

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

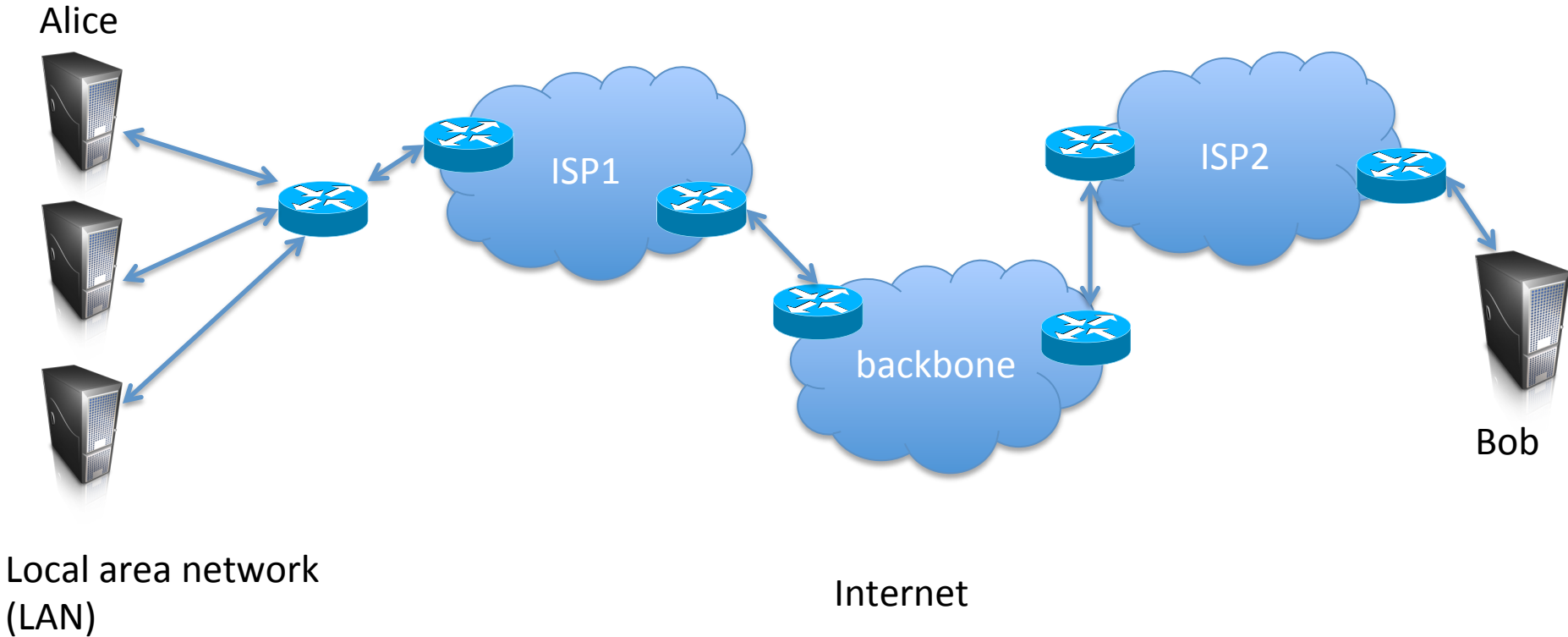
IP spoofing, fragmentation

UDP and TCP

Denial of Service

IP traceback, filtering

Internet



Ethernet

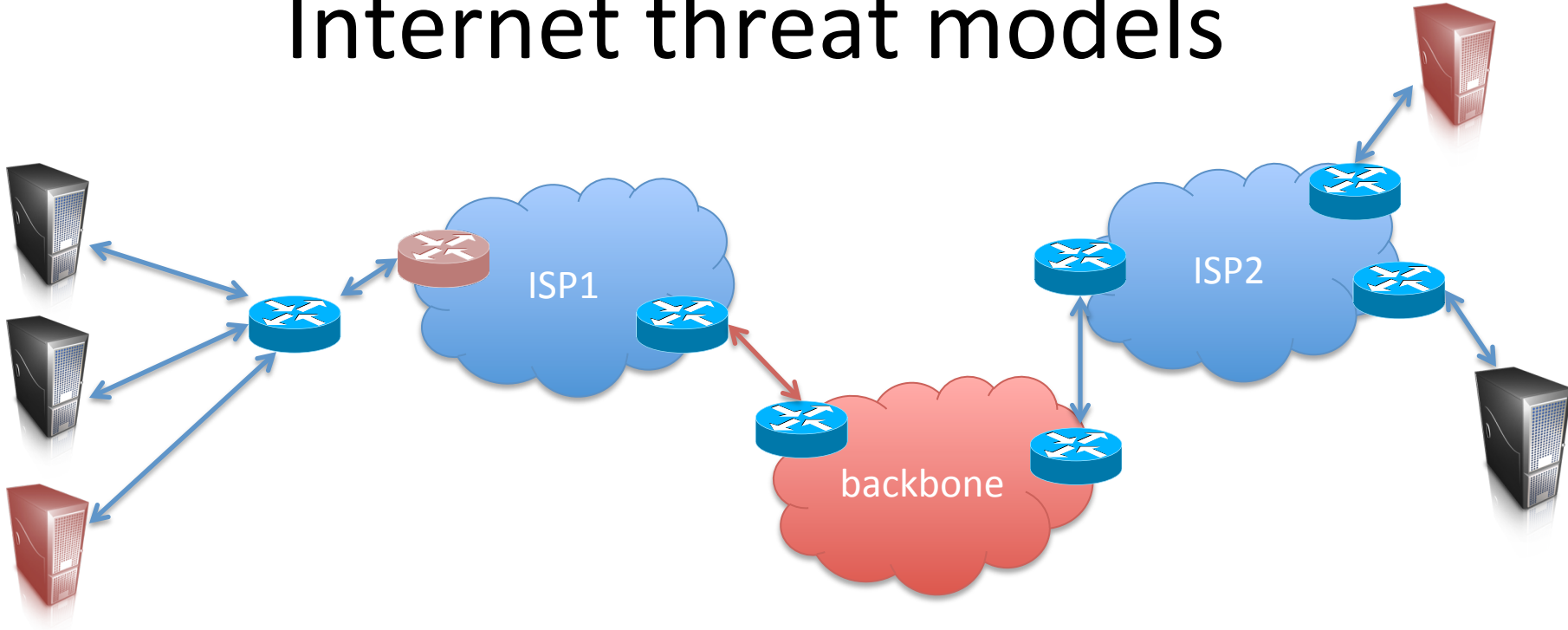
802.11

TCP/IP

BGP (border gateway protocol)

DNS (domain name system)

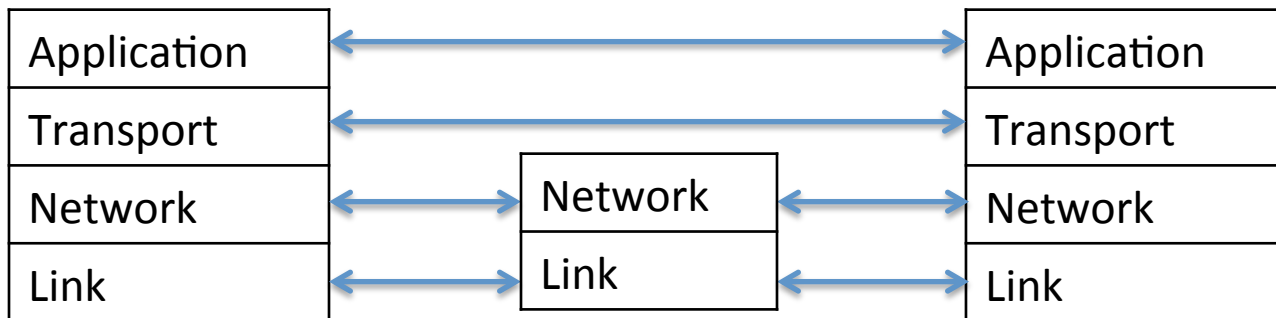
Internet threat models



- (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)

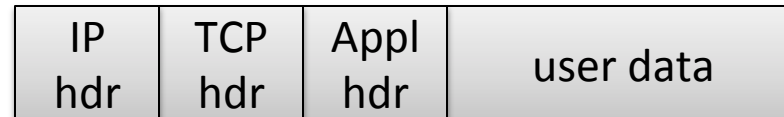
- Connectionless
 - no state
- Unreliable
 - no guarantees
- ICMP (Internet Control Message Protocol)
 - error messages, etc.
 - often used by tools such as ping, traceroute

Internet protocol stack

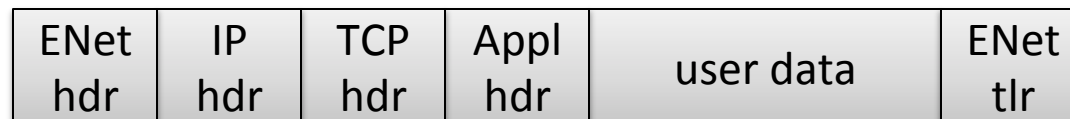
Application
TCP
IP
Ethernet



TCP segment



IP datagram



Ethernet frame

14 20 20



46 to 1500 bytes

IPv4


















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)				

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

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>

Classless Inter-Domain routing (CIDR)

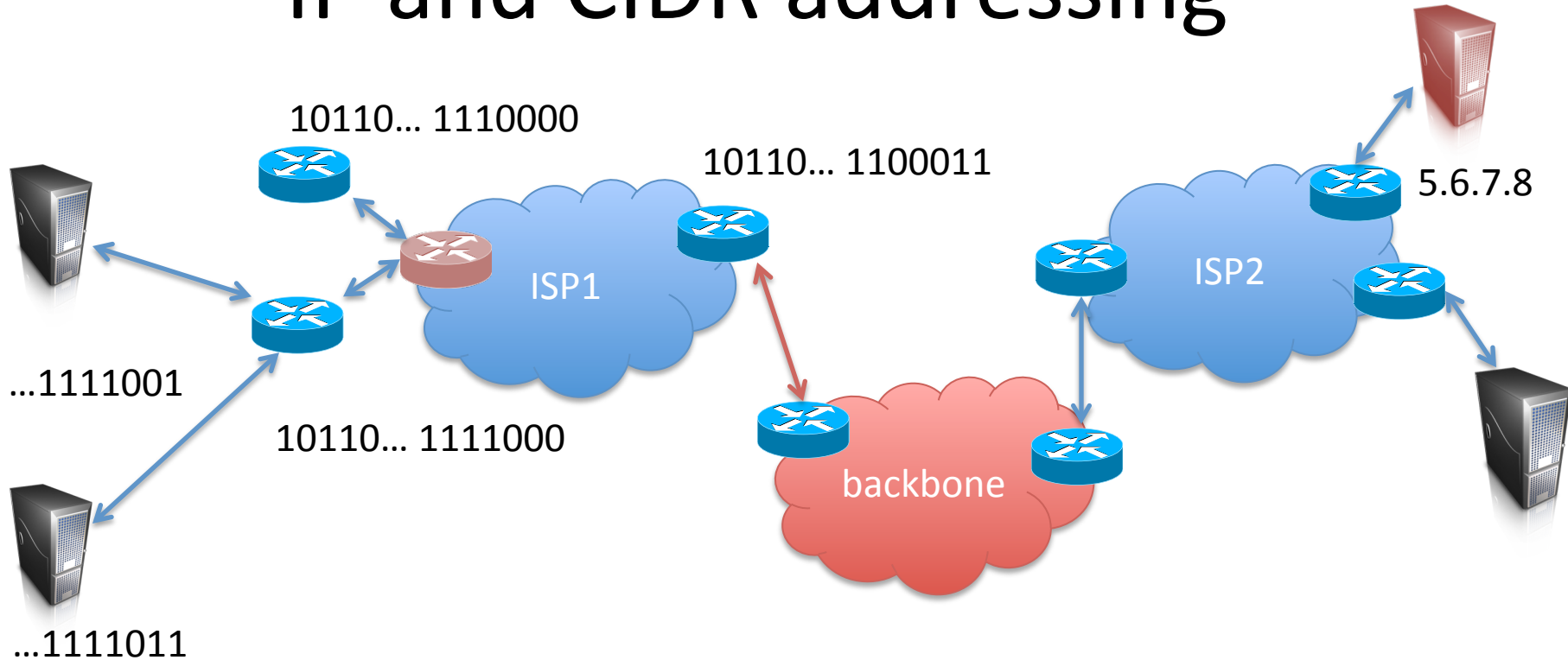
128.168.0.0/16

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

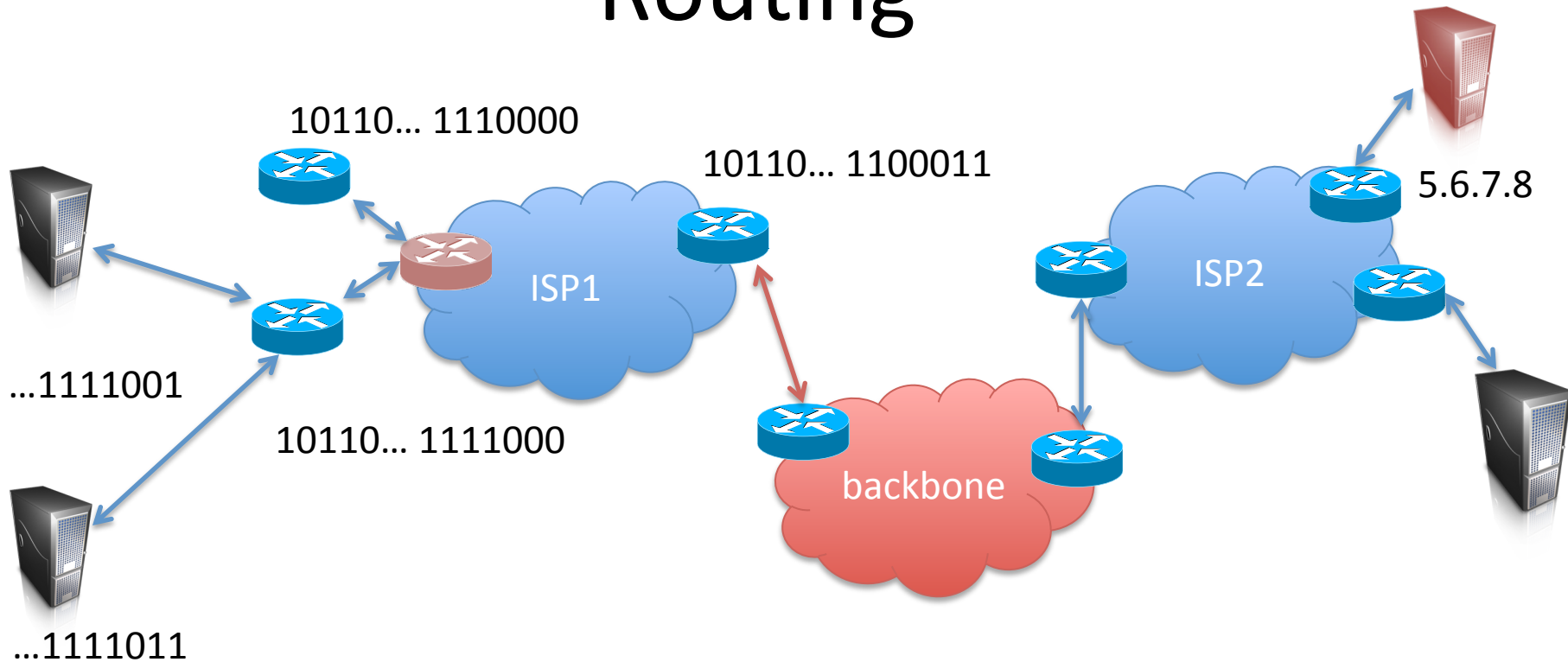
IP and CIDR addressing



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



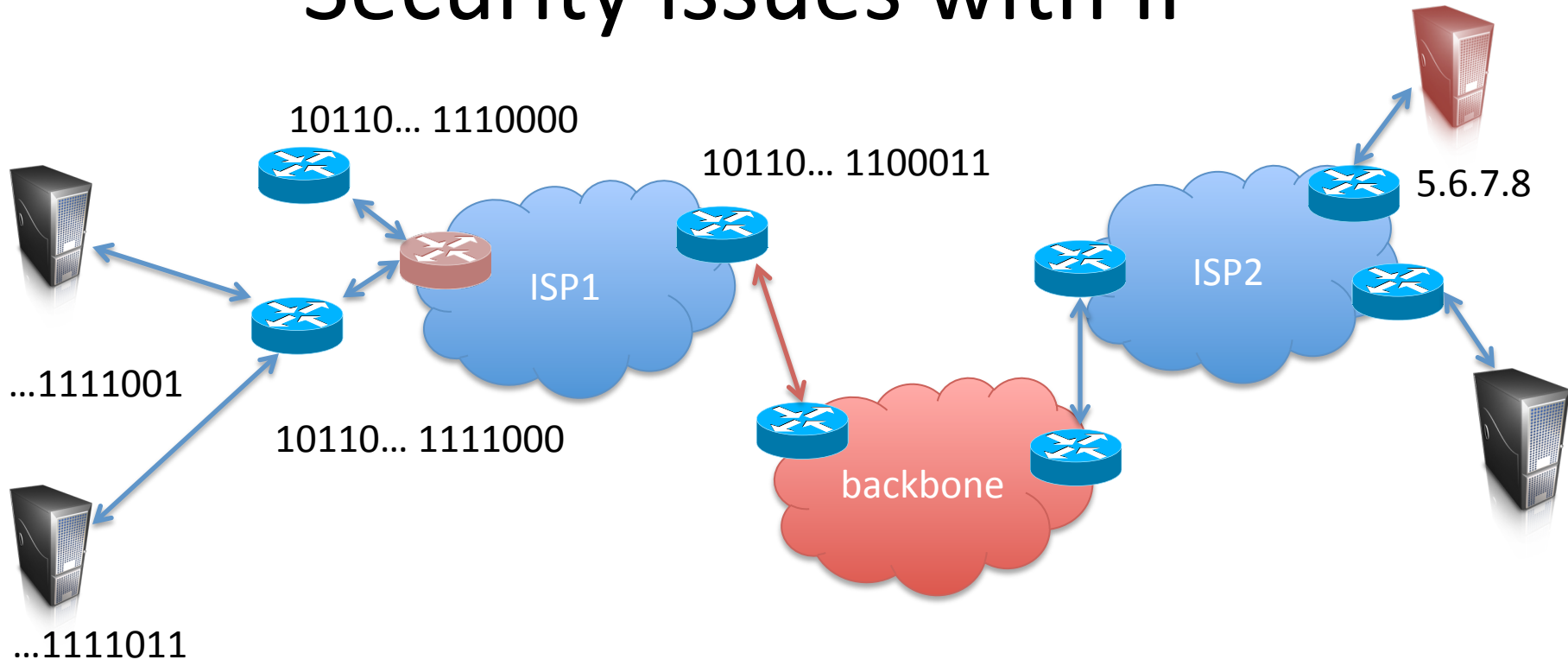
Autonomous systems (AS) are organizational building blocks

- 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)

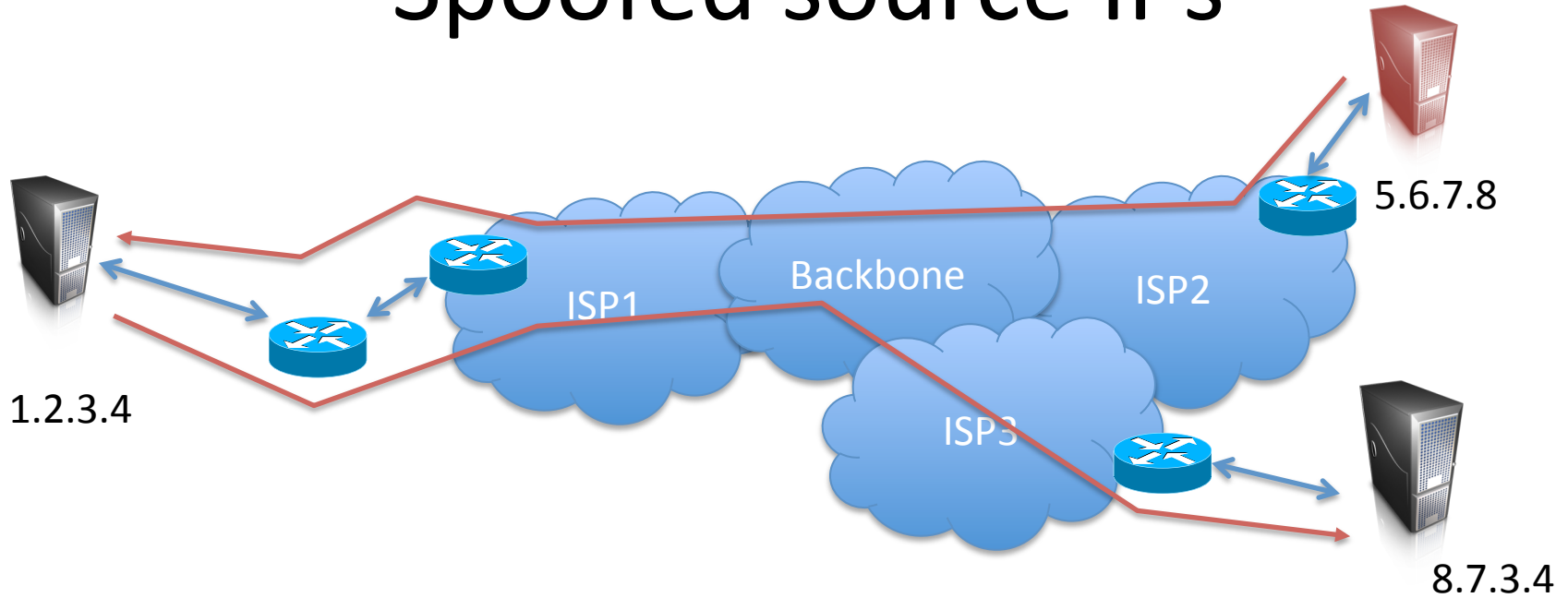
Security issues with IP



Routing has issues, we'll get to that later
What else?

- No source address authentication
- IP spoofing

Spoofer source IPs

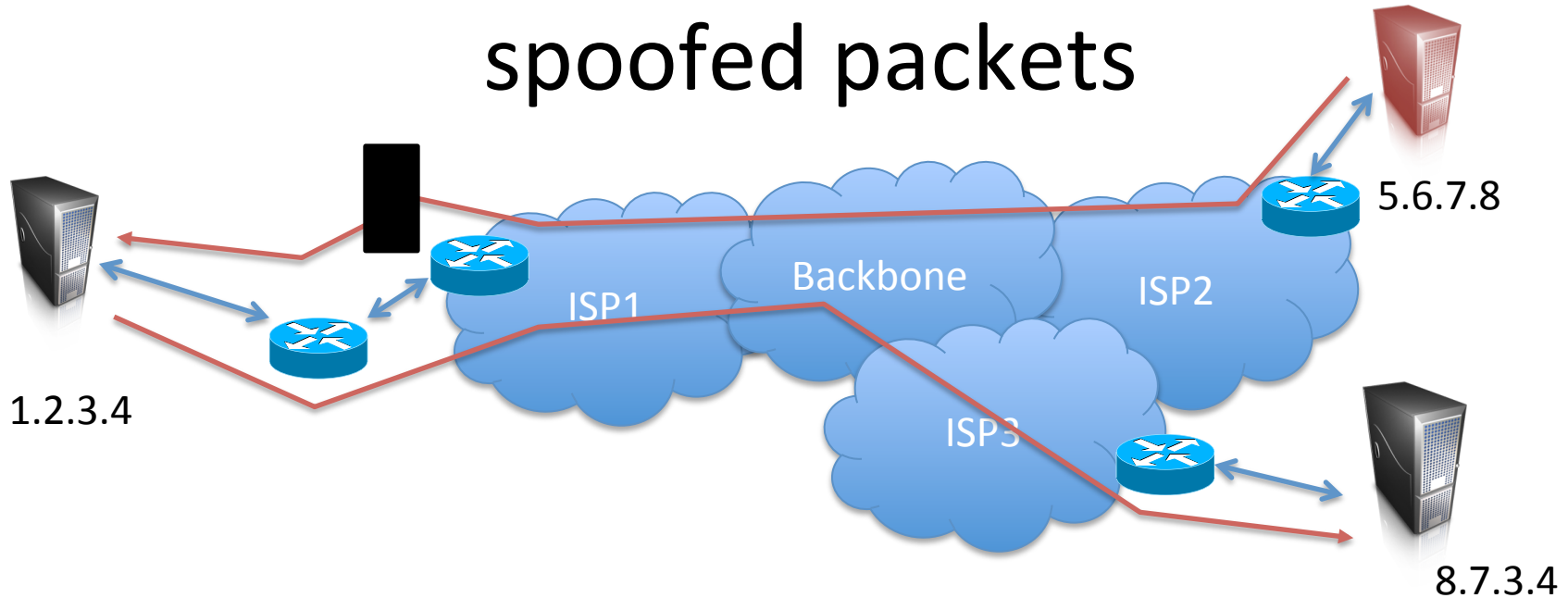


Attacker can send packet with fake source IP
Packet will get routed correctly
Replies will not

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

How might an attacker abuse this?

Denial of Service using spoofed packets



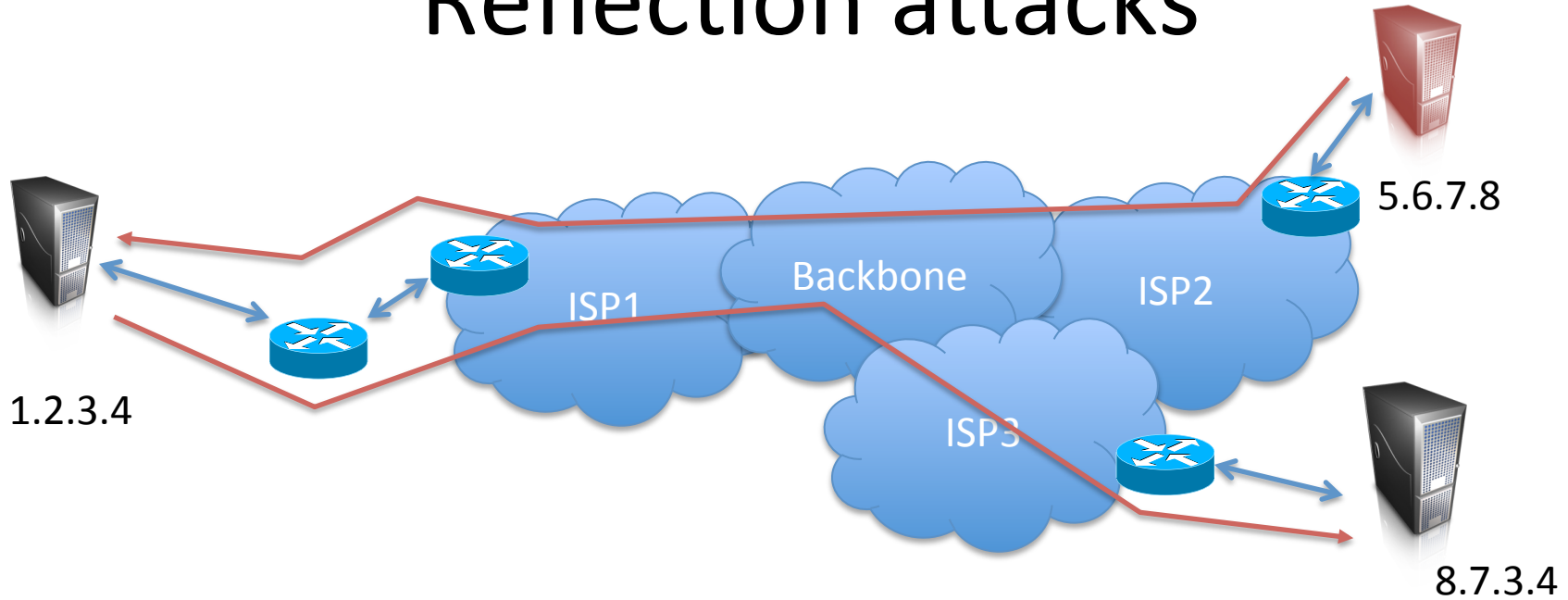
Easy to use ingress filtering to prevent simple DoS

- Attacker doesn't send lots of packets with src 5.6.7.8

Early DoS would use randomly chosen spoofed source IP

- Ingress filtering for 1.2.3.4 less effective
- Egress filtering may still work at ISP2

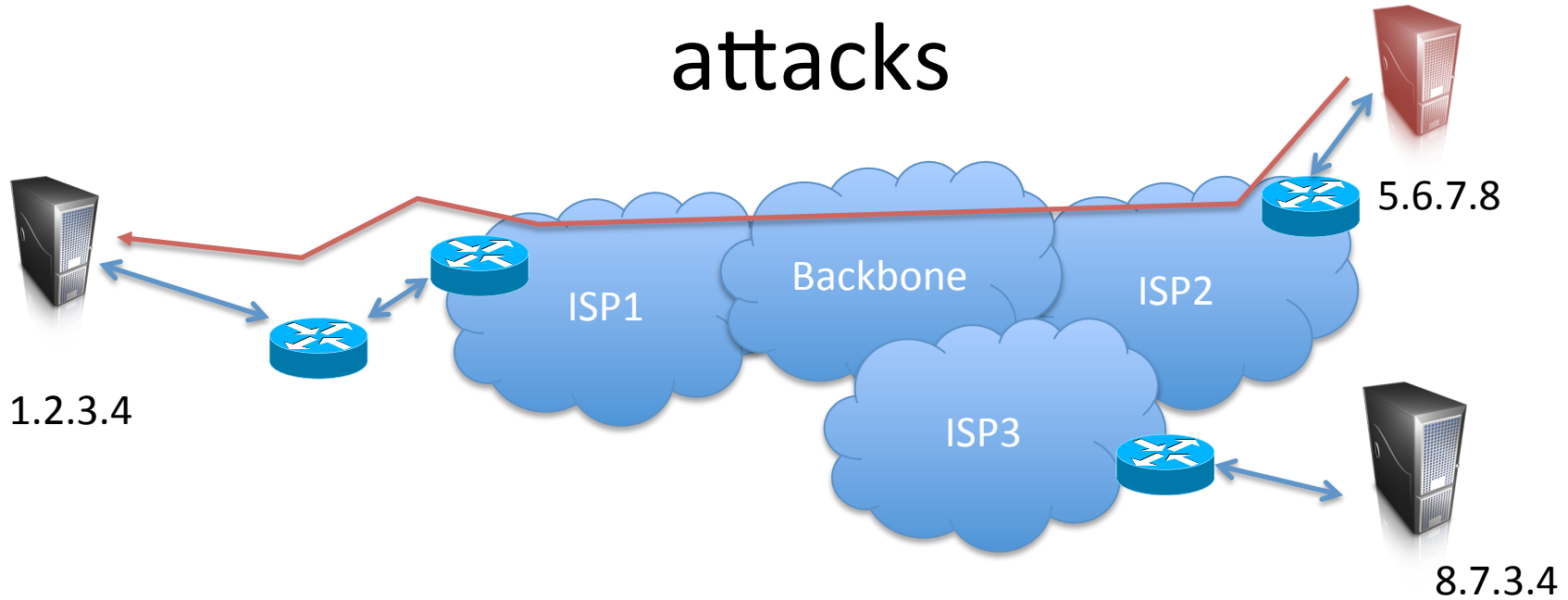
Reflection attacks



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
- Hides a single-packet exploit even better
(1.2.3.4 in foreign country)

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  
AAAAAAAAAAAA.....
```

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

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



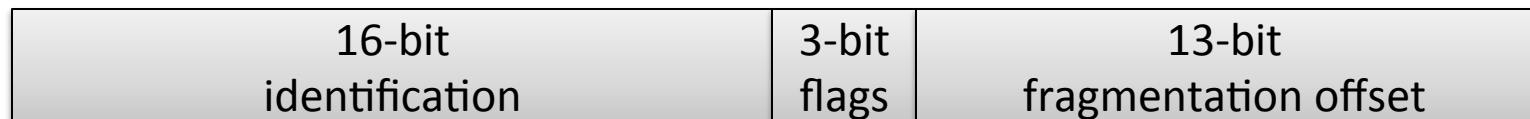
Ethernet frame
containing
IP datagram

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

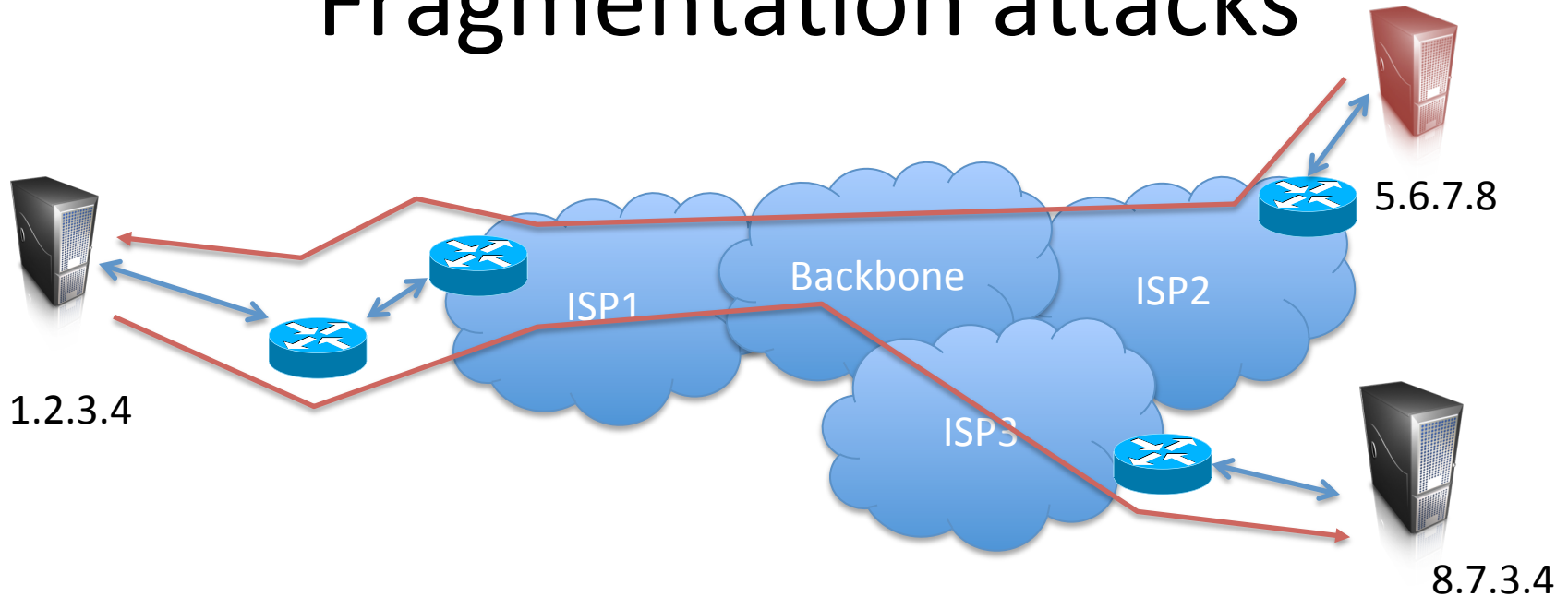
Flags:

0 b1 b2

where b1 = May Fragment (0) / Don't Fragment (1)

where b2 = Last Fragment (0) / More Fragments (1)

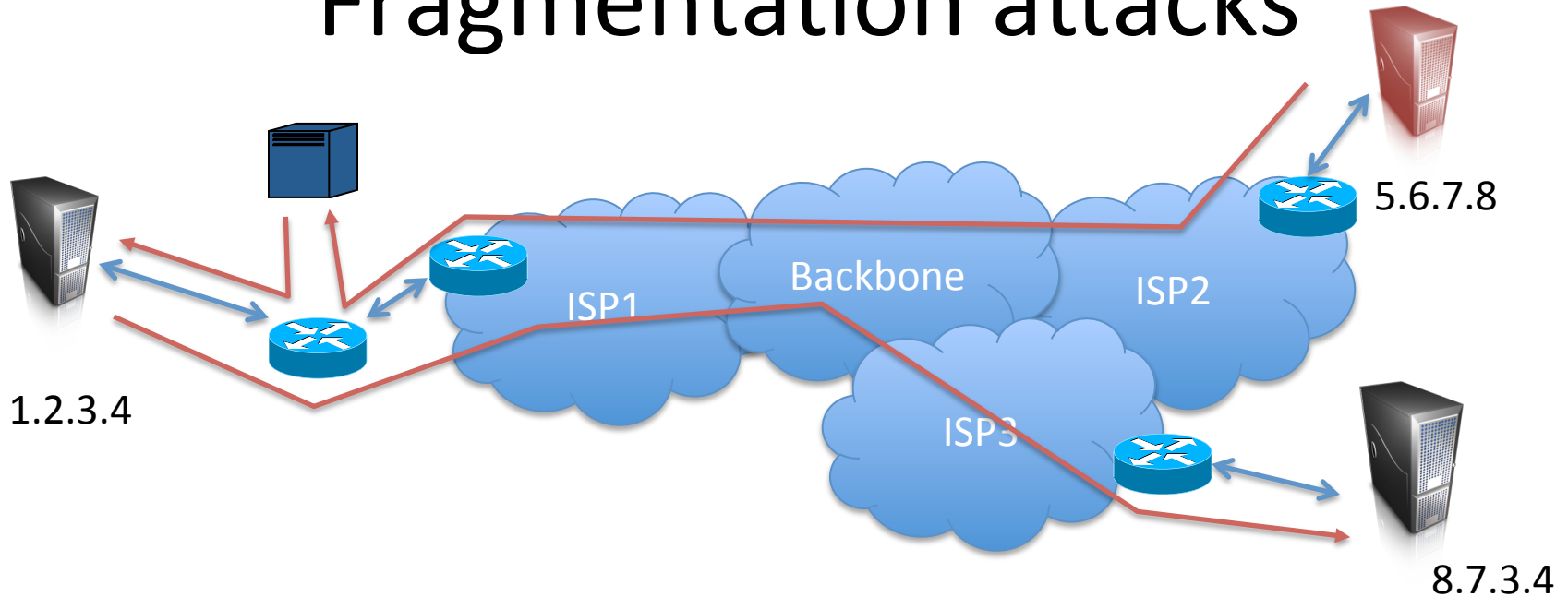
Fragmentation attacks



Fragmentation is a mess:

- Teardrop DoS (mangled fragmentation crashes reconstruction code). Set offsets so that two packets have overlapping data

Fragmentation attacks



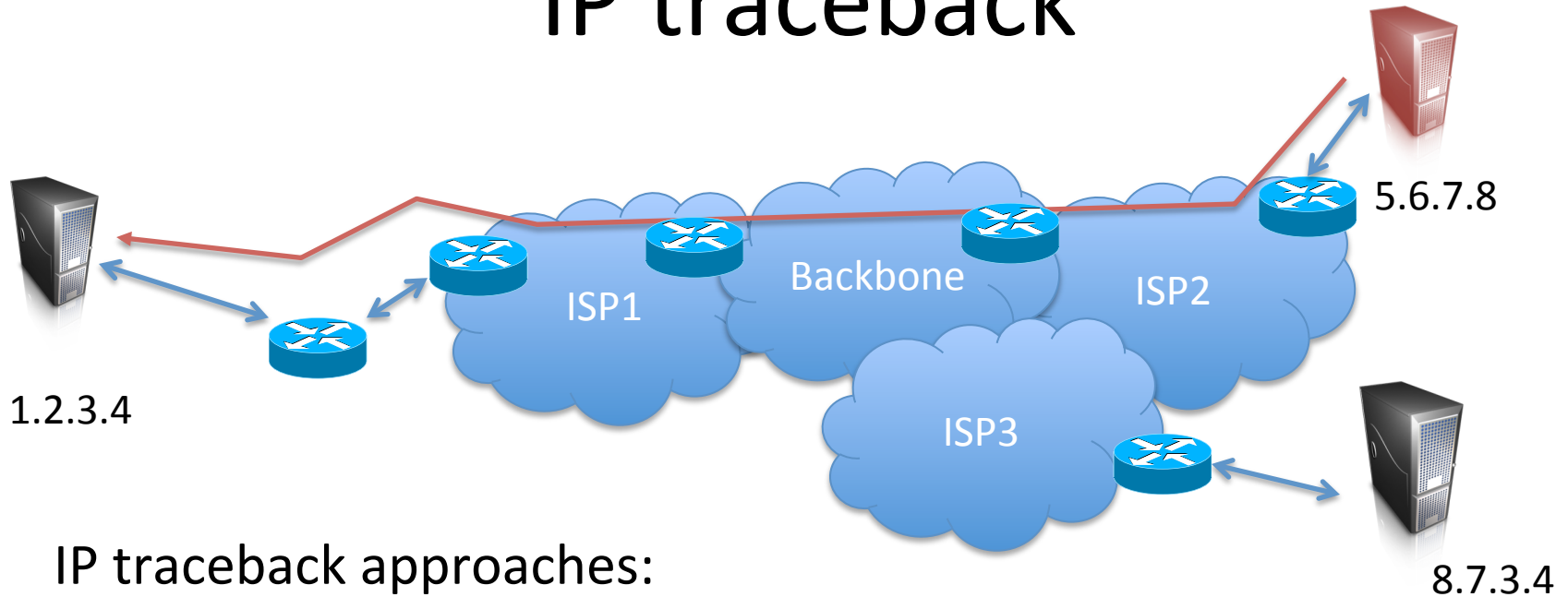
Fragmentation is a mess:

- Teardrop DoS (mangled fragmentation crashes reconstruction code). Set offsets so that two packets have overlapping data
- Such manipulations also allow avoiding IDS systems
 - IDS scans packets for exploit strings
 - Add random data into packets, overwrite later during reconstruction due to overlapping fragments
- DoS by filling up fragmentation buffers

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

IP traceback

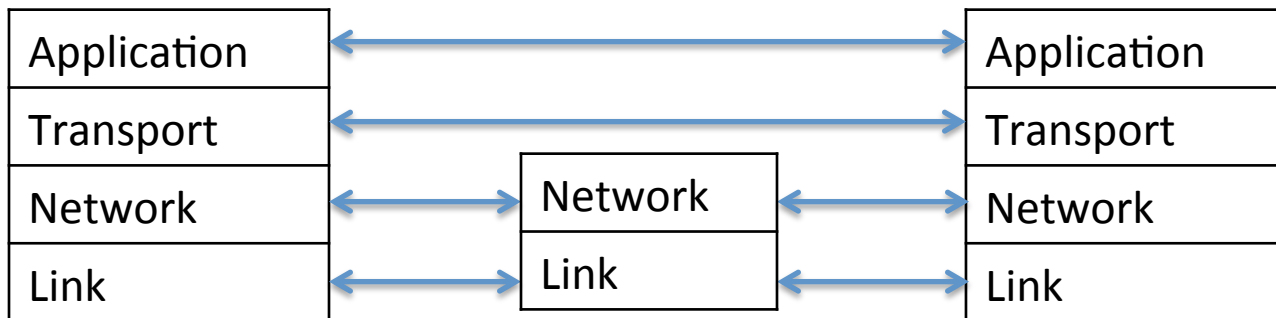
	Management overhead	Network overhead	Router overhead	Distributed capability	Post-mortem capability	Preventative/reactive
Ingress filtering	Moderate	Low	Moderate	N/A	N/A	Preventative
Link testing						
Input debugging	High	Low	High	Good	Poor	Reactive
Controlled flooding	Low	High	Low	Poor	Poor	Reactive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

Limitations?

Internet protocol stack

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

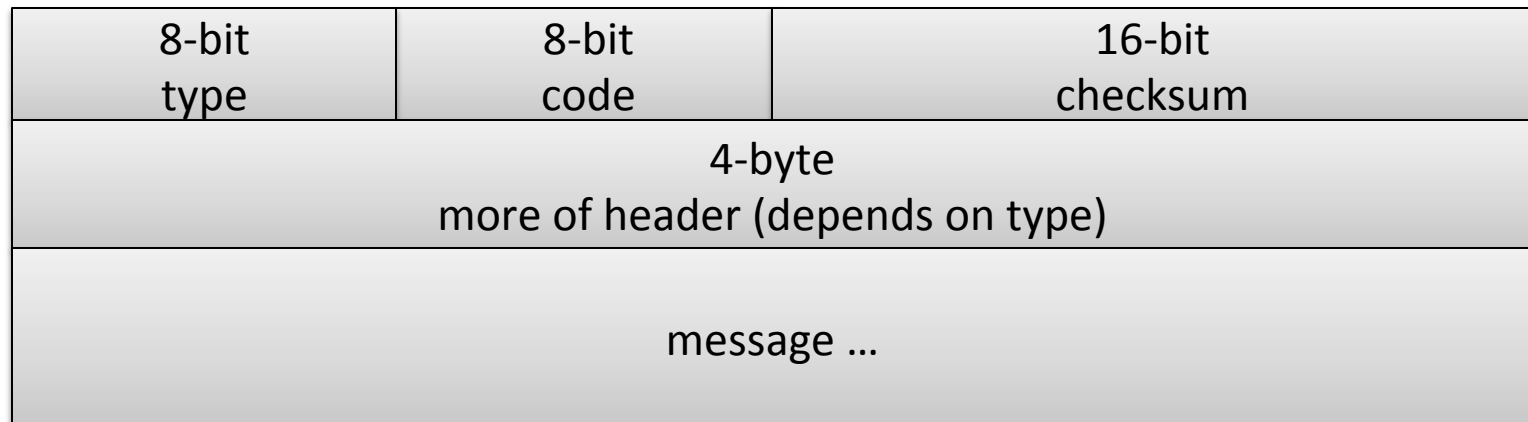


ICMP

(Internet Control Message Protocol)



IP datagram
containing
UDP datagram

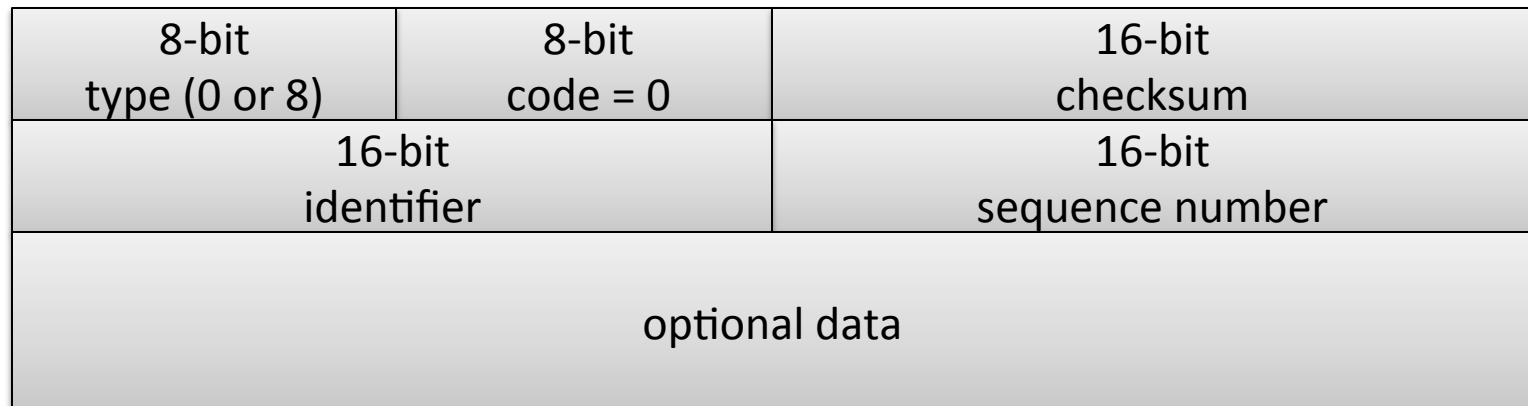


ICMP

(Internet Control Message Protocol)

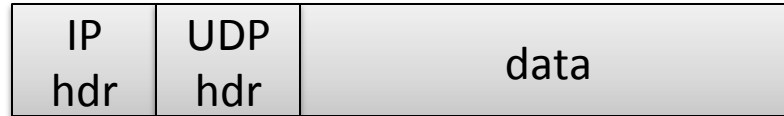


IP datagram
containing
UDP datagram



Echo request (used by ping)

UDP (user datagram protocol)



IP datagram
containing
UDP datagram

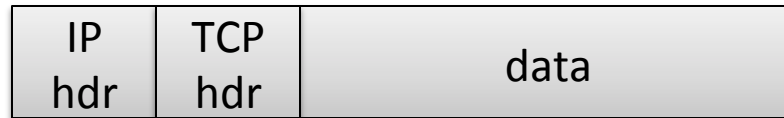
16-bit source port number	16-bit destination port number
16-bit UDP length	16-bit UDP checksum

length = header len + data len

TCP (transport control protocol)

- Connection-oriented
 - state initialized during handshake and maintained
- Reliability is a goal
 - generates segments
 - timeouts segments that aren't ack'd
 - checksums headers,
 - reorders received segments if necessary
 - flow control

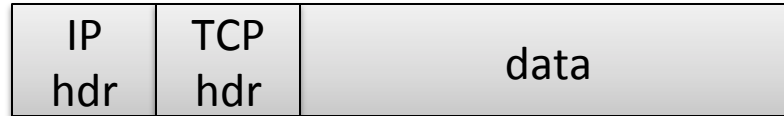
TCP (transport control protocol)



IP datagram
containing
UDP datagram

16-bit source port number		16-bit destination port number	
32-bit sequence number			
32-bit acknowledgement number			
4-bit hdr len	6-bits reserved	6-bits flags	16-bit window size
16-bit TCP checksum		16-bit urgent pointer	
options (optional)			
data (optional)			

TCP (transport control protocol)

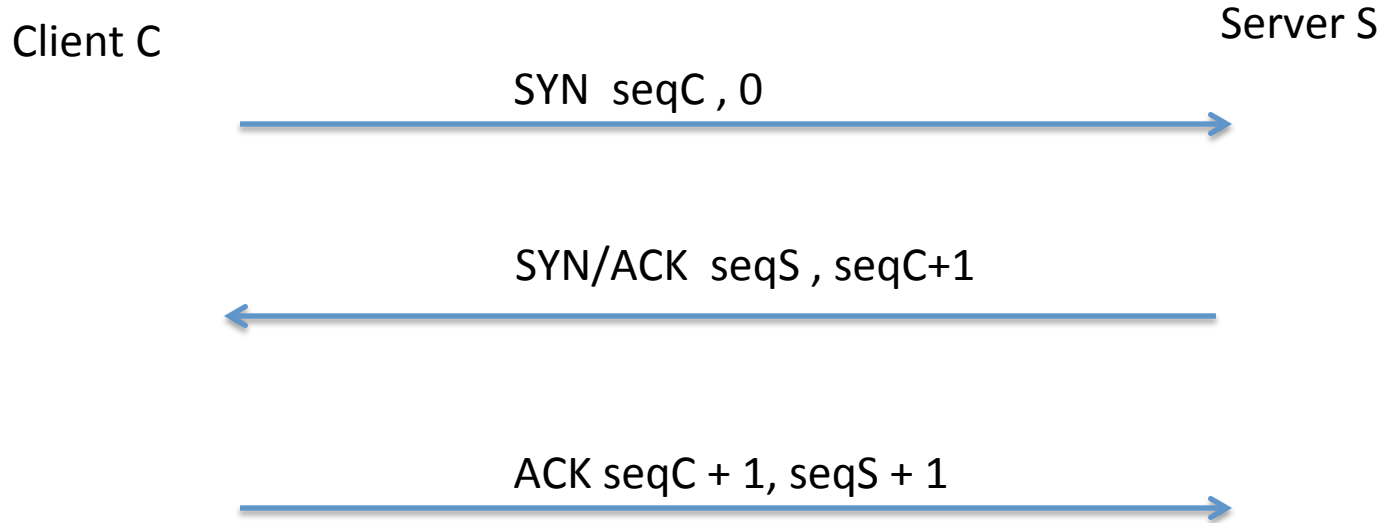


IP datagram
containing
UDP datagram

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

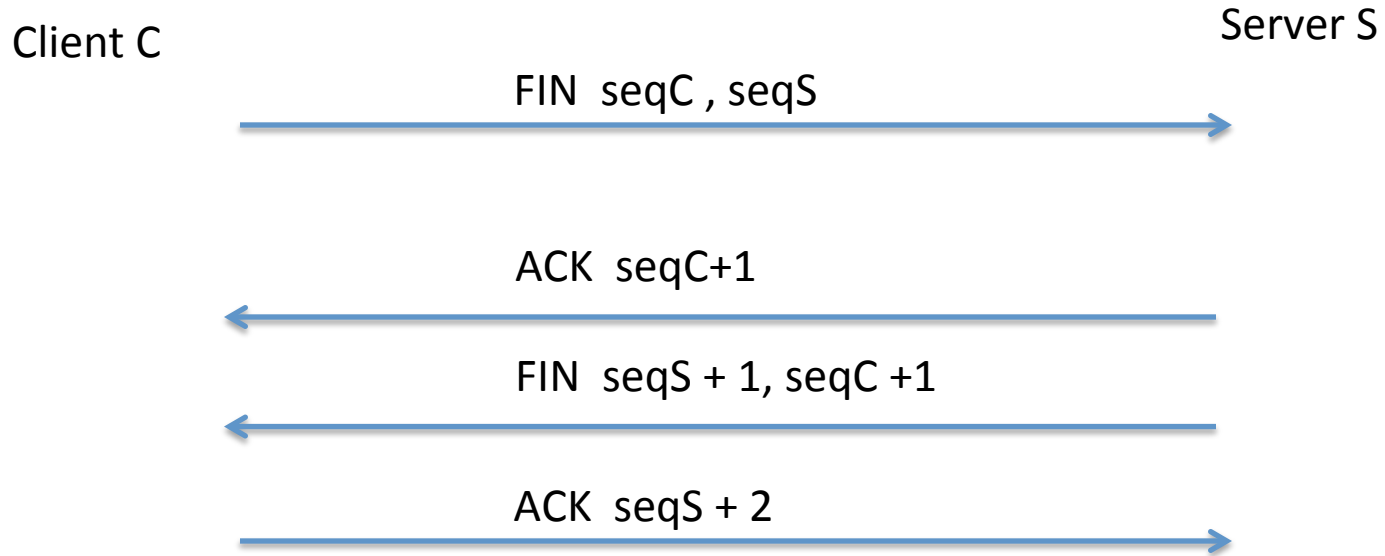


SYN = syn flag set

ACK = ack flag set

x,y = x is sequence #, y is acknowledge #

TCP teardown

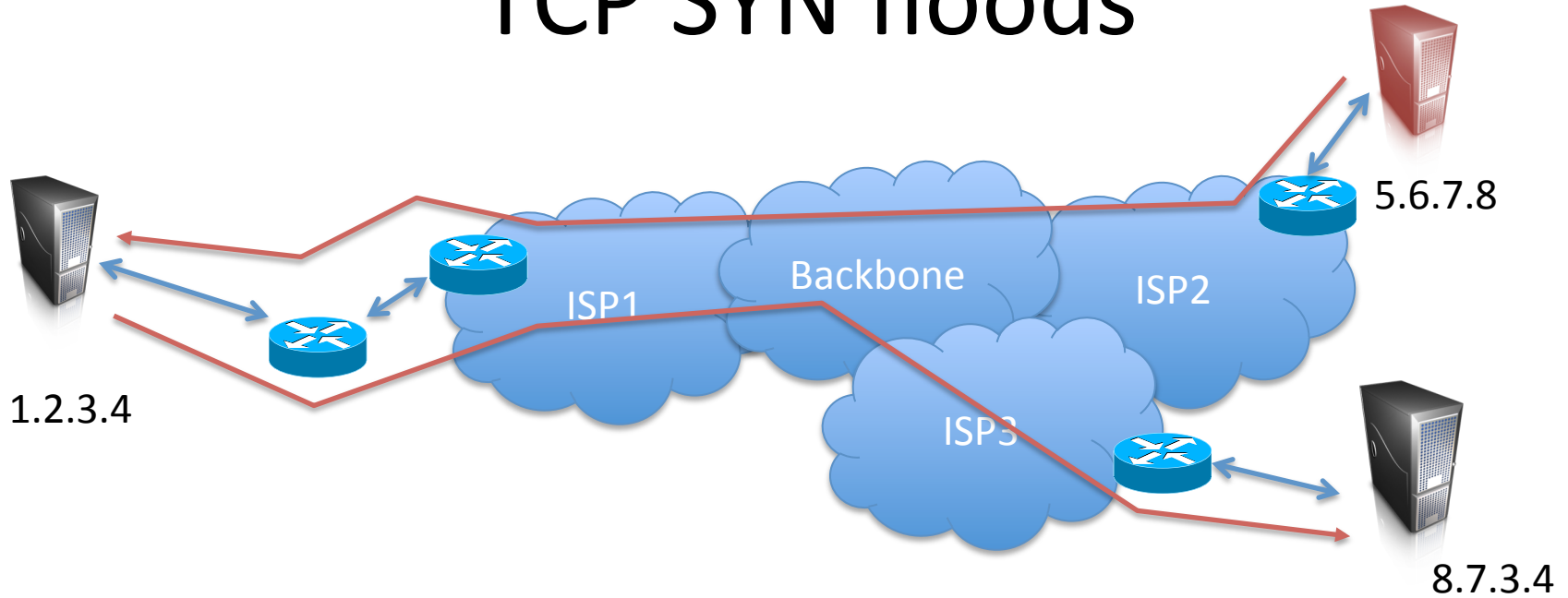


SYN = syn flag set

ACK = ack flag set

x,y = x is sequence #, y is acknowledge #

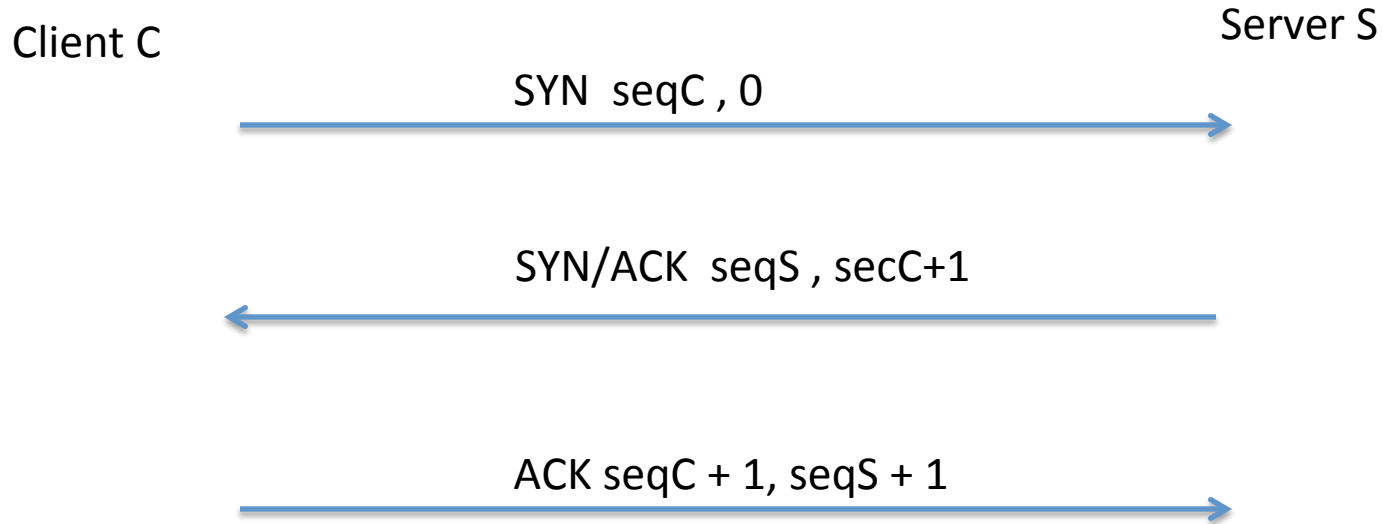
TCP SYN floods



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?

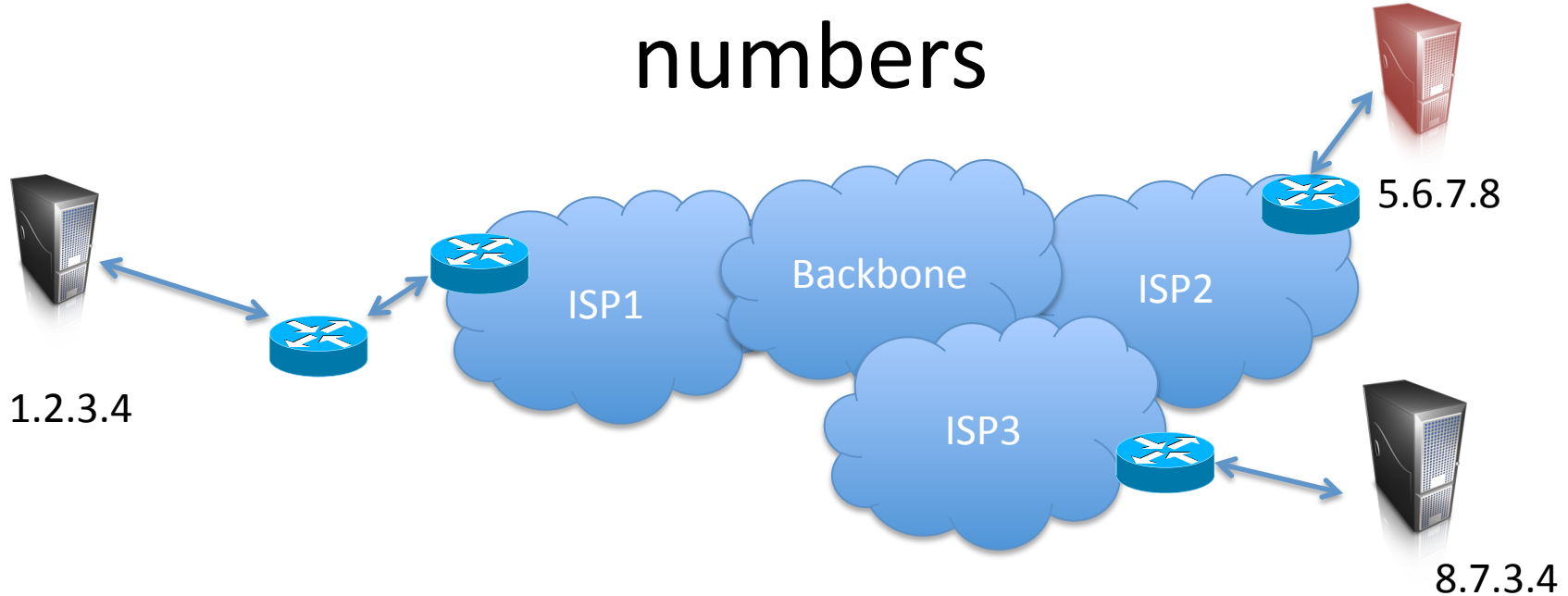
TCP handshake



How are $secC$ and $seqS$ selected?

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

Predictable sequence numbers

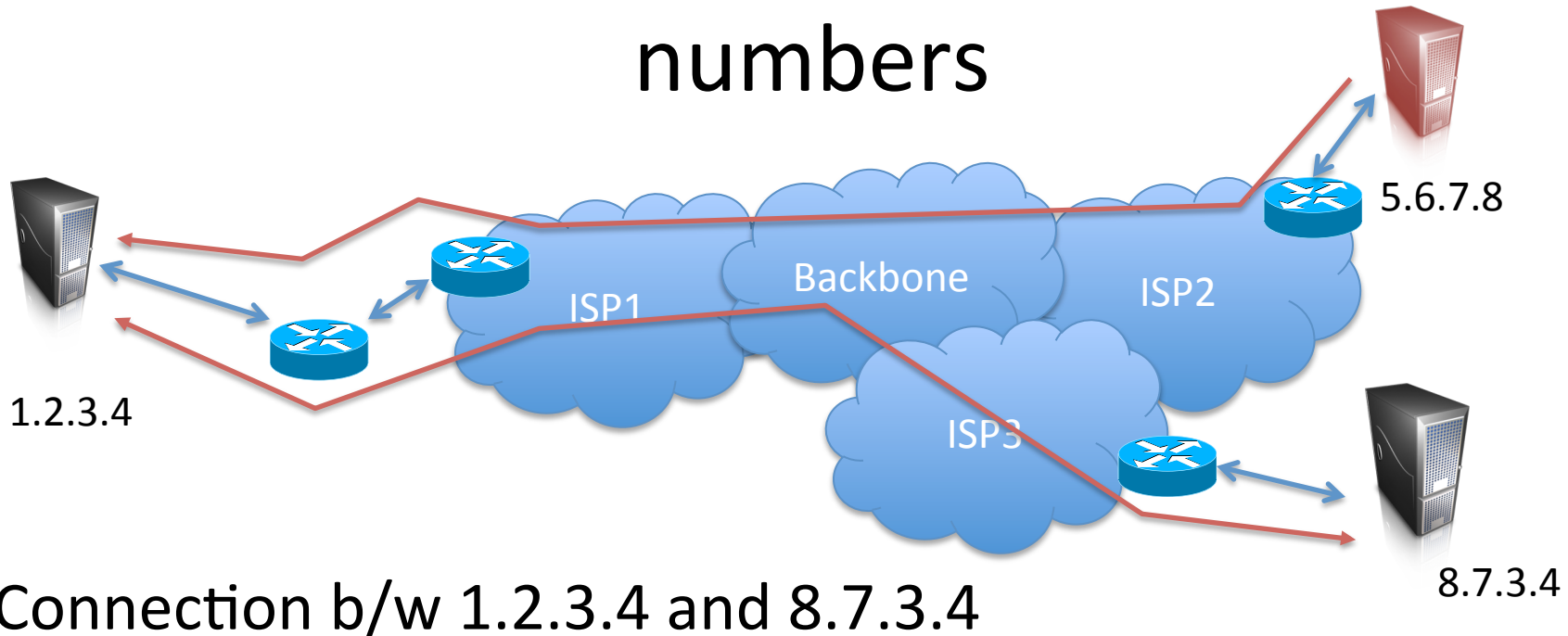


4.4BSD used predictable initial sequence numbers (ISNs)

- 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

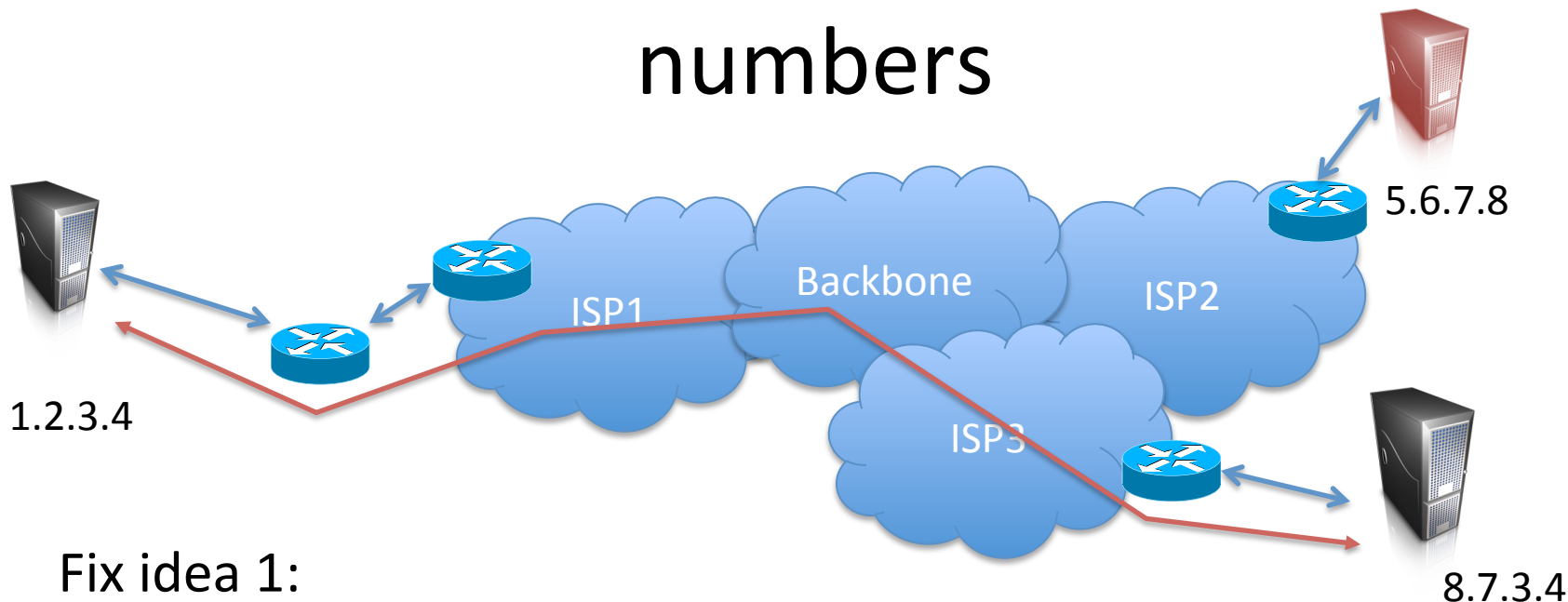
Forge a FIN packet from 8.7.3.4 to 1.2.3.4

```
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

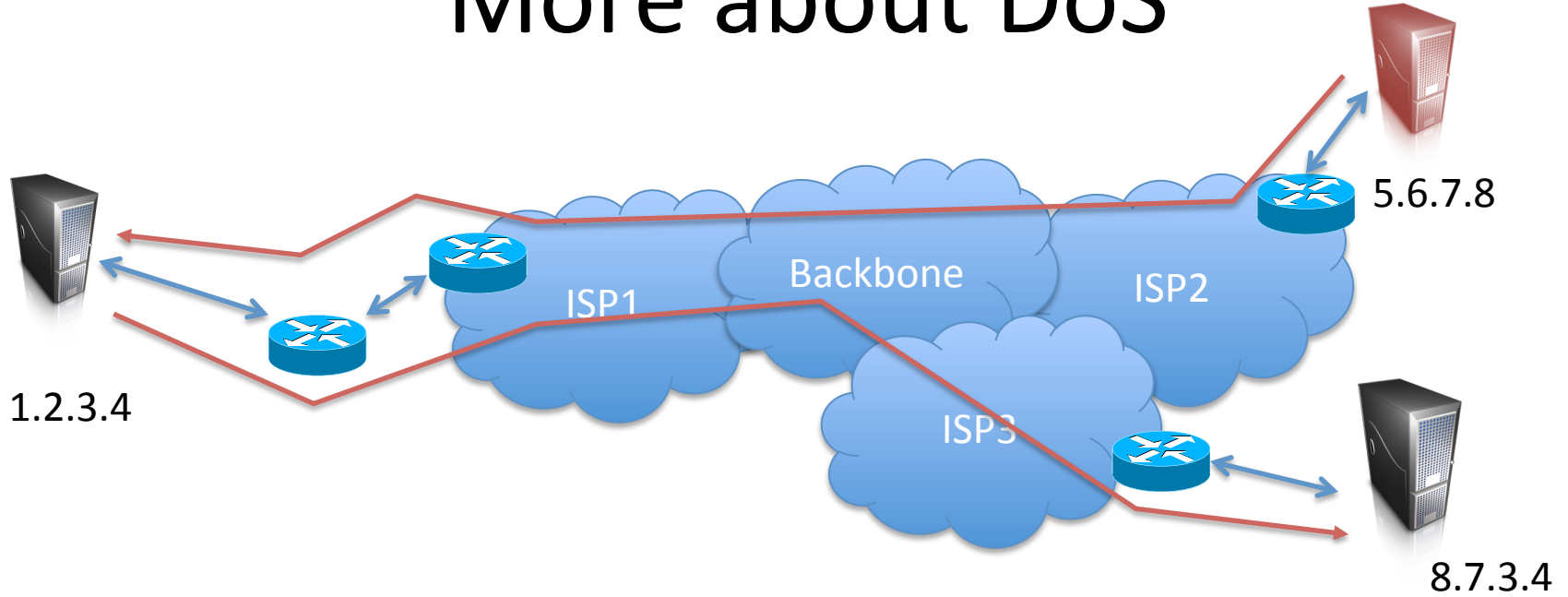
Still issues:

- 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

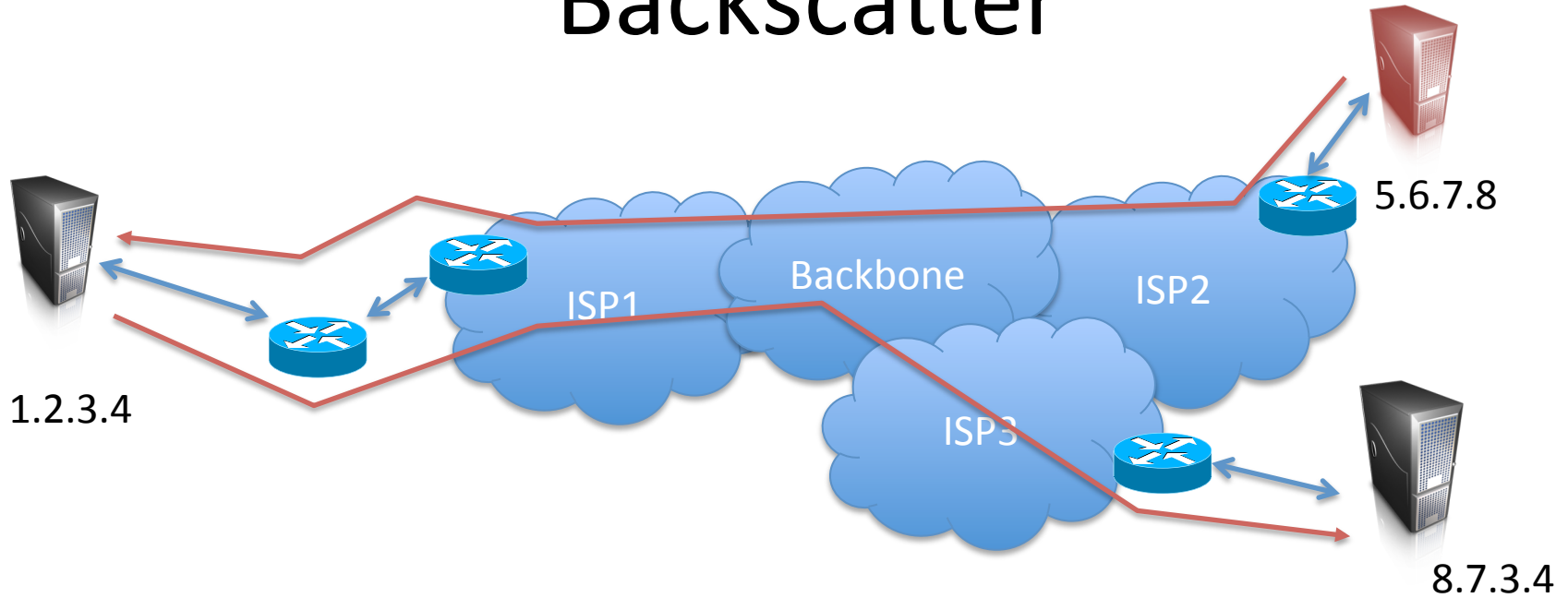
More about DoS



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

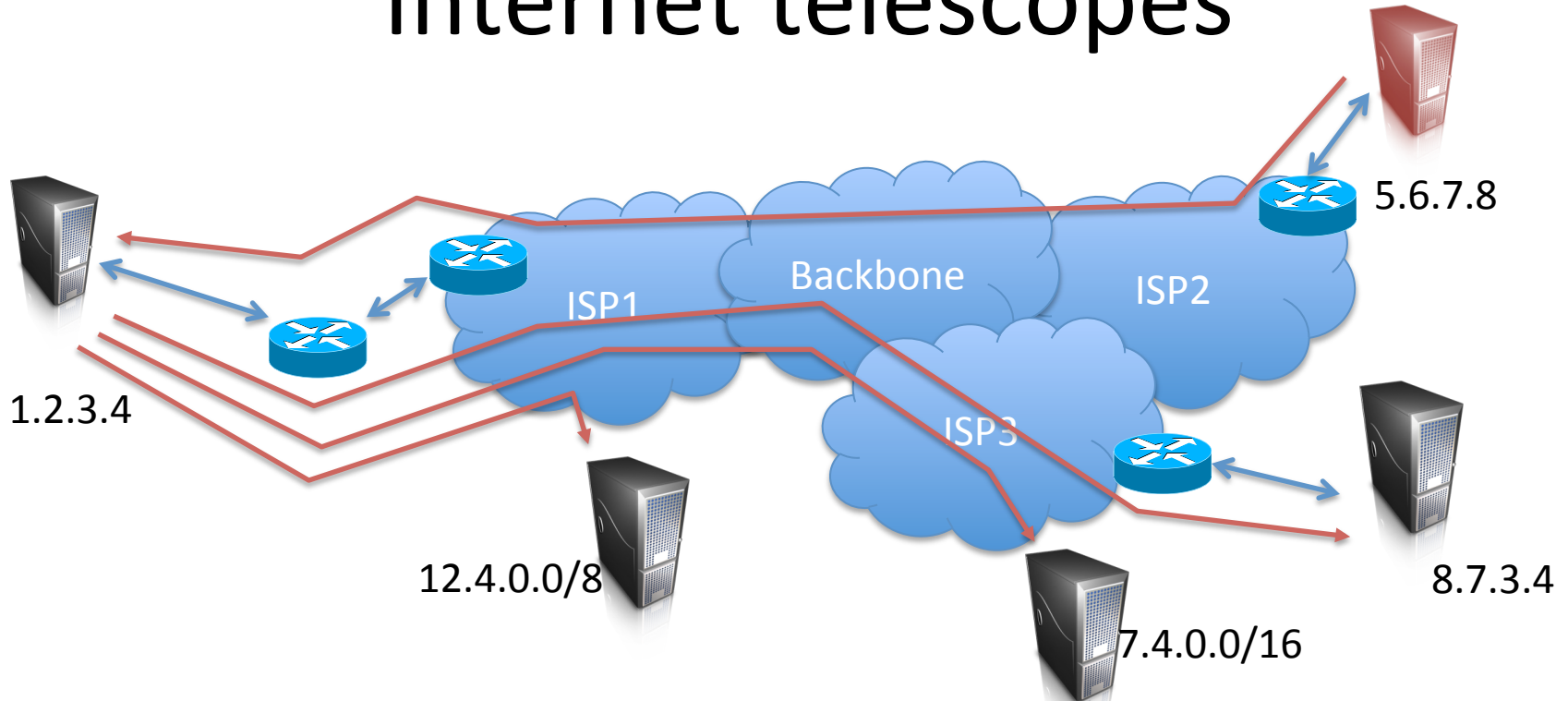
Types of responses to floods

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.

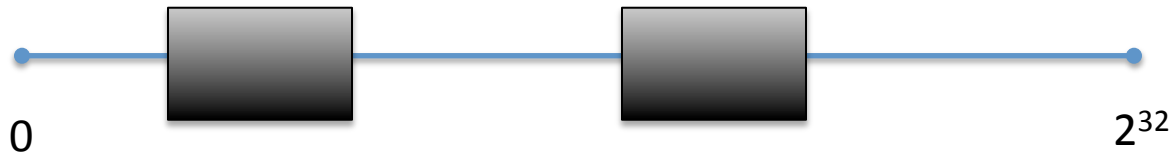
From Moore et al., "Inferring Internet Denial-of-Service Activity"

Internet telescopes



Setup some computers to watch traffic sent to darknets

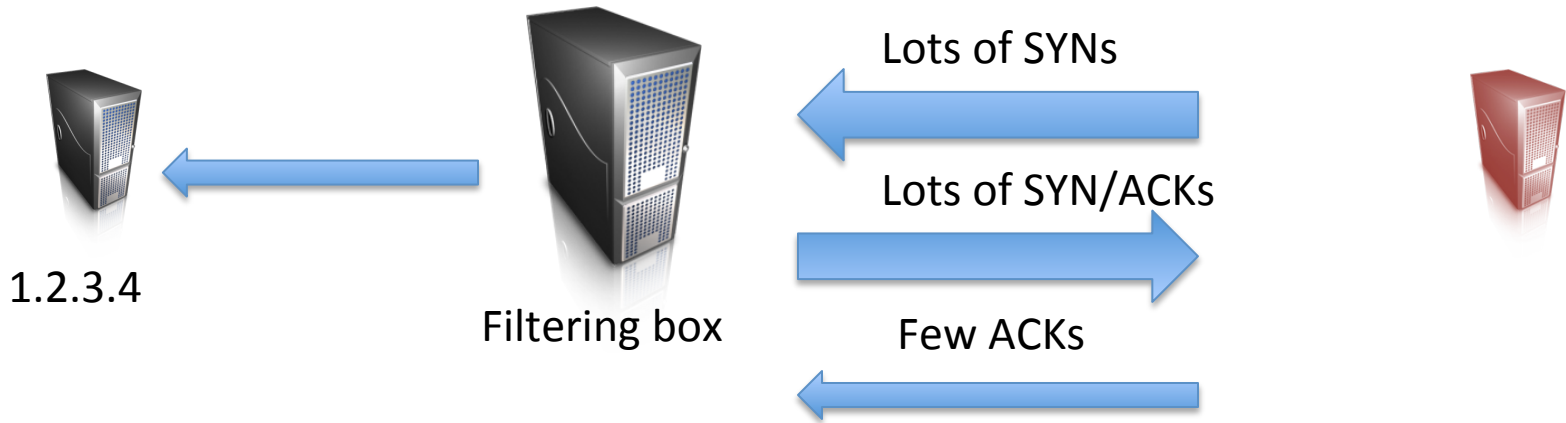
- Darknet = unused routable space



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

Distributed DoS

- Botnets change the game
 - What have we seen today that changes?
- Estonia
 - First “cyber” war
 - sustained DDoS for over 10 hours
 - Estonian solution: stop foreign connections

