January 25, 2016

Denial of Service

Computer Security 2 (GA02)

Emiliano De Cristofaro, Gianluca Stringhini

Thanks to Aurelien Francillon for
letting us re-use some of the slides

Reading

Stallings 21.5

What?

- Denial of Service, in short DoS
- Goal (informal): **to** take out a large site with little computing work
- How: **Amplification**
  - Use small number of packets $\Rightarrow$ obtain a big effect
- (Roughly speaking) two types of amplification attacks:
  - DoS bug:
    - Design flaw allowing one machine to disrupt a service
  - DoS flood:
    - Command botnet to generate flood of requests

A high profile example: Estonia

- Attacked sites: (started April 2007, lasted two weeks)
  - Estonian ministerial sites
  - Various Estonian commercial sites

DoS can happen at any layer

- In this lecture, you'll learn about
  - DoS at different layers
  - Generic DoS solutions
  - Network DoS solutions

- Sad truth:
  - Internet not designed to handle DoS and Distributed DoS (DDoS) attacks

(Challenges → Opportunities ☺)

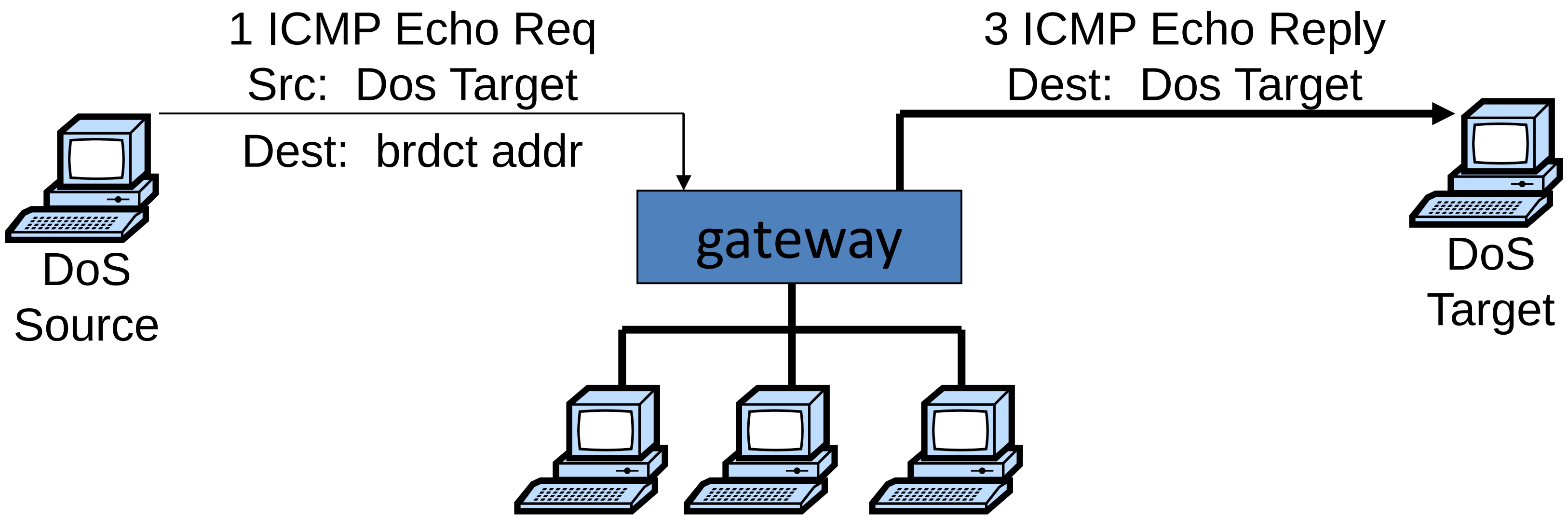
802.11b DoS bugs

- Wireless radio jamming attacks:
  - Trivial, not our focus.

- Protocol DoS bugs: [Bellardo, Savage, 2003]
  - NAV (Network Allocation Vector):
    - 15-bit field, max value 2^15-1=32767
    - Any node can reserve channel for #NAV seconds
    - No one else should transmit during NAV period

  - De-authentication bug:
    - Any node can send de-auth packet to AP
    - Deauth packet unauthenticated
    - … attacker can repeatedly deauth (i.e., kick out) anyone

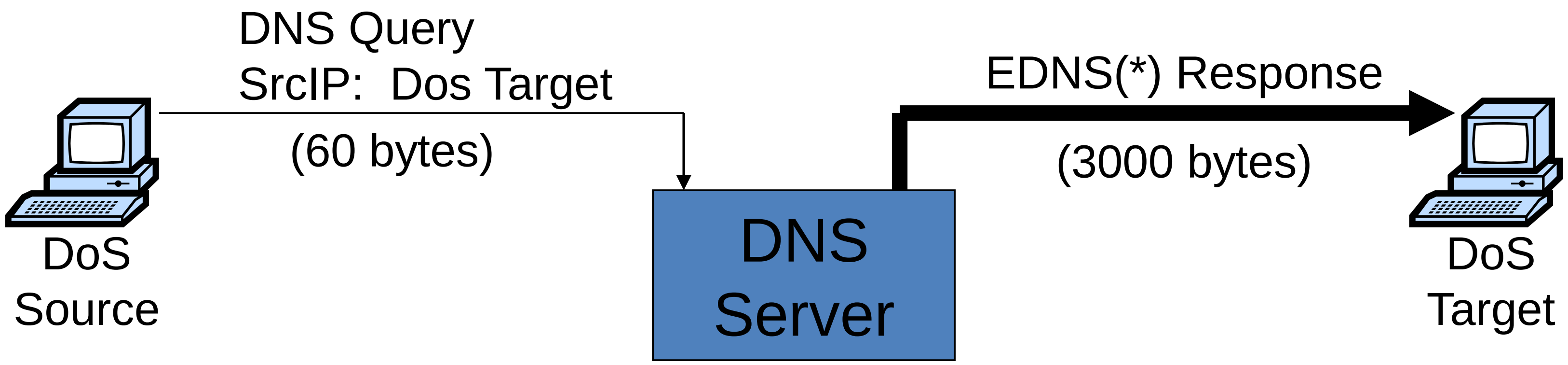
Smurf amplification DoS attack



- 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

DNS Amplification Attacks

×50 amplification



(*) EDNS -> Extension mechanisms for DNS, i.e., a way to add parameters to DNS messages (due to prior size restricions)

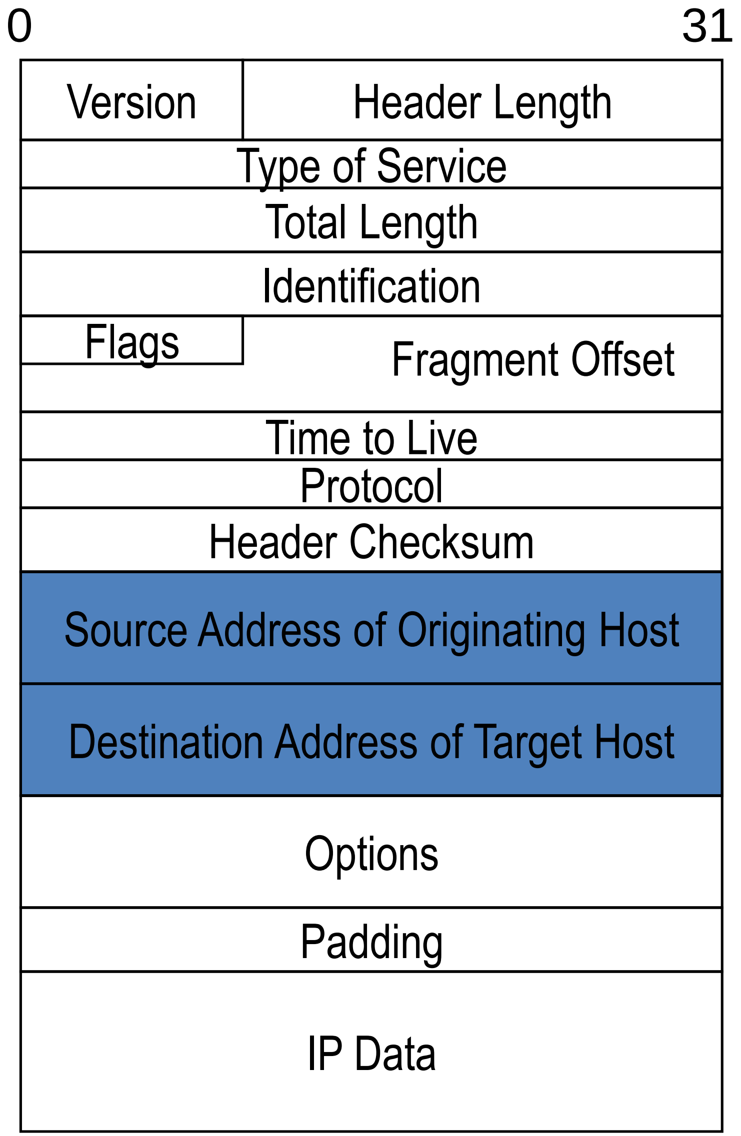
2006: 0.58M open resolvers on Internet (Kaminsky-Shiffman)

2013: 21.7M open resolvers (openresolverproject.org)

⇒ 3/2013: DDoS attack generating 300 Gbps

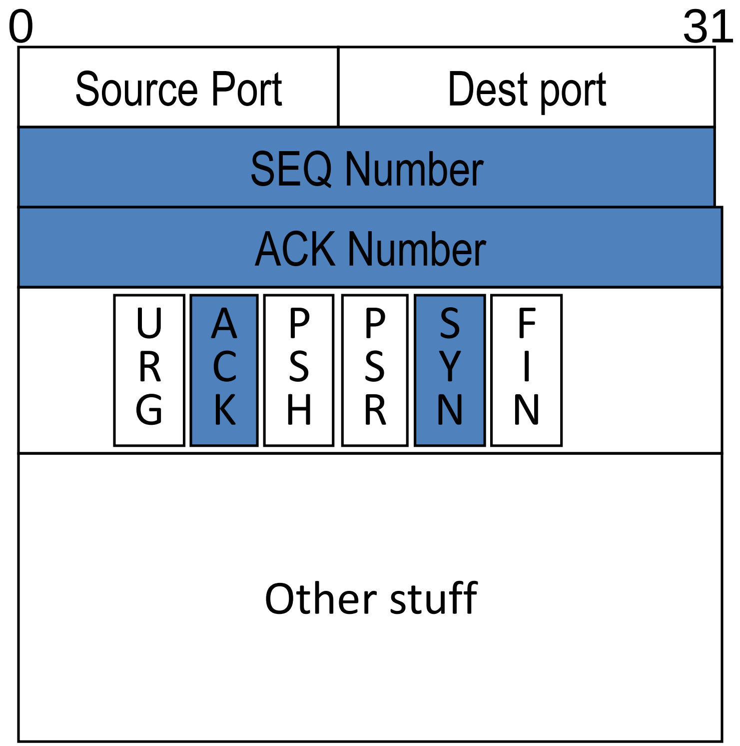
Review: IP Header format

- Connectionless
  - Unreliable
  - Best effort

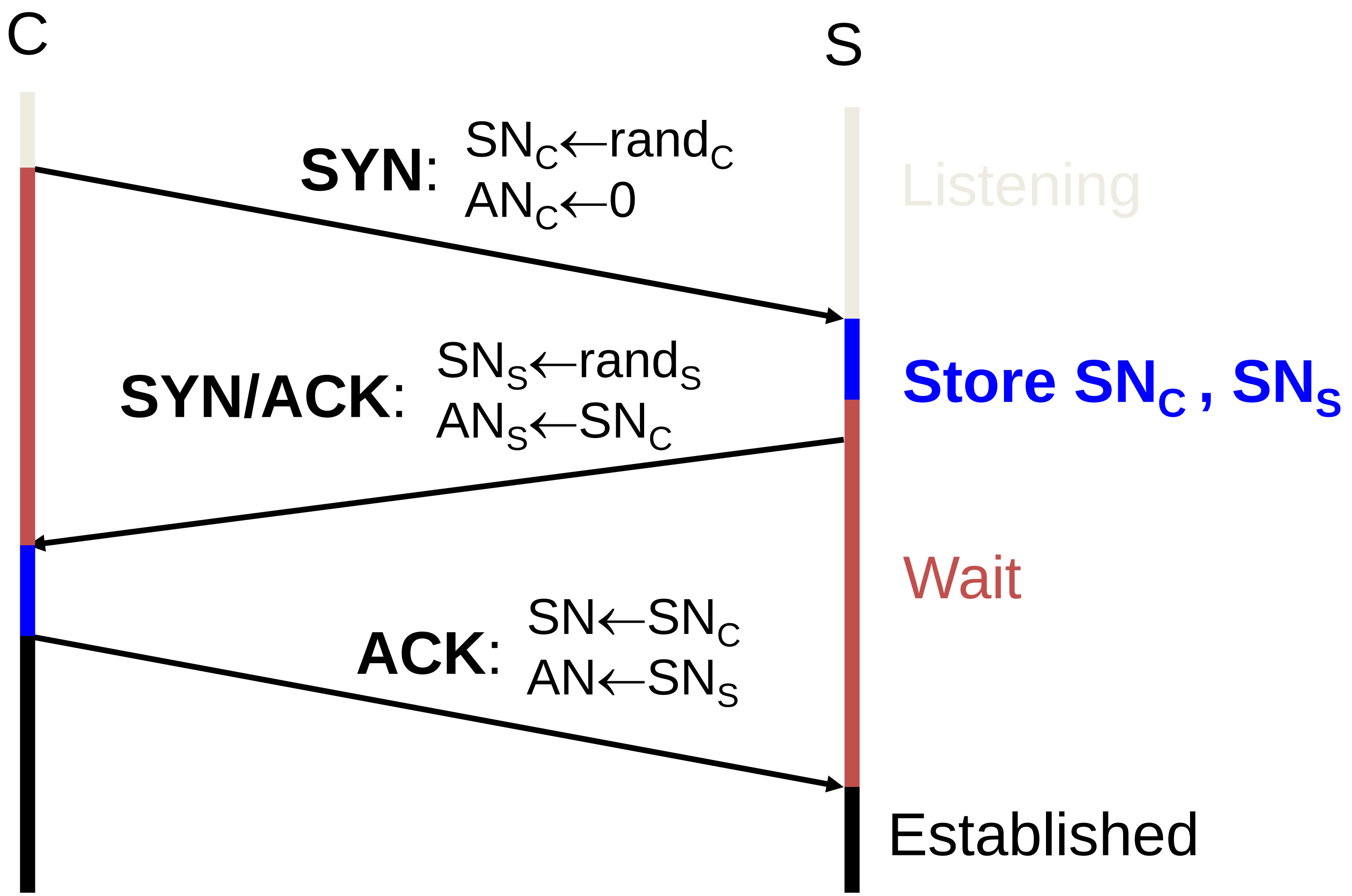


Review: TCP Header format

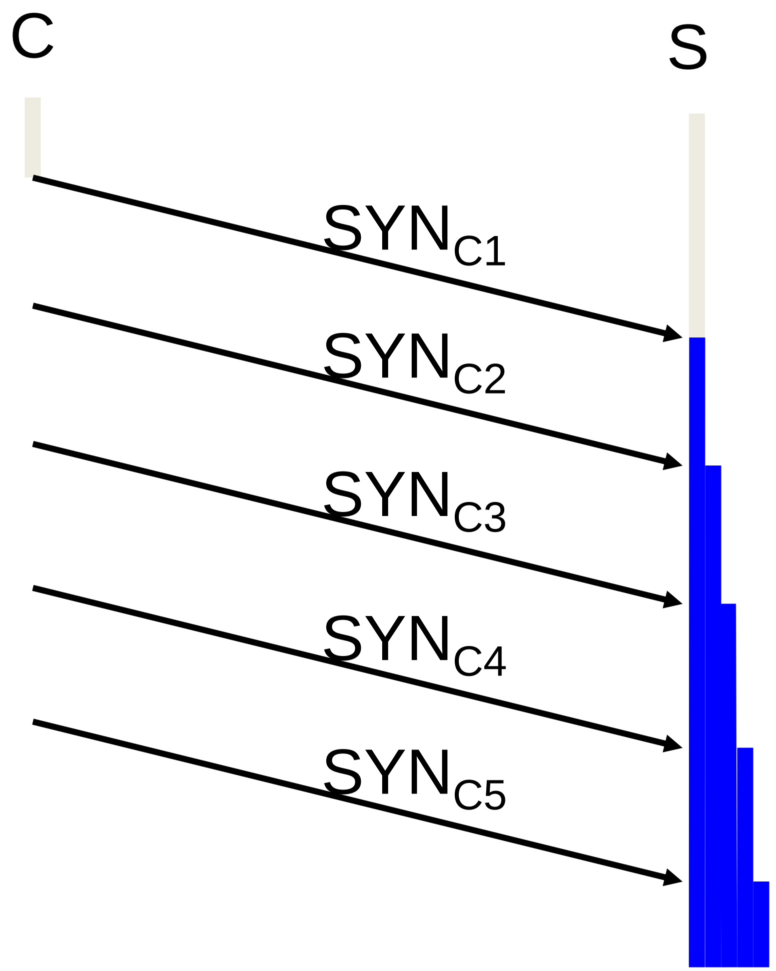
- TCP:
  – Session based
  – Congestion control
  – In order delivery



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

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

Backlog timeout: 3 minutes

Attacker needs 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

- 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

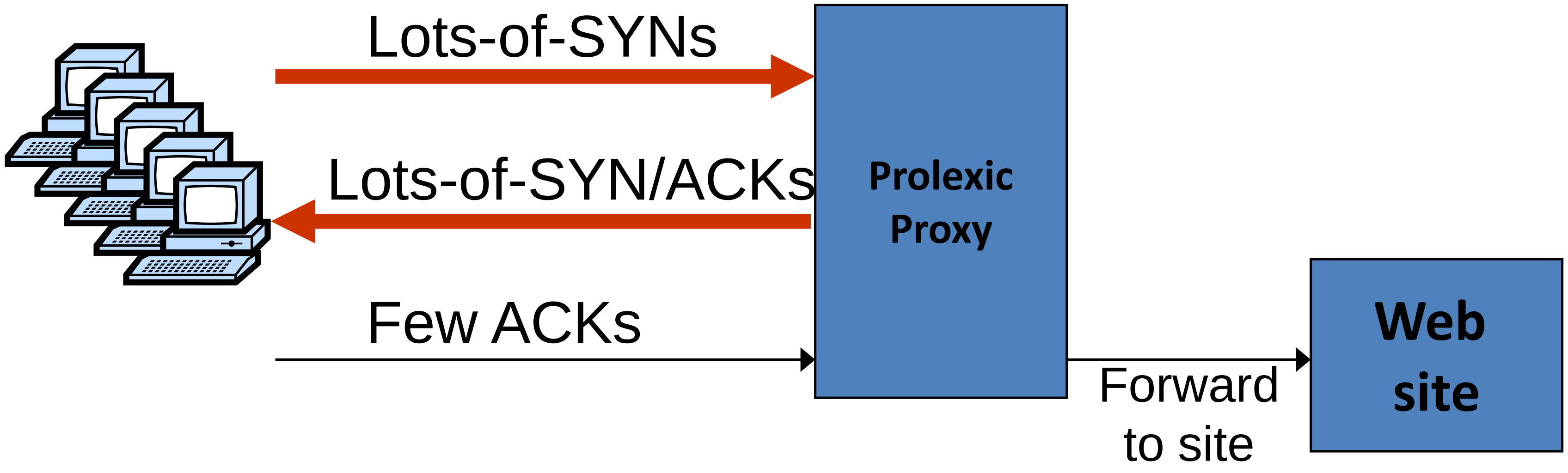
SYN Floods II: Massive flood

- E.g., BetCris.com, 2003

- 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 $\Rightarrow$
      attack SYNs look the same as real SYNs
  - What to do ???

Prolexic / Verisign Design

- Idea: only forward established TCP connections to site



Estonia attack

- 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: block all foreign traffic until attacks stopped

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.

DNS DoS Attacks (Bluesecurity, 2006)

- 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:
  - Bluesecurity DNS hosted at Tucows DNS server
  - DNS DDoS took out Tucows hosting many many sites

- What to do ???

Root level DNS attacks

- Feb. 6, 2007:
  - Botnet attack on the 13 Internet DNS root servers
  - Lasted 2.5 hours
  - None crashed, but two performed badly:
    - g-root (US Department of Defense)
    - l-root (ICANN)
- Attack in Oct. 2002 took out 9 of the 13 TLD servers

DNS DoS solutions

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

- DoS resistant DNS design:
  - **CoDoNS**: [Sire, 2004]
    - 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

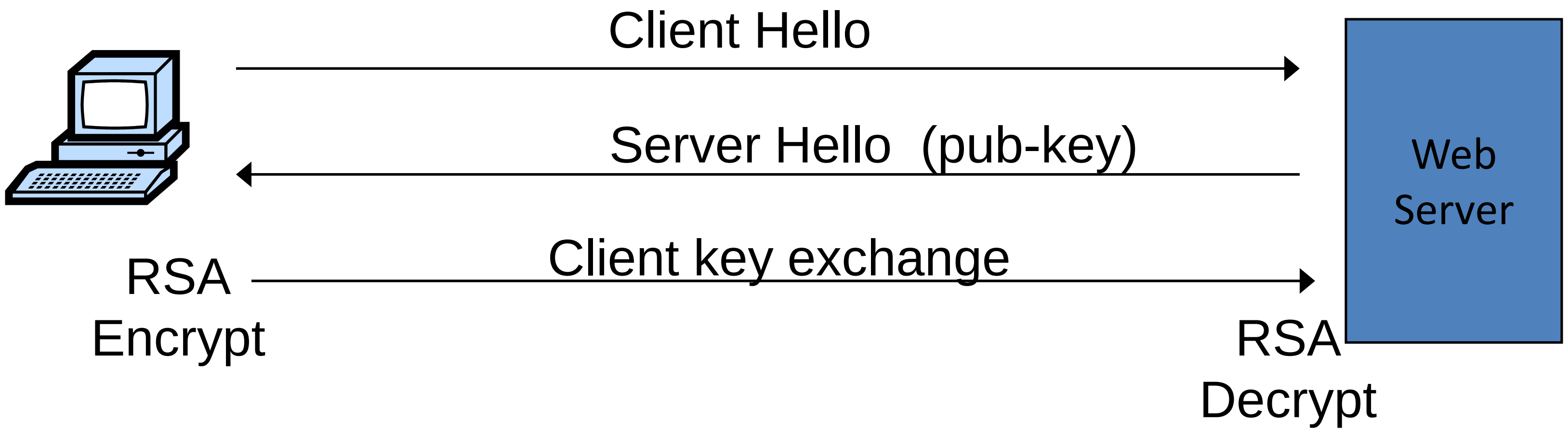
- YouTube is 208.65.152.0**/22** (includes 2^{10} IP addr)
  - youtube.com is 208.65.153.238, …
- Feb. 2008:
  - Pakistan telecom advertised a BGP path for

    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, 203]



RSA-encrypt speed $\approx$ 10$\times$ RSA-decrypt speed

$\Rightarrow$ Single machine can bring down ten web servers

Similar problem with application DoS:

- Send HTTP request for some large PDF file

$\Rightarrow$ 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

    $$\mathrm{LSB}_n\big(\,\text{SHA-1}(\,C \,\|\, X\,)\,\big) = 0^n$$

  - Assumption: takes expected 2^n time to solve
  - For n=16 takes about 0.3sec 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, 1999)
  - Example challenge: C = TCP server-seq-num
  - First data packet must contain puzzle solution
    - Otherwise TCP connection is closed

- SSL handshake DoS: (SD, 2003)
  - 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

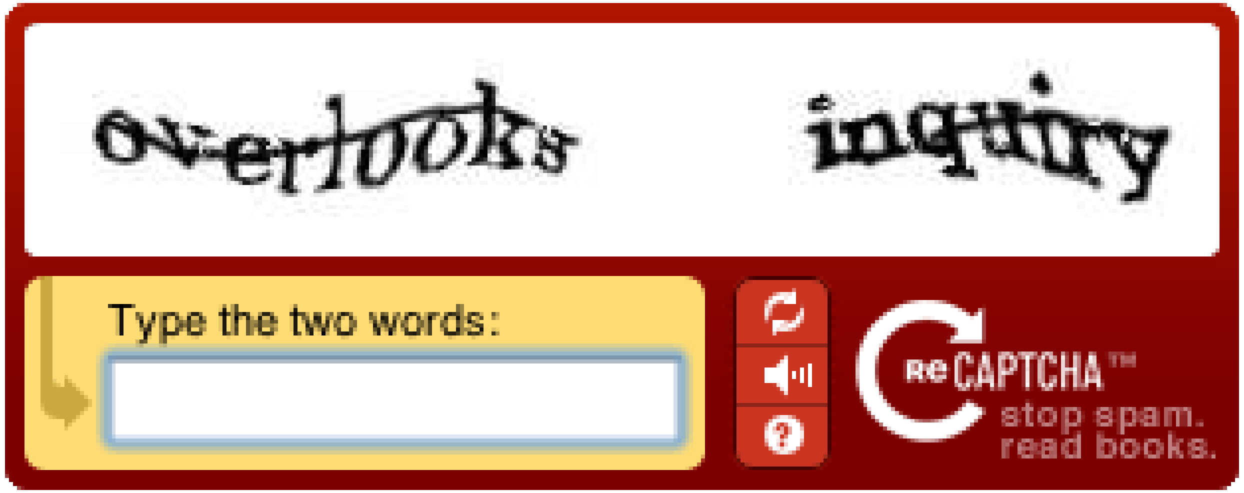
- CPU power ratio:
  - High end server / low end cell phone = 8000
  - Impossible to scale to hard puzzles

- 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 2003
    - Abadi-Burrows-Manasse-Wobber, ACM ToIT 2005

2. CAPTCHAs

- Idea: verify that connection is from a human



- Applies to application layer DDoS [Killbots 2005]
  - During attack: generate CAPTCHAs and process request only if valid solution
  - Present one CAPTCHA per source IP address.

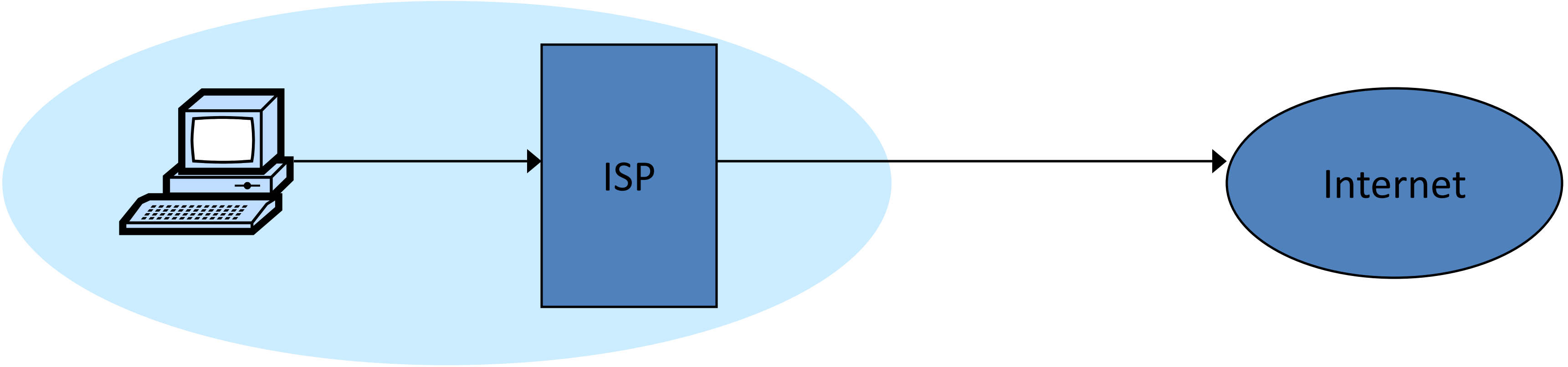
3. Source identification

Goal: identify packet source

Ultimate goal: block attack at the source

3a. Ingress filtering (RFC 2827, 2000)

- Big problem: DDoS with spoofed source IPs
- Question: how to find packet origin?



- Ingress filtering policy: ISP only forwards packets with legitimate source IP

Implementation problems

- ALL ISPs must do this. Requires global trust.
  - If 10% of ISPs do not implement $\Rightarrow$ no defense

3b. Traceback [Savage et al., 2000]

- 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

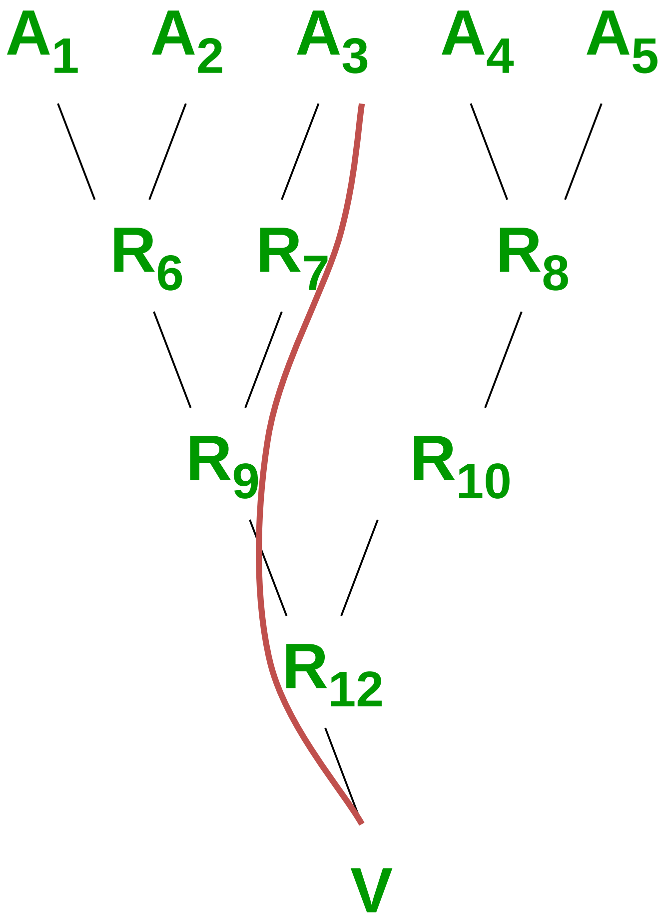
Simple method

- 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

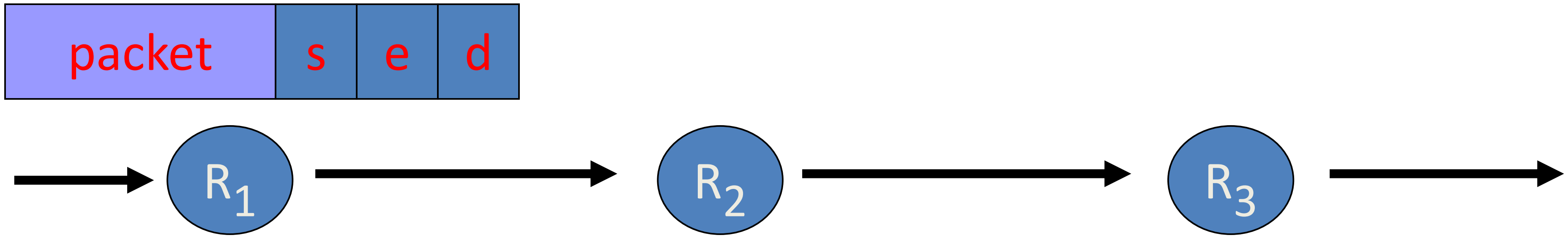
- Data fields written to packet:
  - Edge: *start* and *end* IP addresses
  - Distance: number of hops since edge stored

- 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
```

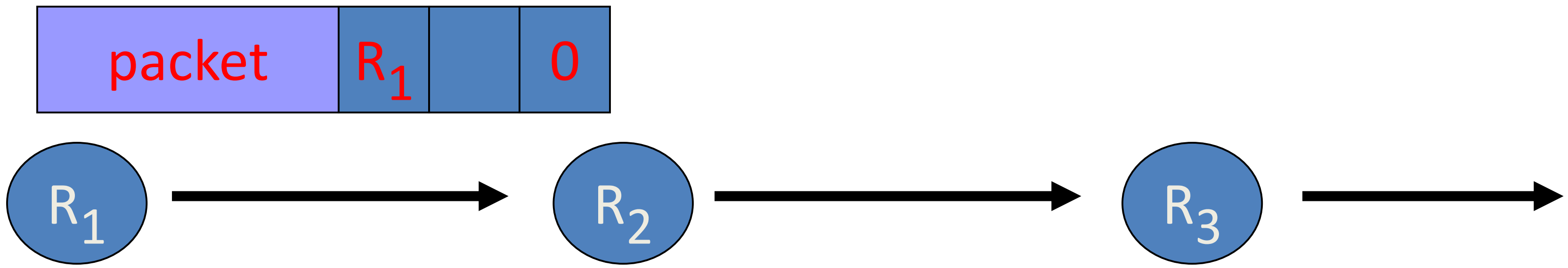
Edge Sampling: picture

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



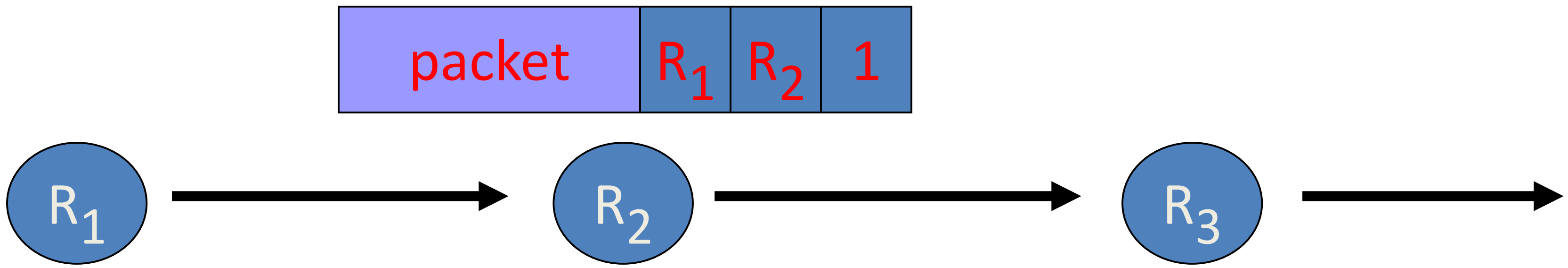
Edge Sampling: picture

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



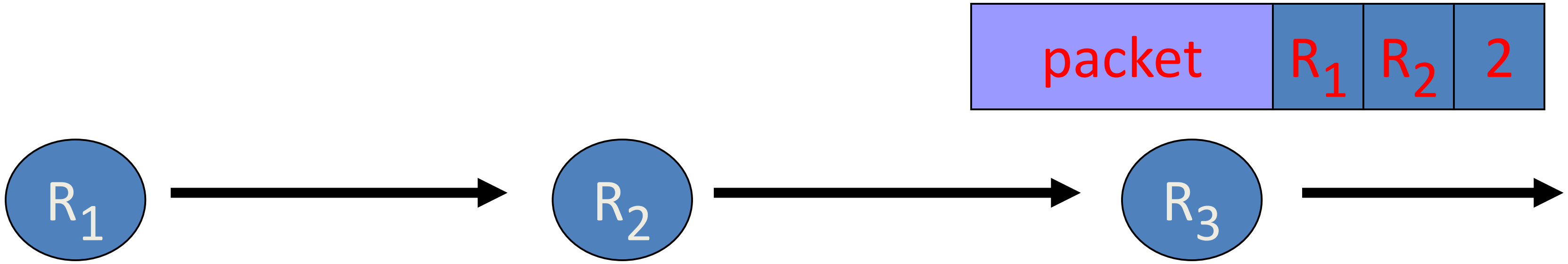
Edge Sampling

- Finish writing edge
  - R2 chooses not to overwrite edge
  - Distance is 0
    - Write end of edge, increment distance to 1



Edge Sampling

- Increment distance
  - R3 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

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

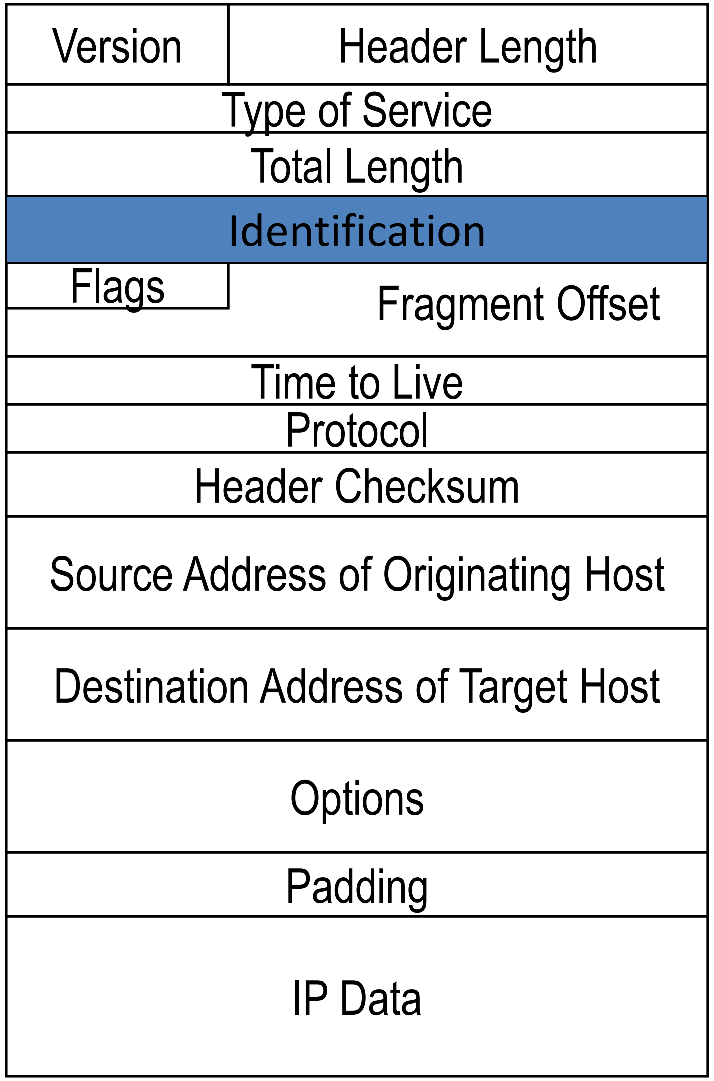
where p is marking probability, d is length of path

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



Capability based defense

Capability based defense

- Anderson, Roscoe, Wetherall.
  - Preventing internet denial-of-service with capabilities. SIGCOMM, 2004.
- Yaar, Perrig, and Song.
  - Siff: A stateless internet flow filter to mitigate DDoS flooding attacks. IEEE S&P, 2004.
- Yang, Wetherall, Anderson.
  - A DoS-limiting network architecture. SIGCOMM, 2005

Pushback filtering

- Mahajan, Bellovin, Floyd, Ioannidis, Paxson, Shenker. Controlling High Bandwidth Aggregates in the Network. *Computer Communications Review* 2002.

- Ioannidis, Bellovin.
  Implementing Pushback: Router-Based Defense Against DoS Attacks. *NDSS* 2002

- Argyraki, Cheriton.
  Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. USENIX 2005.

Overlay filtering

- Keromytis, Misra, Rubenstein.
  SOS: Secure Overlay Services.   SIGCOMM 2002.

- D. Andersen. Mayday.
  Distributed Filtering for Internet Services.
  Usenix USITS 2003.

- Lakshminarayanan, Adkins, Perrig, Stoica.
  Taming IP Packet Flooding Attacks. HotNets 2003.

Take home message:

- Denial of Service attacks are real and must be considered at design time

- Sad truth:
  - Current Internet is ill-equipped to handle DDoS attacks

- Many good proposals for core redesign…

Conclusion

- Denial of Service
  - Attempt to defeat availability
    - E.g., user is denied access to service or data
- Flooding or Overload
  - Presenting commands more quickly than a server can handle them
    - Targeting applications or resources
- Blocked access / access failure
  - Preventing a service from functioning

Vulnerabilities

- Internet not designed to handle DoS…
- Insufficient resources (e.g., capacity)
- Traffic redirection

More… ?