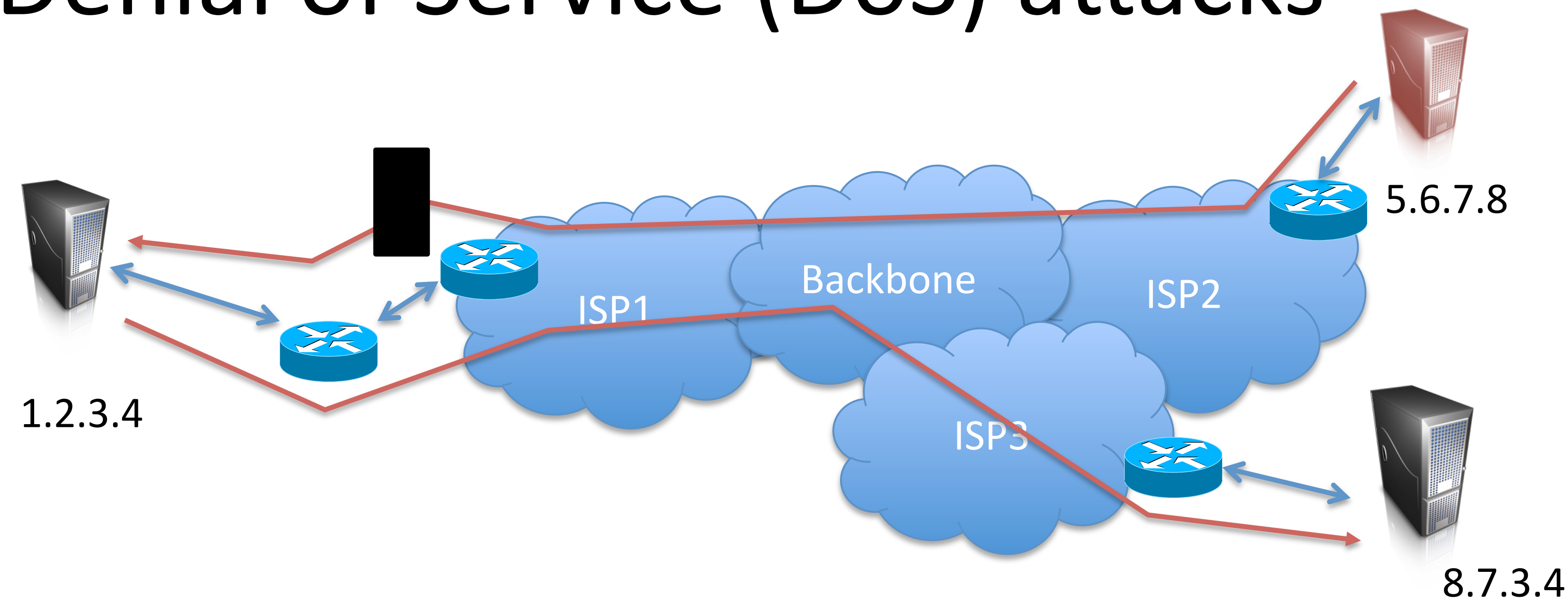


Goal is to prevent legitimate users from accessing victim (1.2.3.4)

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 near victim

Denial of Service (DoS) attacks



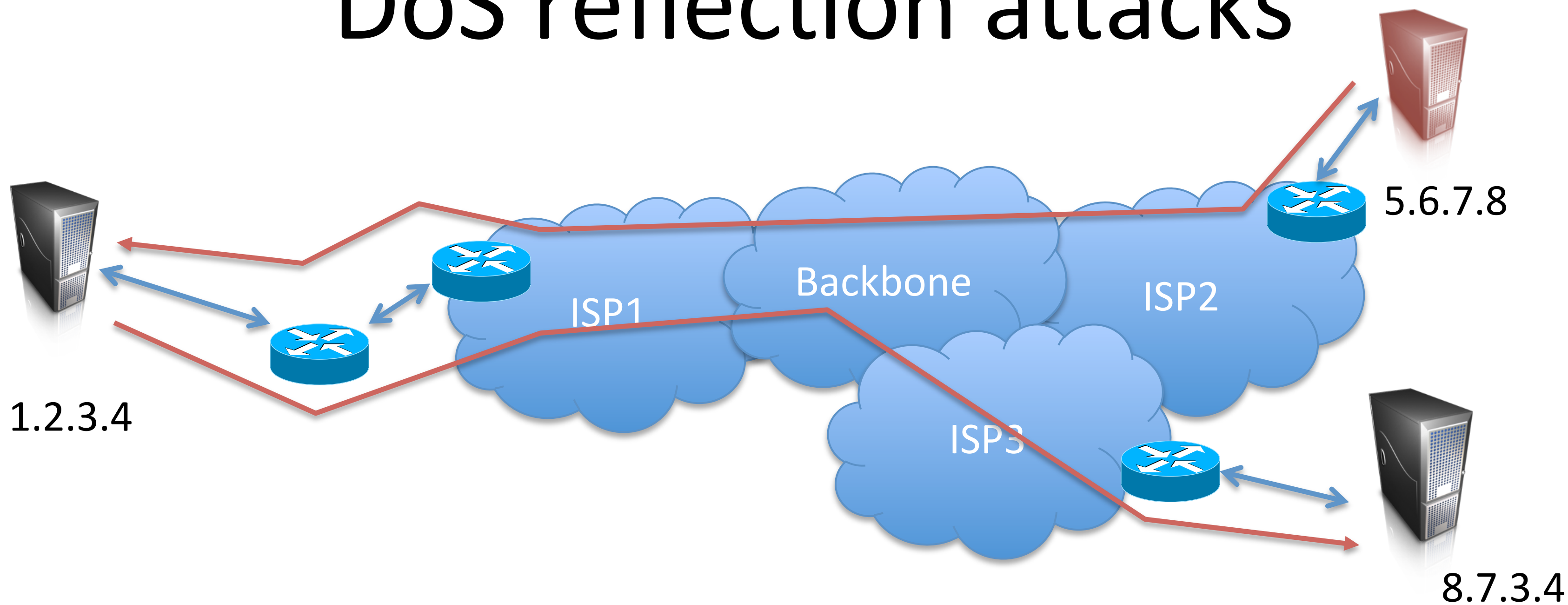
How can attacker avoid ingress filtering?

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

Filter based on source may be incorrect

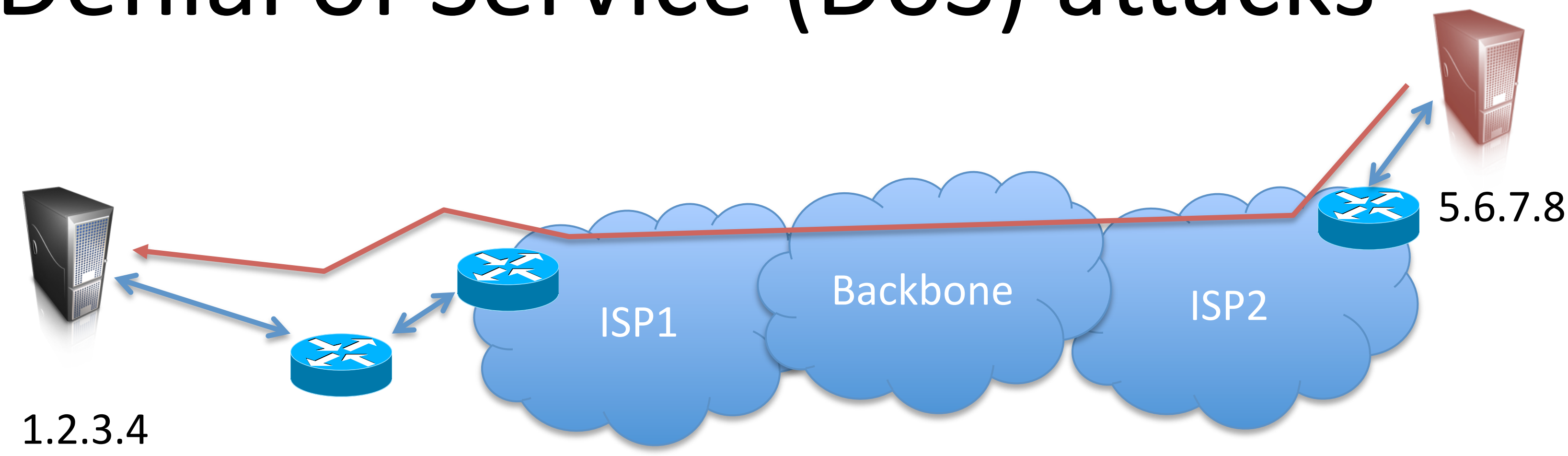
DoS reflection attacks



Note a valid packet sends a reply to 8.7.3.4

- Attacker can bounce an attack against 8.7.3.4 off 1.2.3.4
- "Frame" 1.2.3.4
- Single-packet exploit (1.2.3.4 in foreign country)

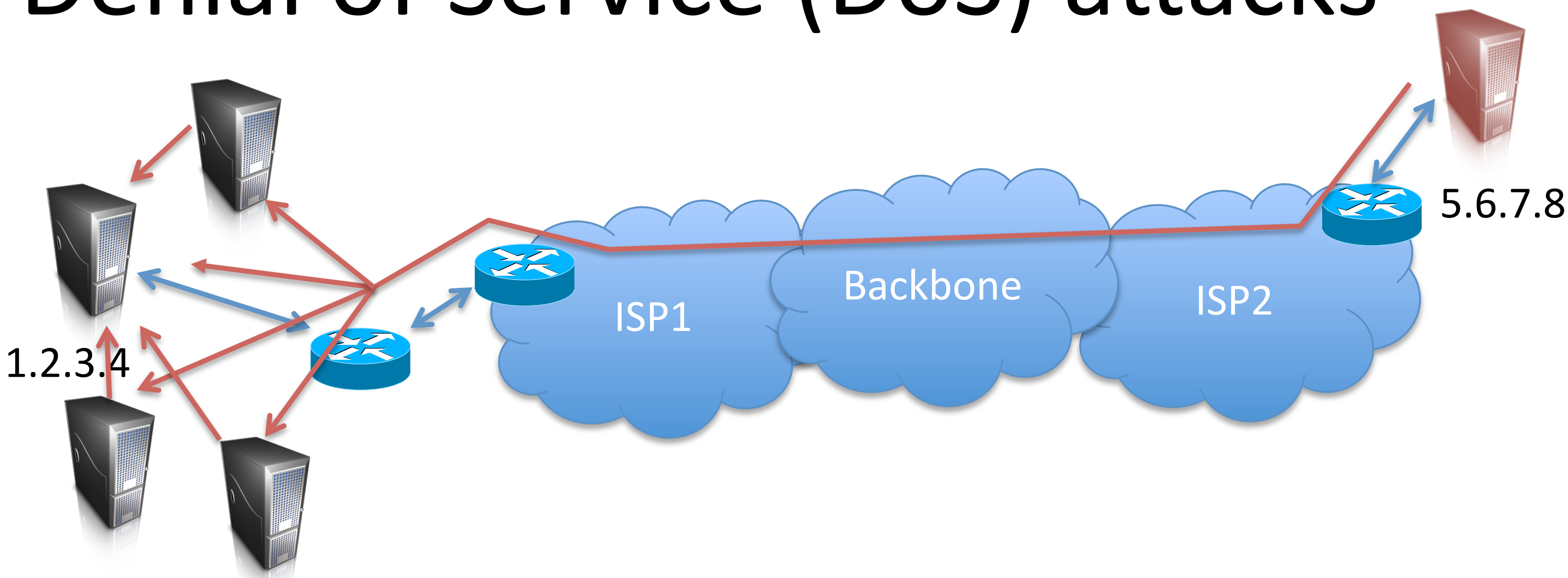
Denial of Service (DoS) attacks



DoS works better when there is ***asymmetry*** between victim and attacker

- Attacker uses few resources to cause
- Victim to consume lots of resources

Denial of Service (DoS) attacks



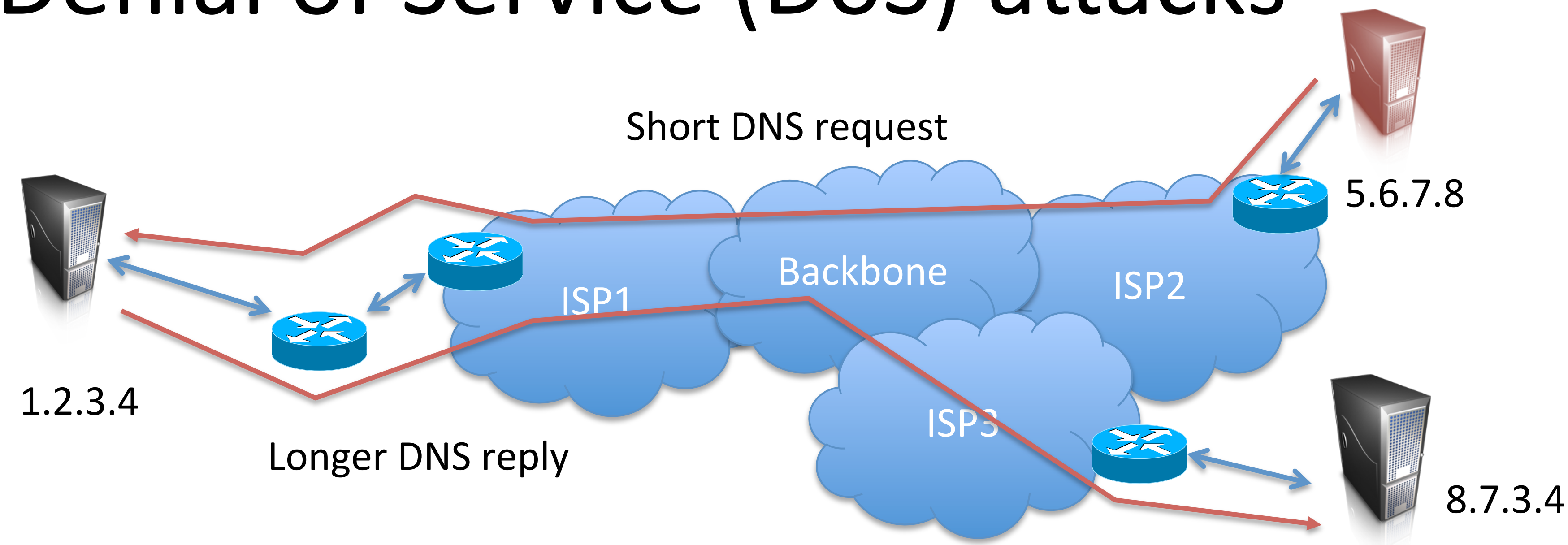
DoS works better when there is ***asymmetry*** between victim and attacker

- Attacker uses few resources to cause
- Victim to consume lots of resources

Old example: Smurf attack

Router allows attacker to send broadcast ICMP ping on network. Attacker spoofs SRC address to be 1.2.3.4

Denial of Service (DoS) attacks



DoS works better when there is ***asymmetry*** between victim and attacker

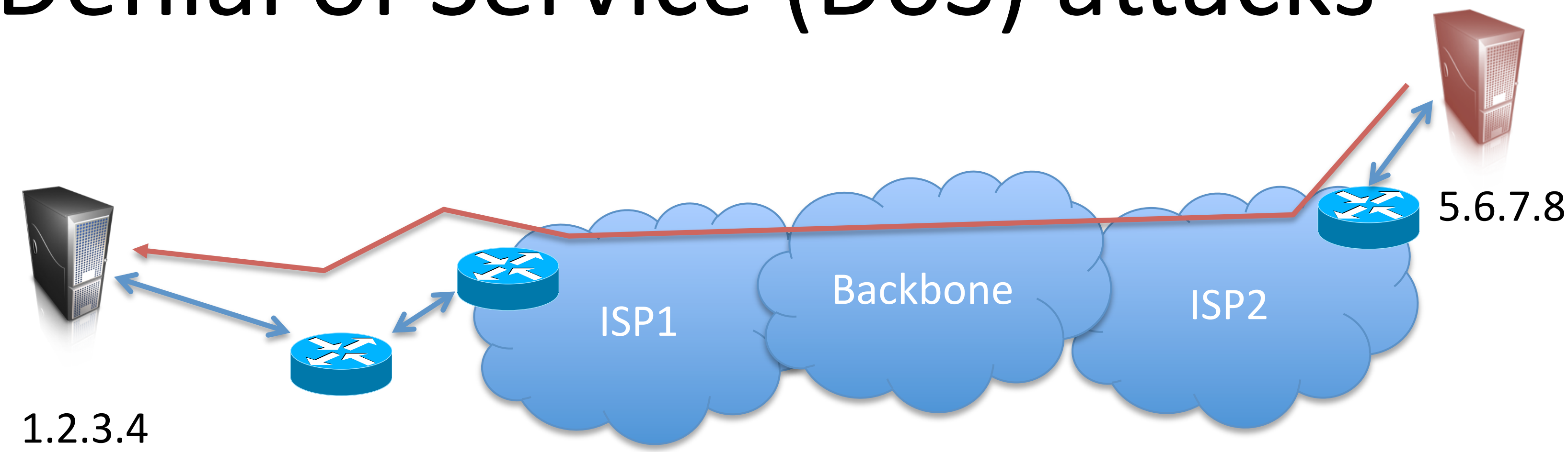
- Attacker uses few resources to cause
- Victim to consume lots of resources

More recent: DNS reflection attacks

Send DNS request w/ spoofed target IP (~65 byte request)

DNS replies sent to target (~512 byte response)

Denial of Service (DoS) attacks



DoS works better when there is ***asymmetry*** between victim and attacker

- Attacker uses few resources to cause
- Victim to consume lots of resources

Big asymmetry: ping of death

A single packet that causes crash on remote system

Early on: ping packet with size > 65,535

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

ENet hdr	IP hdr	data	ENet tlr

Ethernet frame containing IP datagram

<table>
<tr><td>4-bit
version</td><td>4-bit
hdr len</td><td>8-bit
type of service</td><td colspan="2">16-bit
total length (in bytes)</td></tr>
<tr><td colspan="3">16-bit
identification</td><td>3-bit
flags</td><td>13-bit
fragmentation offset</td></tr>
<tr><td colspan="2">8-bit
time to live (TTL)</td><td>8-bit
protocol</td><td colspan="2">16-bit
header checksum</td></tr>
<tr><td colspan="5">32-bit
source IP address</td></tr>
<tr><td colspan="5">32-bit
destination IP address</td></tr>
<tr><td colspan="5">options (optional)</td></tr>
</table>

IPv4 fragmenting

Ethernet frame containing IP datagram

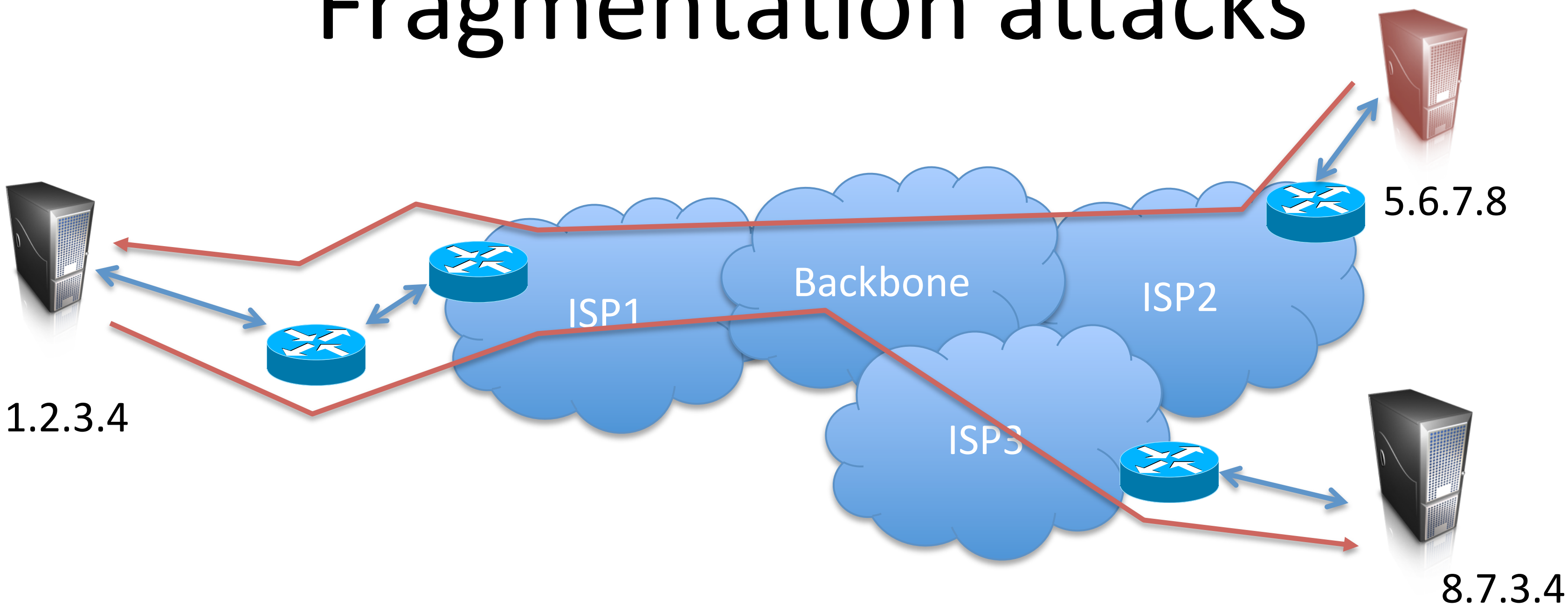
Source-specified “unique” number identifying datagram

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)

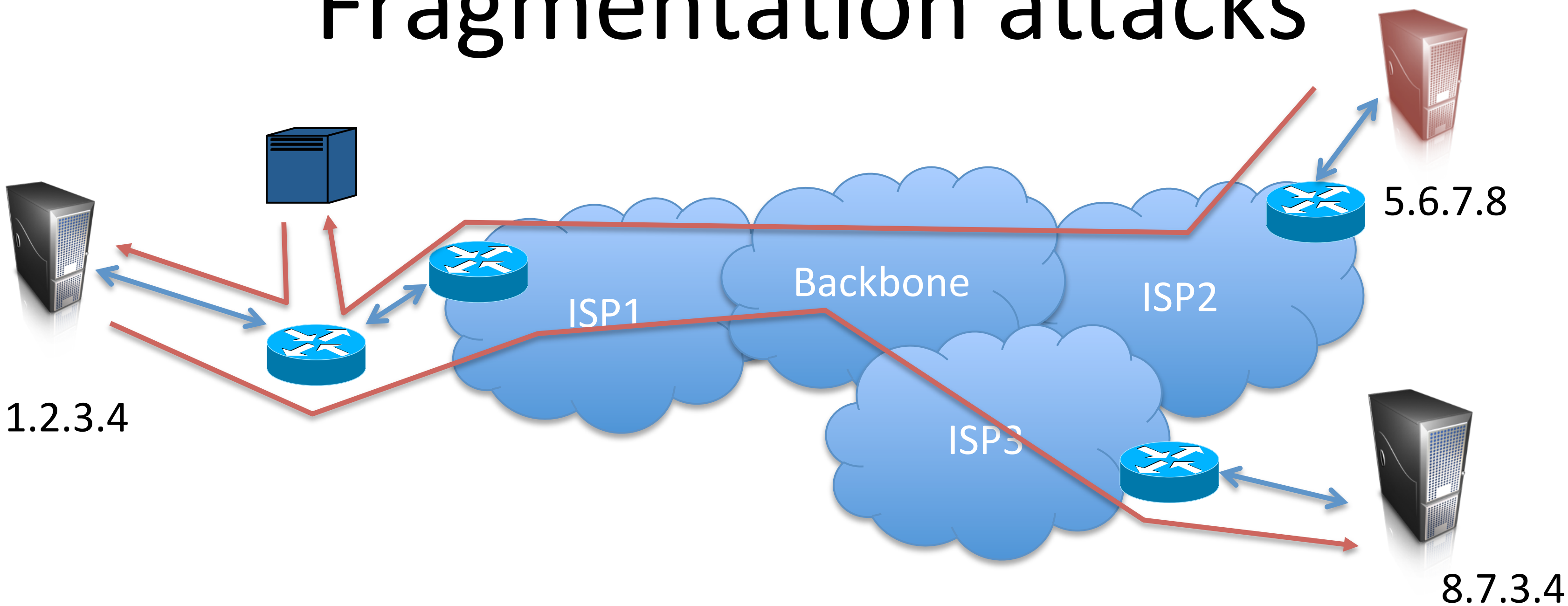
Fragmentation attacks



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



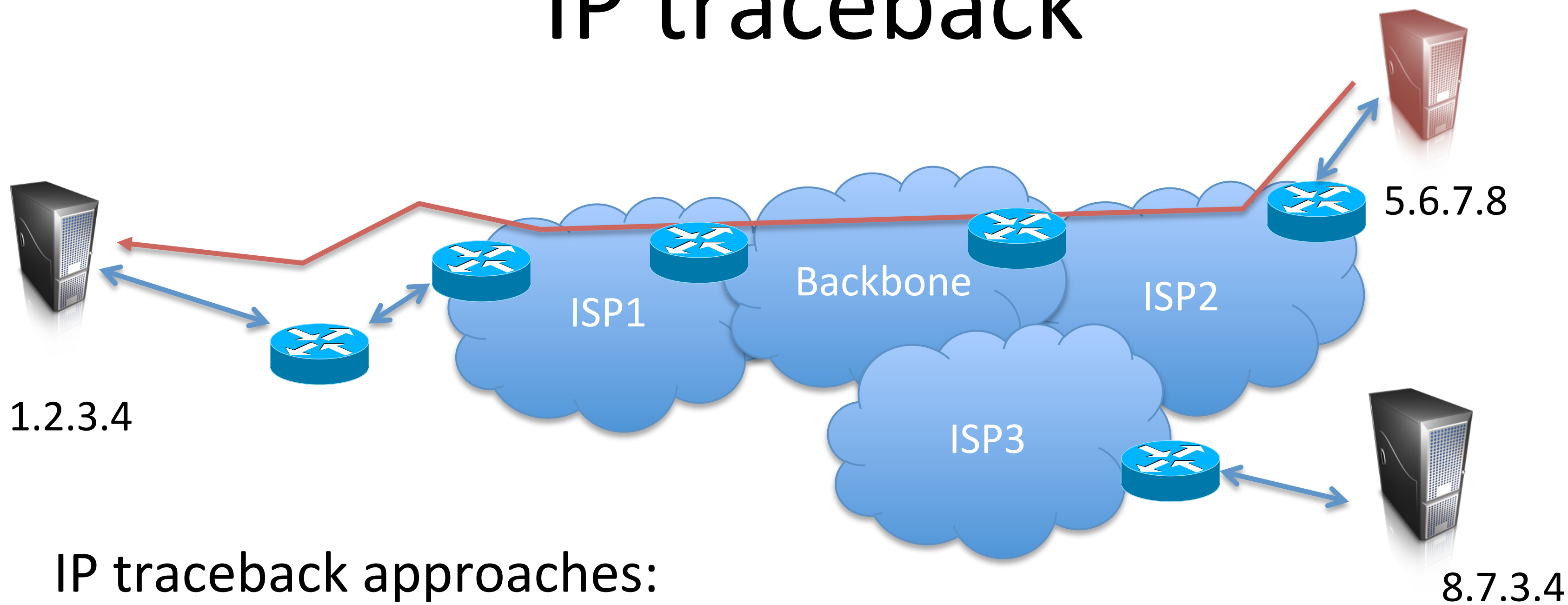
Fragmentation abused in lots of vulnerabilities:

- 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

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



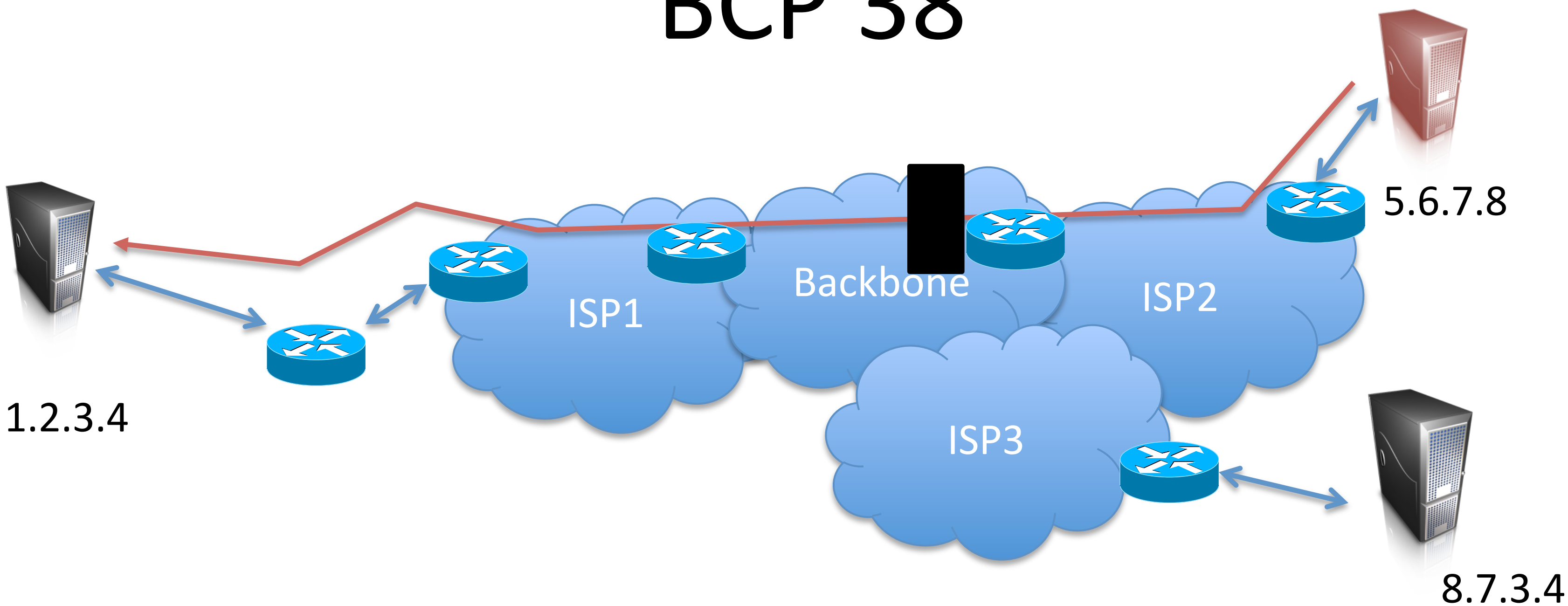
IP traceback approaches:

- 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

Dealing with spoofing: BCP 38

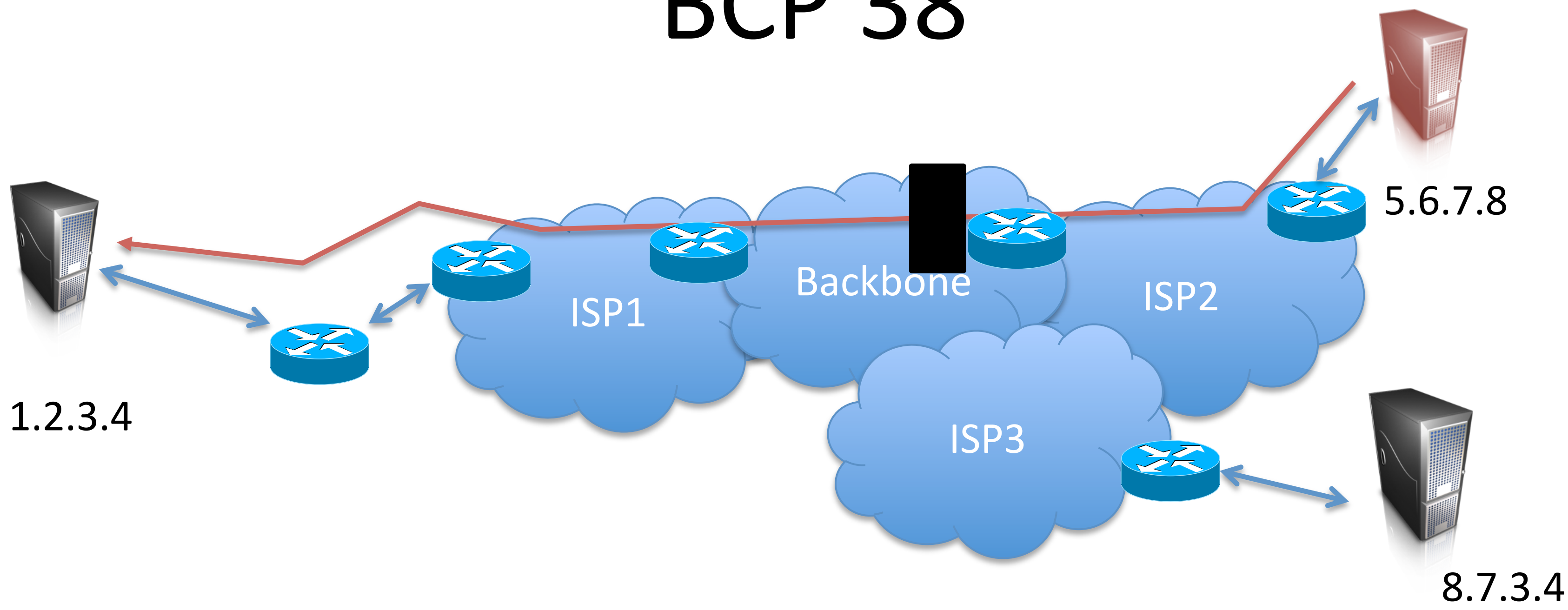
- Spoofed IPs means we cannot know where packets came from
- BCP 38 (RFC 2827): upstream ingress filtering to drop spoofed packets

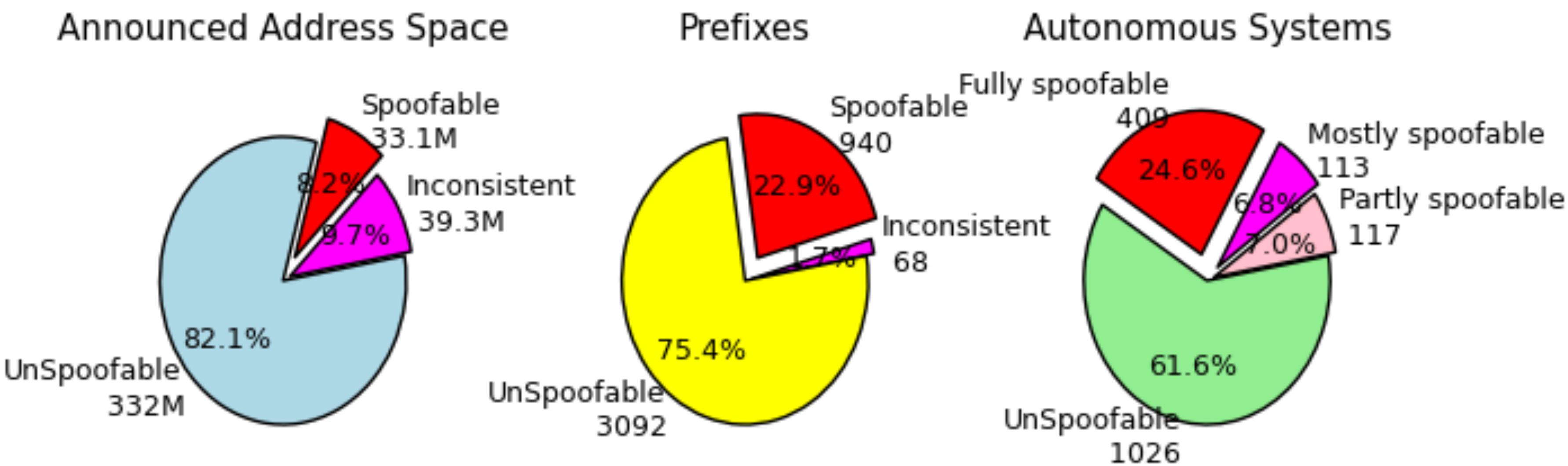
BCP 38



Before forwarding on packets, check at ingress that source IP legitimate

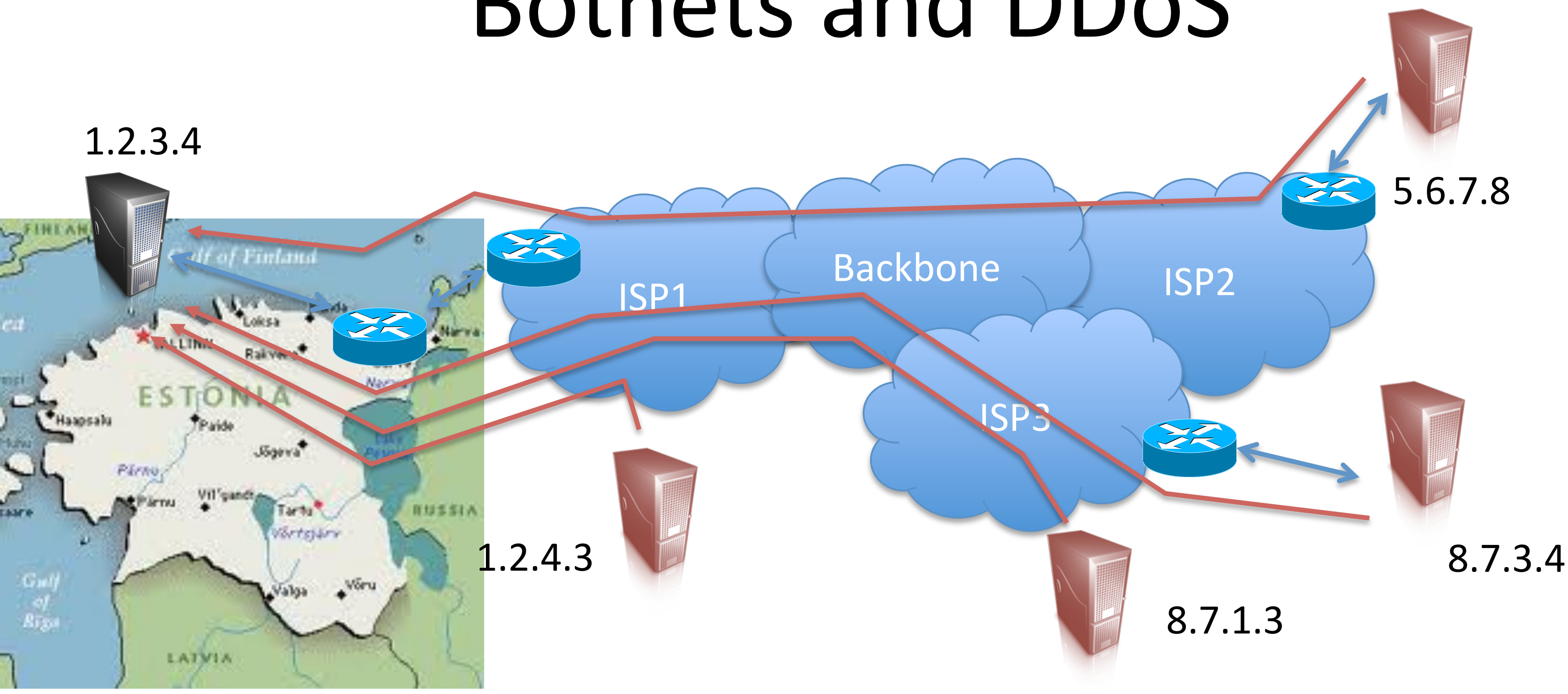
BCP 38





http://spoofer.cmand.org/summary.php

Botnets and DDoS



April 27, 2007

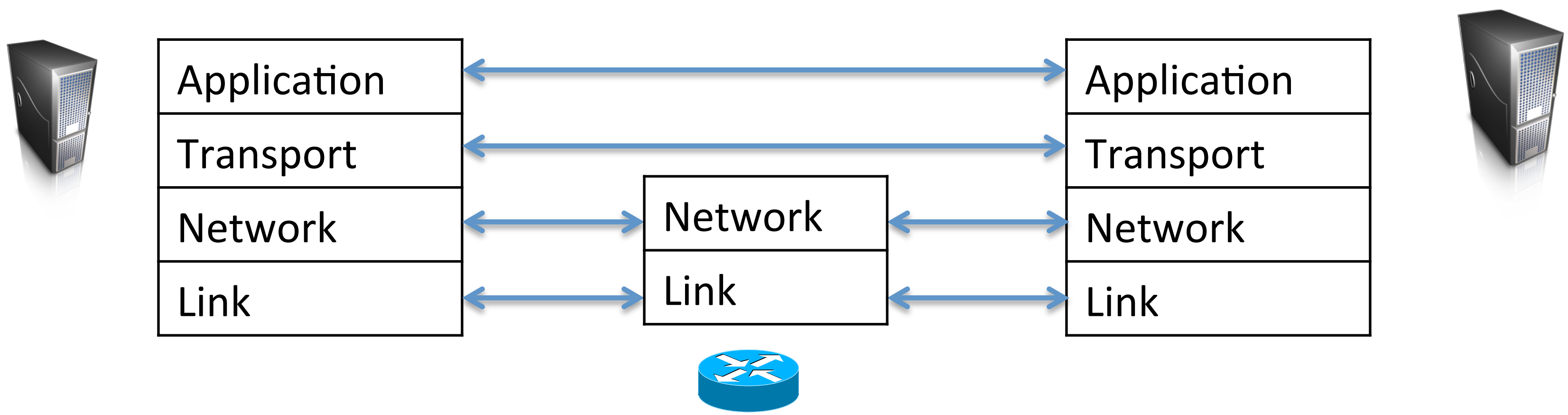
Continued for weeks, with varying levels of intensity

Government, banking, news, university websites

Government shut down international Internet connections

Internet protocol stack

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



UDP (user datagram protocol)

IP hdr	UDP hdr	data

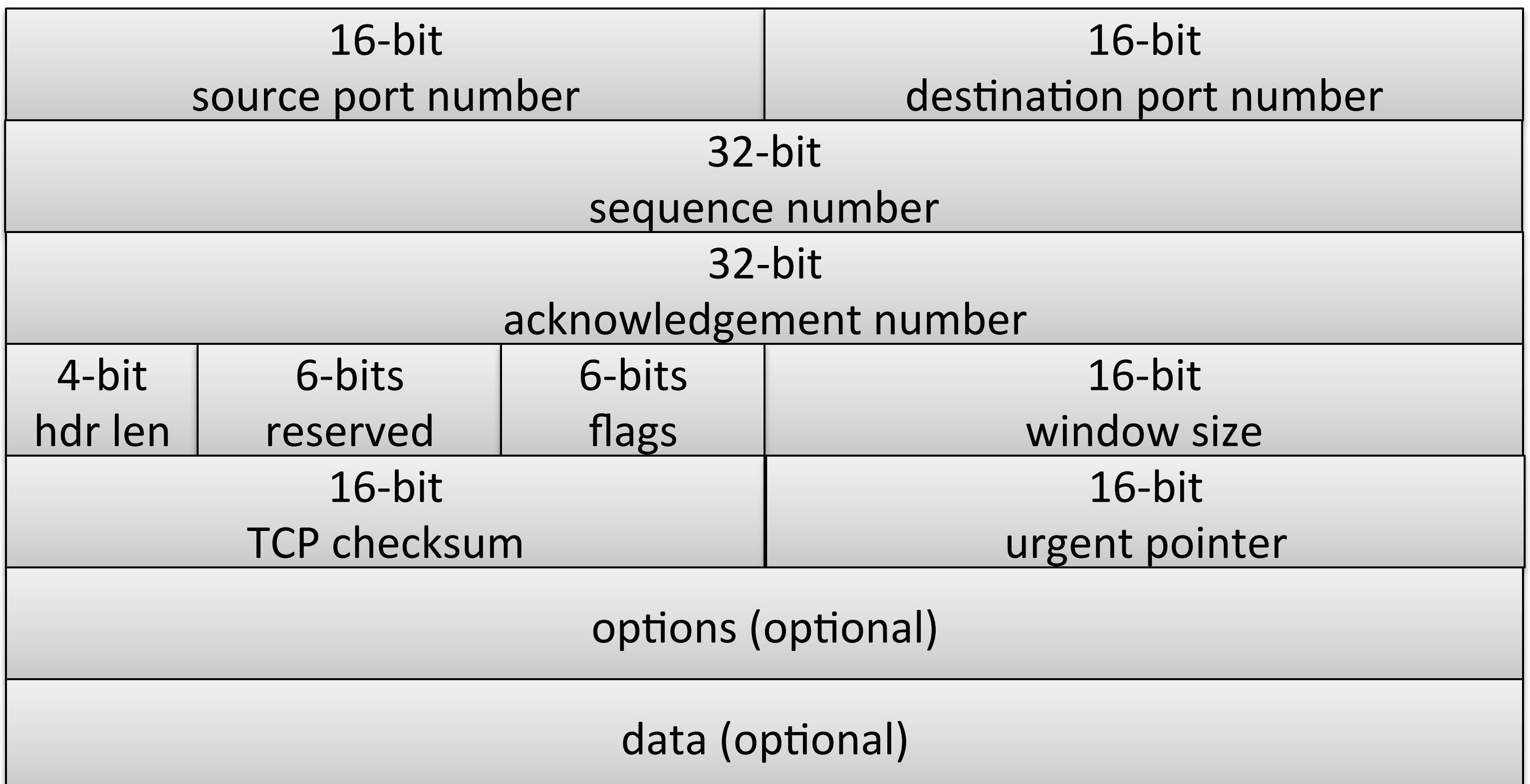
16-bit source port number	16-bit destination port number
16-bit UDP length	16-bit UDP checksum

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

TCP (transport control protocol)

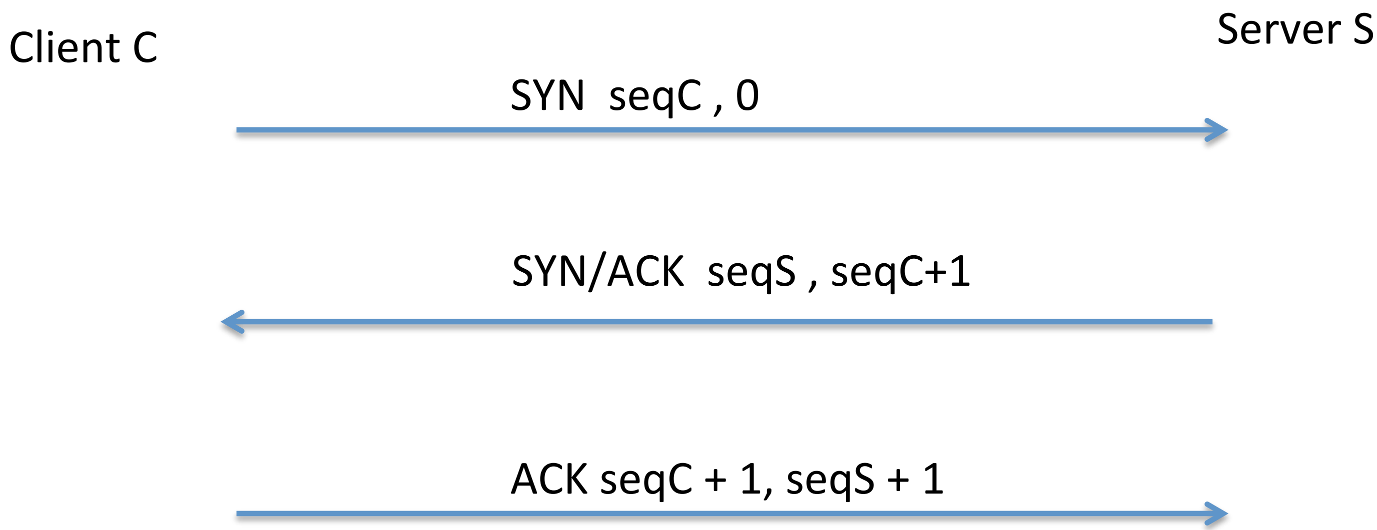


TCP (transport control protocol)

TCP flags:

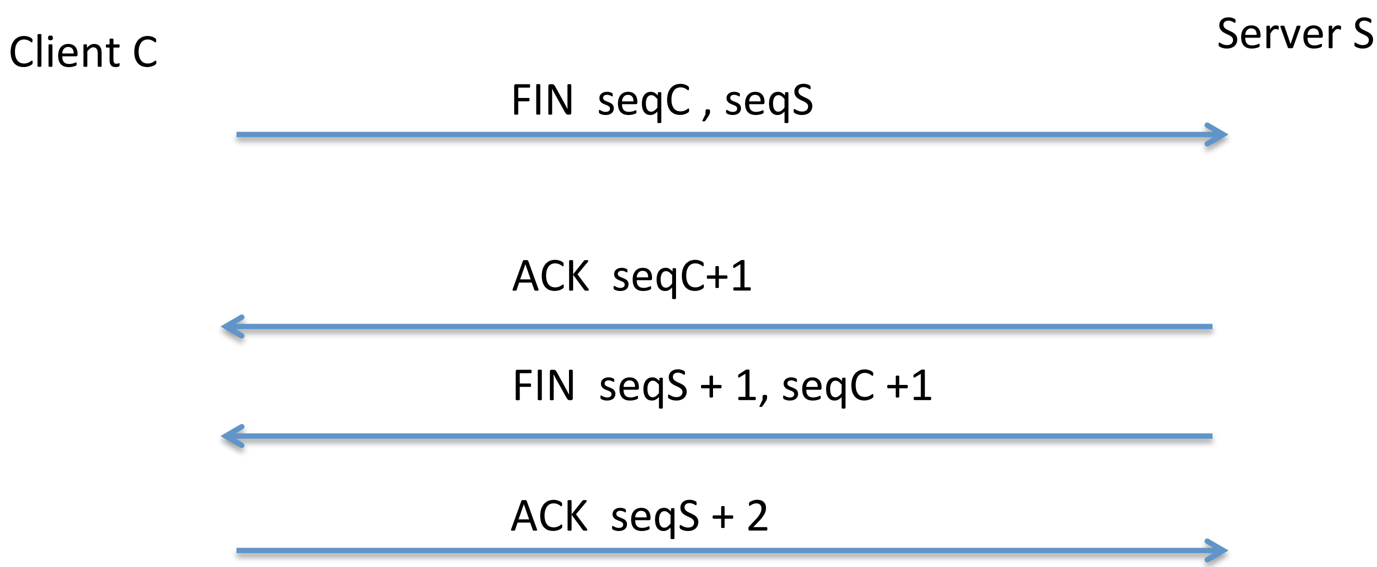
URG	urgent pointer valid
ACK	acknowledgement number valid
PSH	pass data to app ASAP
RST	reset connection
SYN	synchronize sequence #'s
FIN	finished sending data

TCP handshake



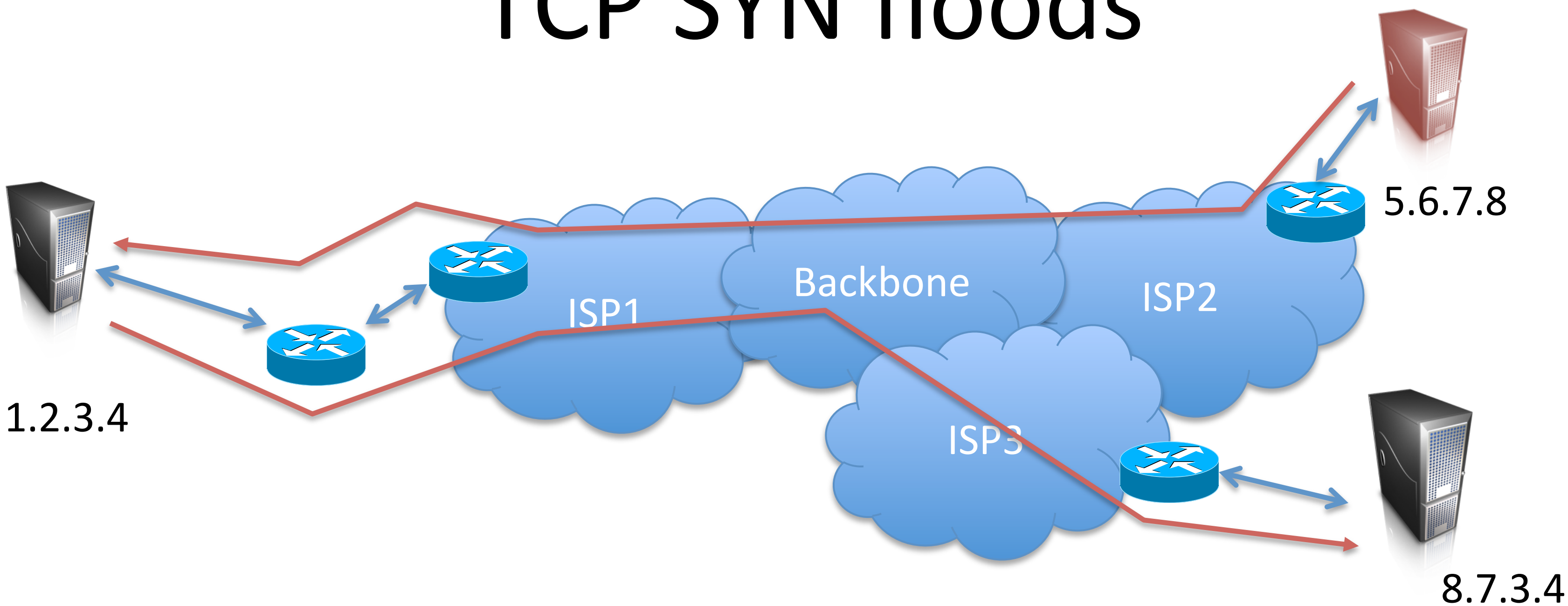
SYN = syn flag set
ACK = ack flag set
x,y = x is sequence #, y is acknowledge #

TCP teardown



SYN = syn flag set
ACK = ack flag set
x,y = x is sequence #, y is acknowledge #

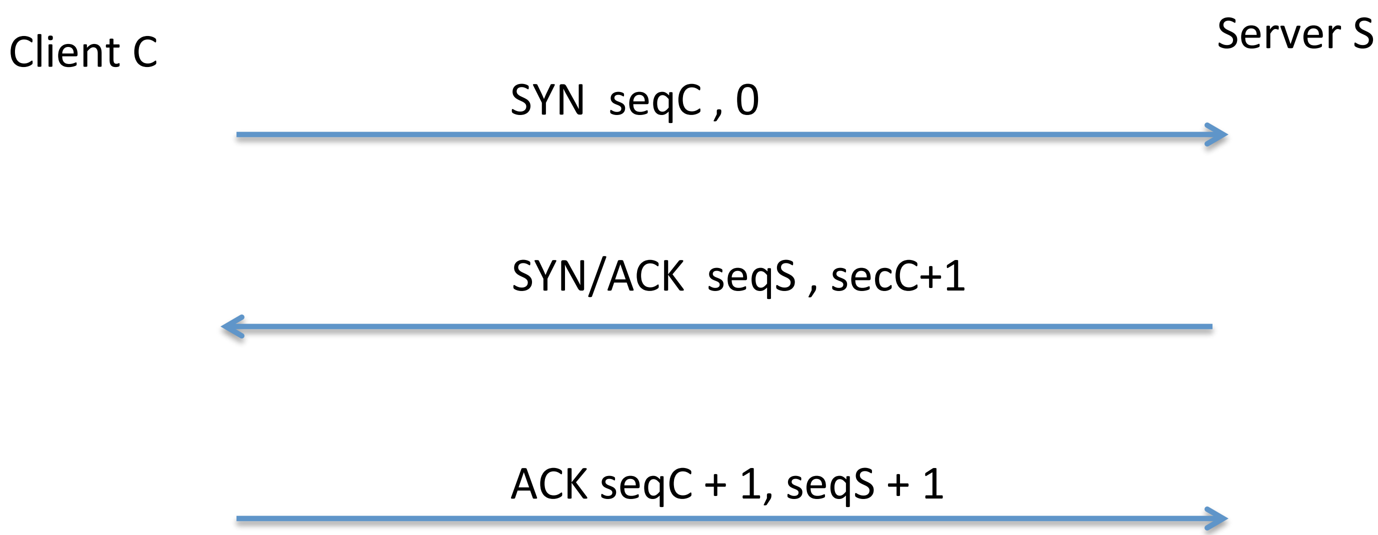
TCP SYN floods



Send lots of TCP SYN packets to 1.2.3.4

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

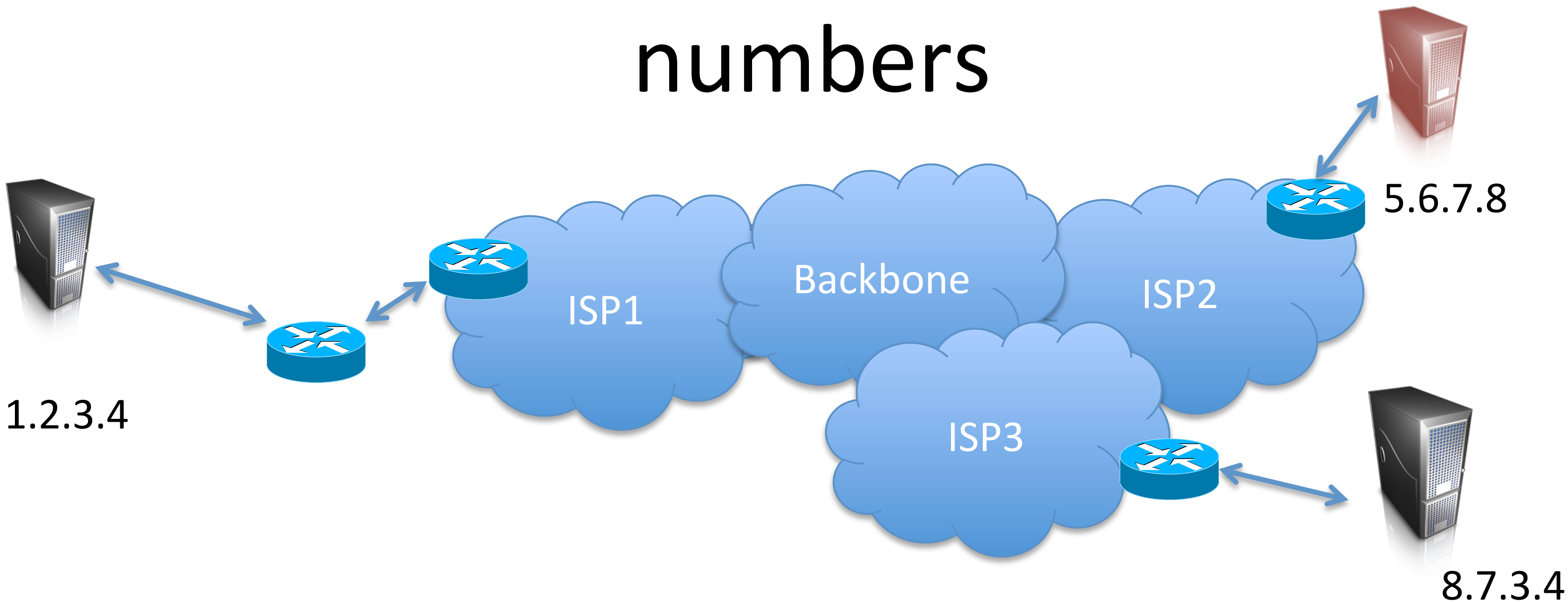
TCP handshake



How are secC and seqS selected?

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

Predictable sequence numbers

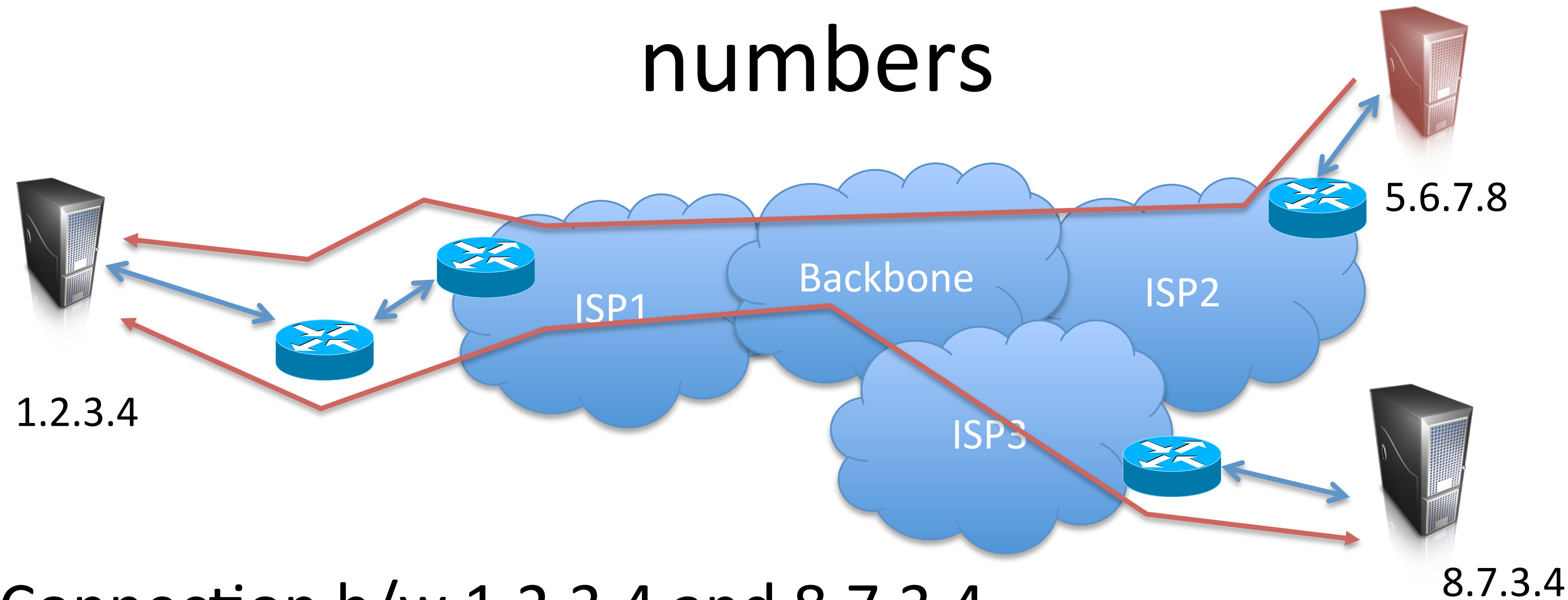


4.4BSD used predictable initial sequence numbers (ISNs)

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

What can a clever attacker do?

Predictable sequence numbers



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

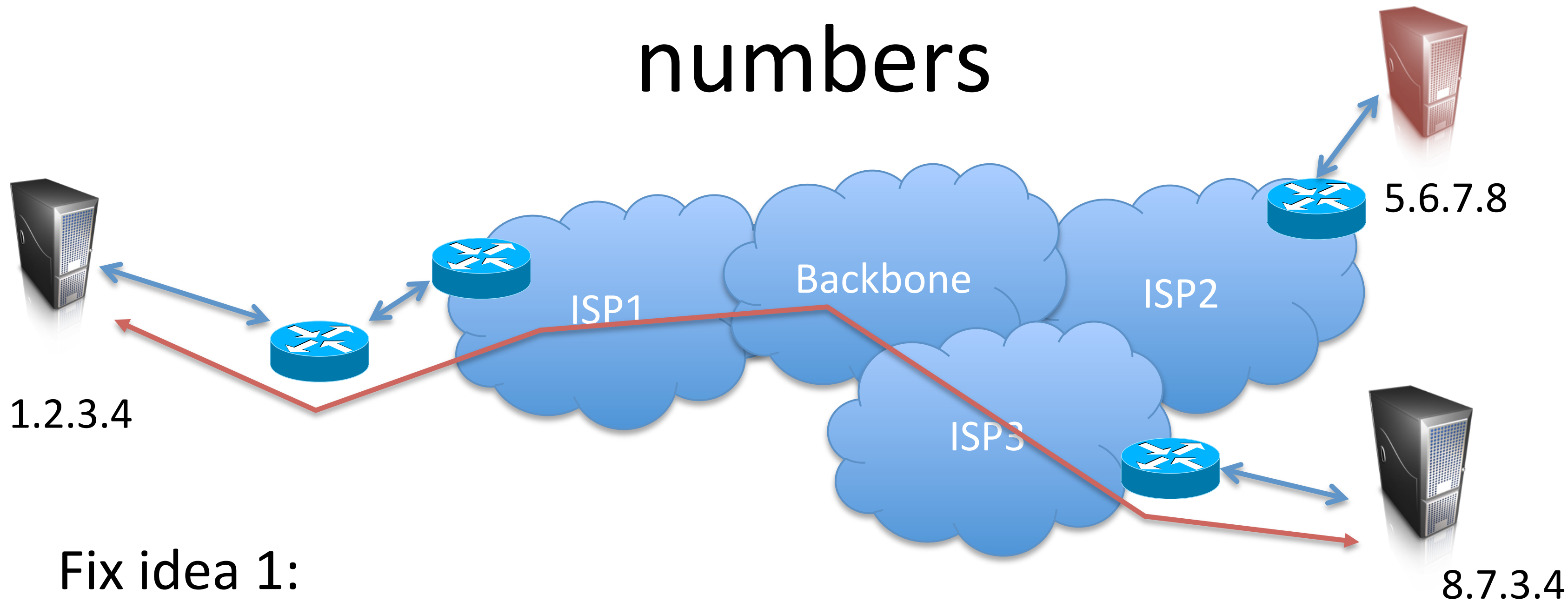
seq#(8.7.3.4)
FIN

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

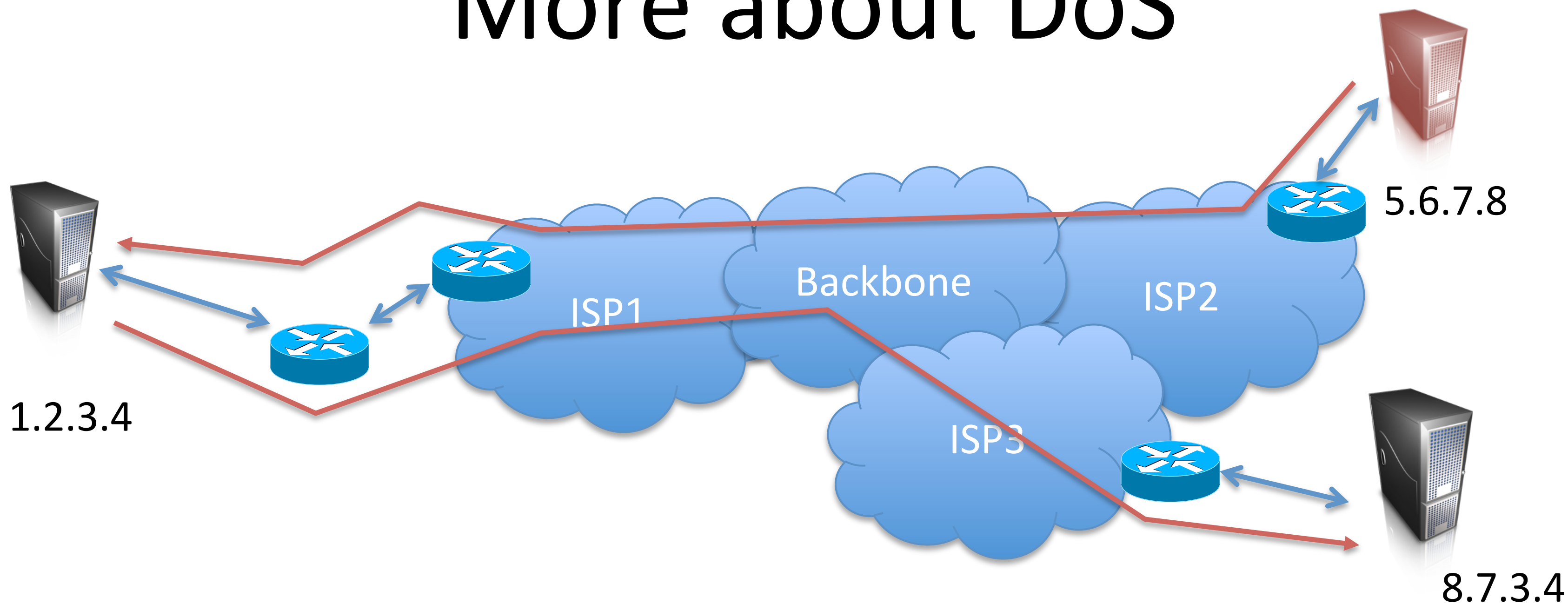
Better fix:
- Random ISN for every connection

Still issues:
- Any FIN accepted with seq# in receive window: 2^{17} attempts

TCP/IP security: other issues

- Congestion control abuse
  - can allow cheaper DoS
- No crypto
  - We'll talk about IPsec and TLS later
- BGP routing
  - we'll talk about later
- DNS (mapping from IP to domain names)
  - We'll talk about later

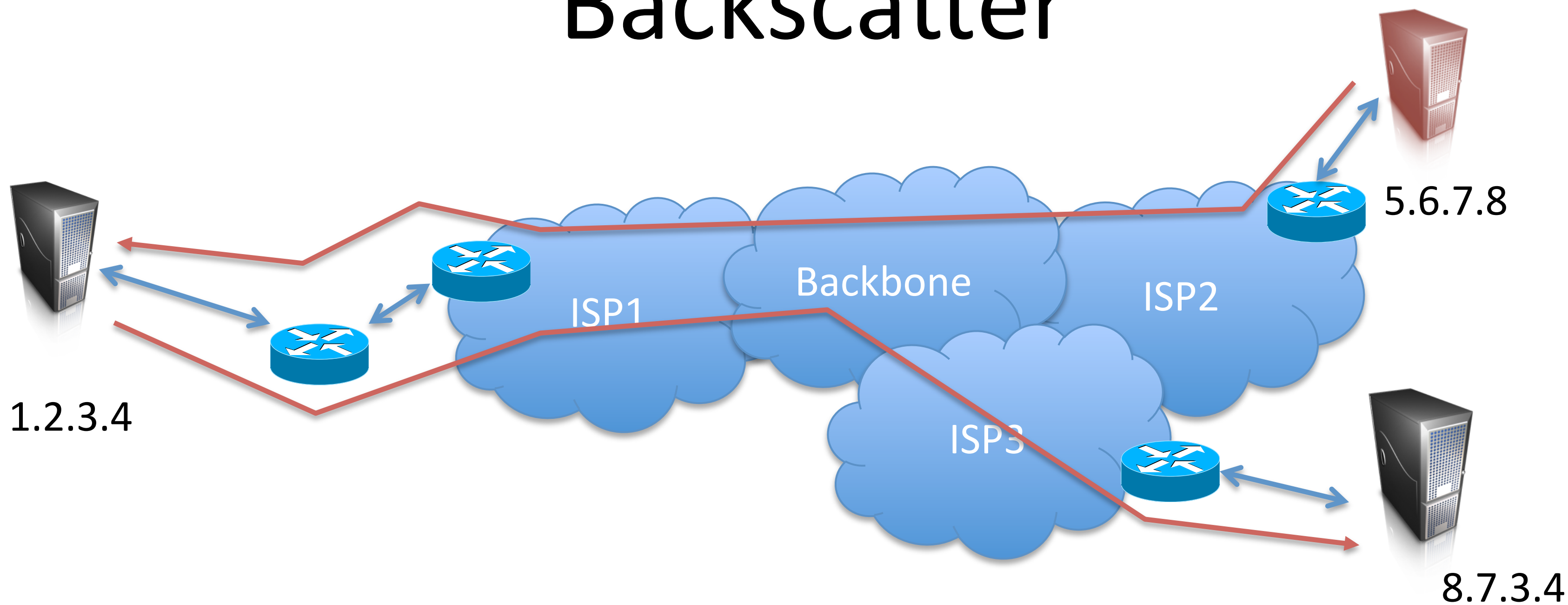
More about DoS



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

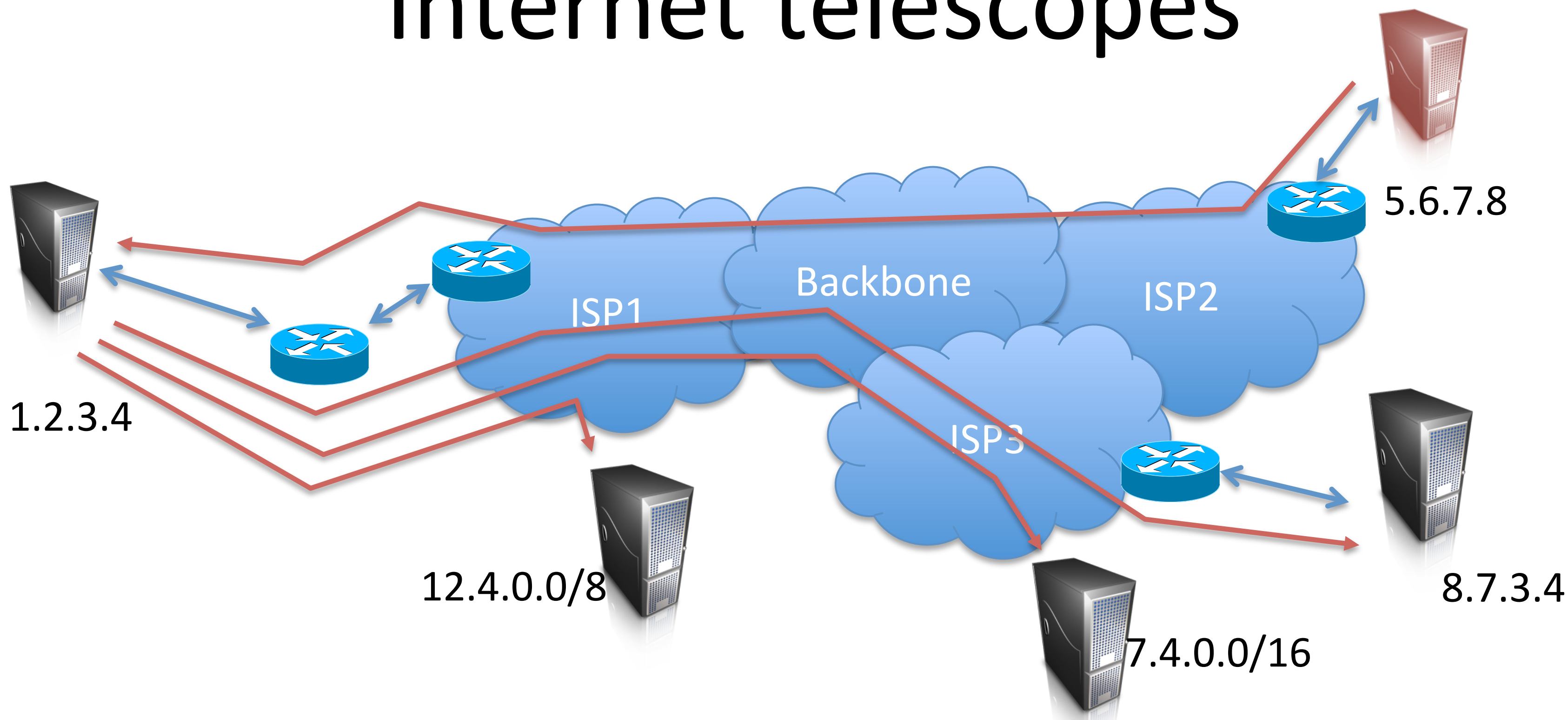
Types of responses to floods

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.

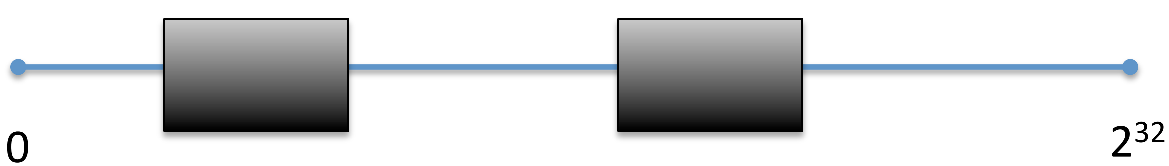
From Moore et al., "Inferring Internet Denial-of-Service Activity"

Internet telescopes



Setup some computers to watch traffic sent to darknets

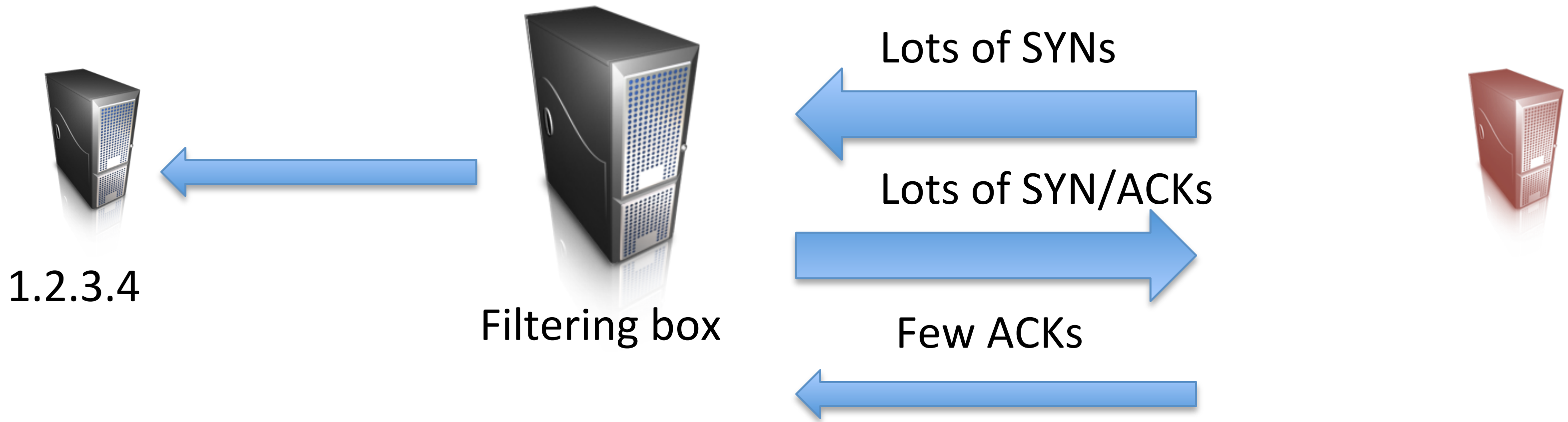
- Darknet = unused routable space



2001: 400 SYN attacks per week

2008: 4425 SYN attacks per 24 hours

Preventing DoS: Prolexic approach



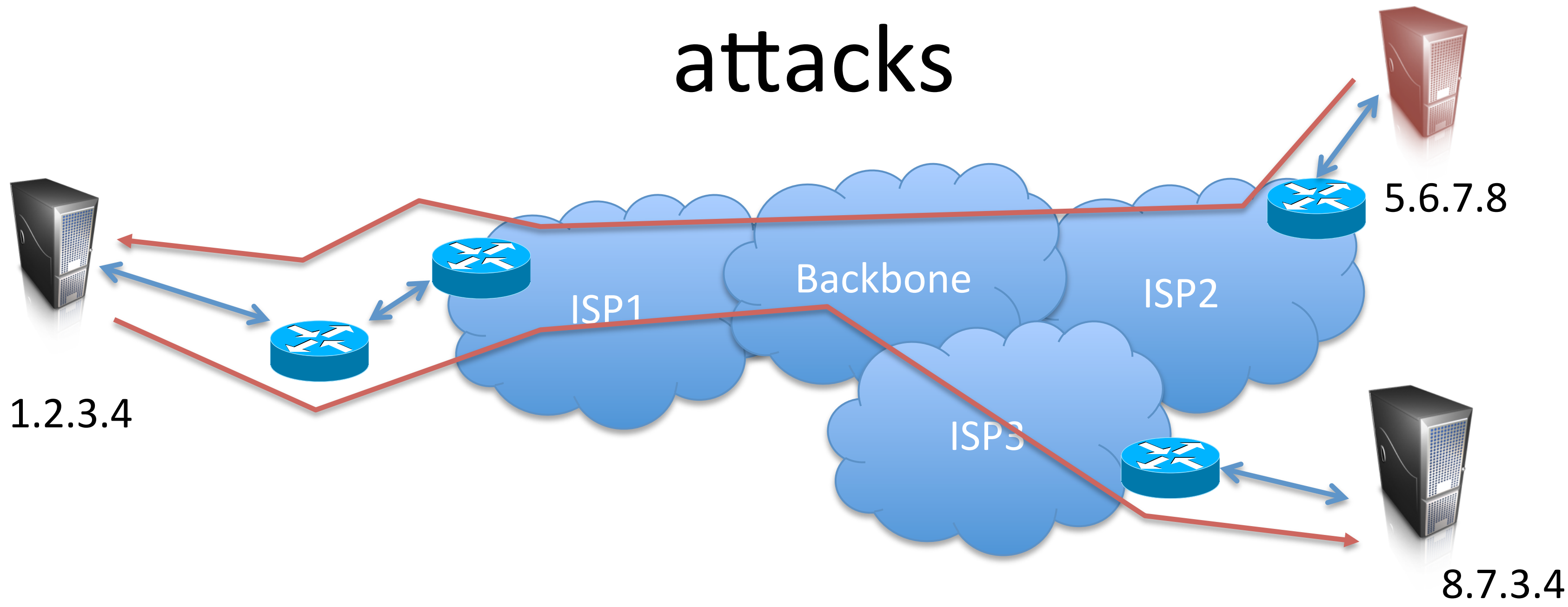
Just need a beefy box to help with filtering.
Companies pay Prolexic to do it for them

Hierarchical addressing

128.168.3.4

Class A	0	7 bits netid	24 bit hostid		
Class B	1	0	14 bits netid	16 bits hostid	
Class C	1	1	0	21 bits netid	8 bits hostid
Class D	1	1	1	0	28 bits multicast group ID
Class E	1	1	1	1	28 bits reserved for future use

Anonymous single-packet attacks



1.2.3.4 contains a buffer overflow in web server

src: 8.7.3.4
dst: 1.2.3.4

HTTP/1.1 GET
AAAAAAAAAA….

Untraceable packet of death

src: 8.7.3.4
dst: 1.2.3.4

HTTP/1.1 GET
exploit buffer

Untraceable single-packet exploit + payload

IP traceback

	Management overhead	Network overhead	Router overhead	Distributed capability	Post-mortem capability	Preventative/ reactive
Ingress filtering	Moderate	Low	Moderate	N/A	N/A	Preventative
Link testing						
Input debugging	High	Low	High	Good	Poor	Reactive
Controlled flooding	Low	High	Low	Poor	Poor	Reactive
Logging	High	Low	High	Excellent	Excellent	Reactive
ICMP Traceback	Low	Low	Low	Good	Excellent	Reactive
Marking	Low	Low	Low	Good	Excellent	Reactive

From Savage et al., "Practical Network Support for IP Traceback"