Unwanted Traffic: Denial of Service and Spam email

Dan Boneh

What is network DoS?

- Goal: take out a large site with little computing work
- How: **Amplification**
  - Small number of packets $\Rightarrow$ 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

A high profile example: Estonia

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

(more on this later)

DoS can happen at any layer

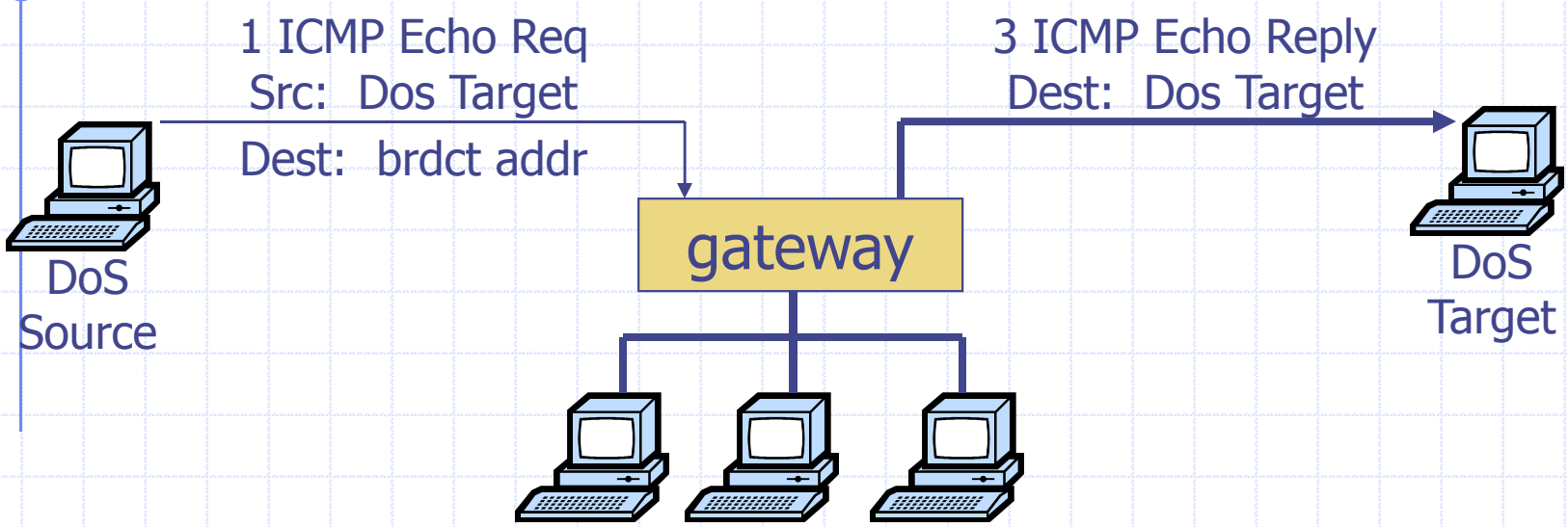
- This lecture:
  - Sample Dos at different layers (by order):
    - Link
    - TCP/UDP
    - Application
    - Payment
  - Generic DoS solutions
  - Network DoS solutions
- Sad truth:
  - Current Internet not designed to handle DDoS attacks

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

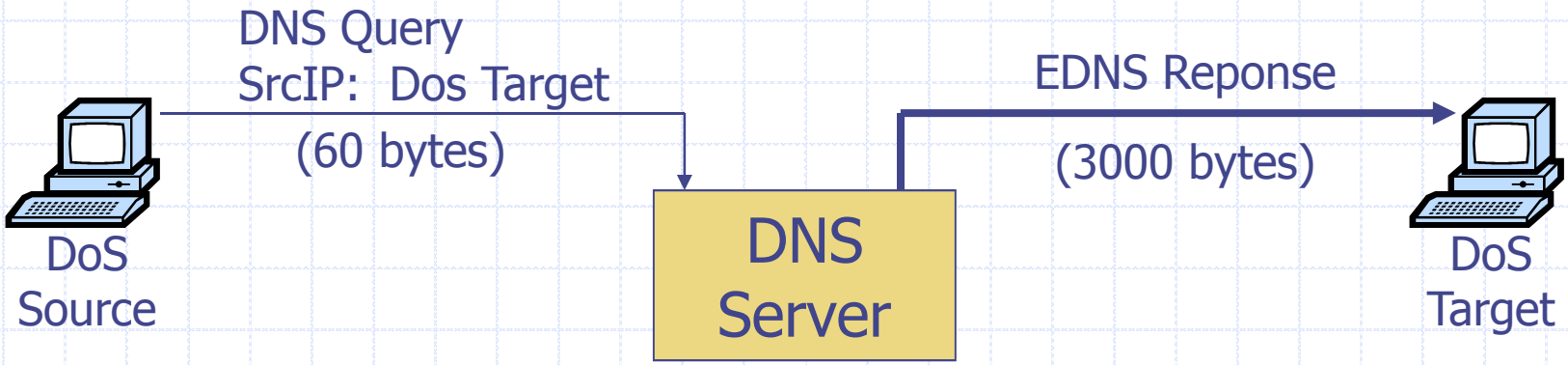


- Send ping request to broadcast addr (ICMP Echo Req)
- Lots of responses:
  - Every host on target network generates a ping reply (ICMP Echo Reply) to victim

Prevention: reject external packets to broadcast address

Modern day example (May '06)

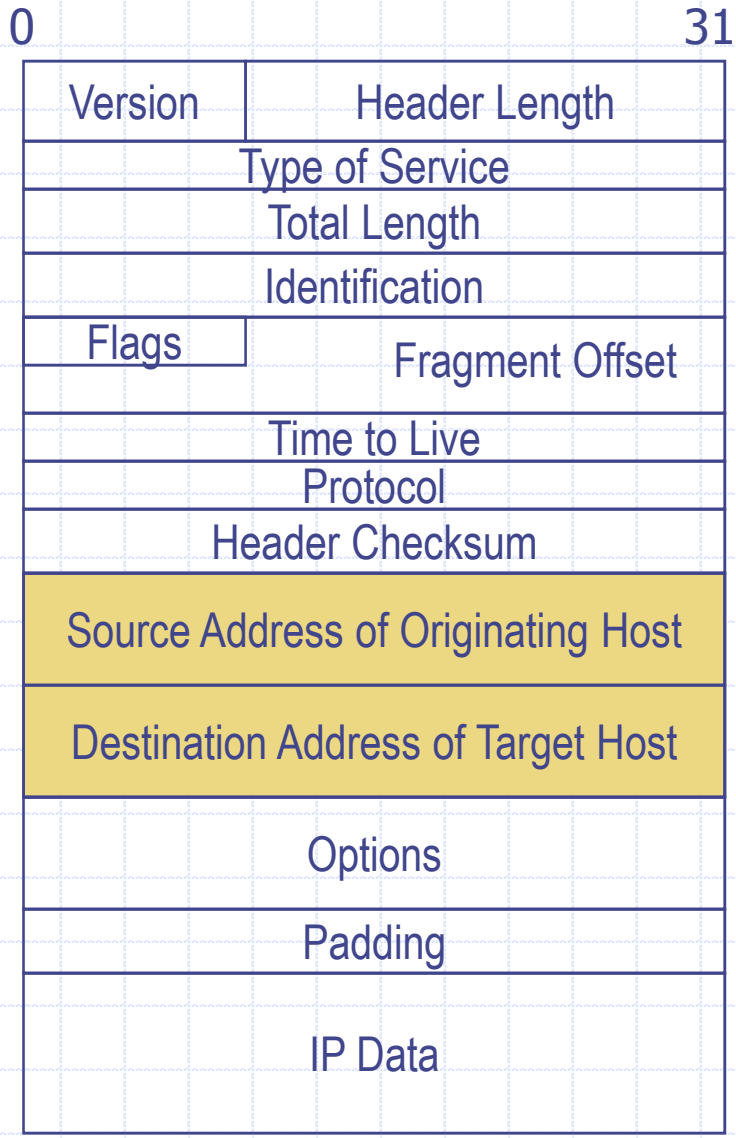
DNS Amplification attack: (×50 amplification)



580,000 open resolvers on Internet (Kaminsky-Shiffman'06)

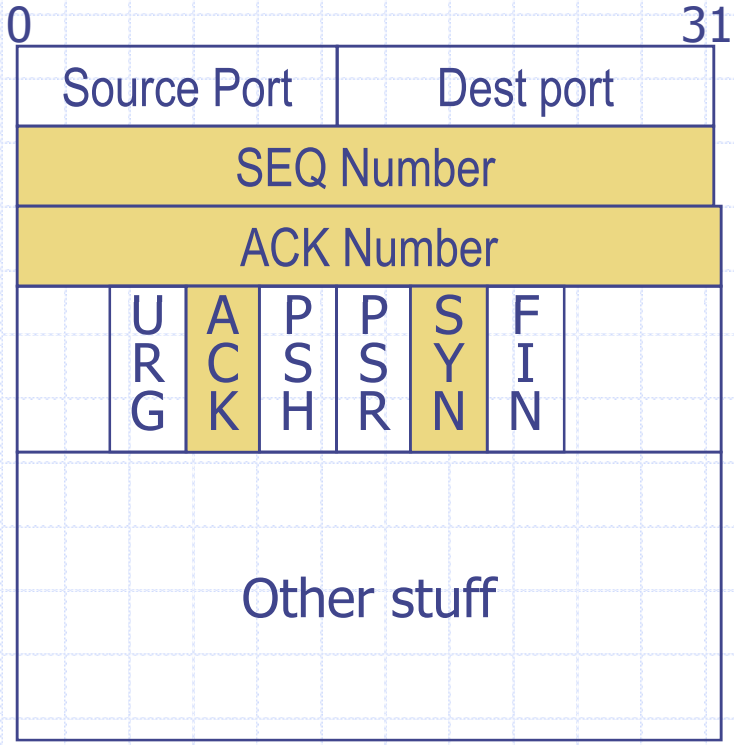
Review: IP Header format

- Connectionless
  - Unreliable
  - Best effort

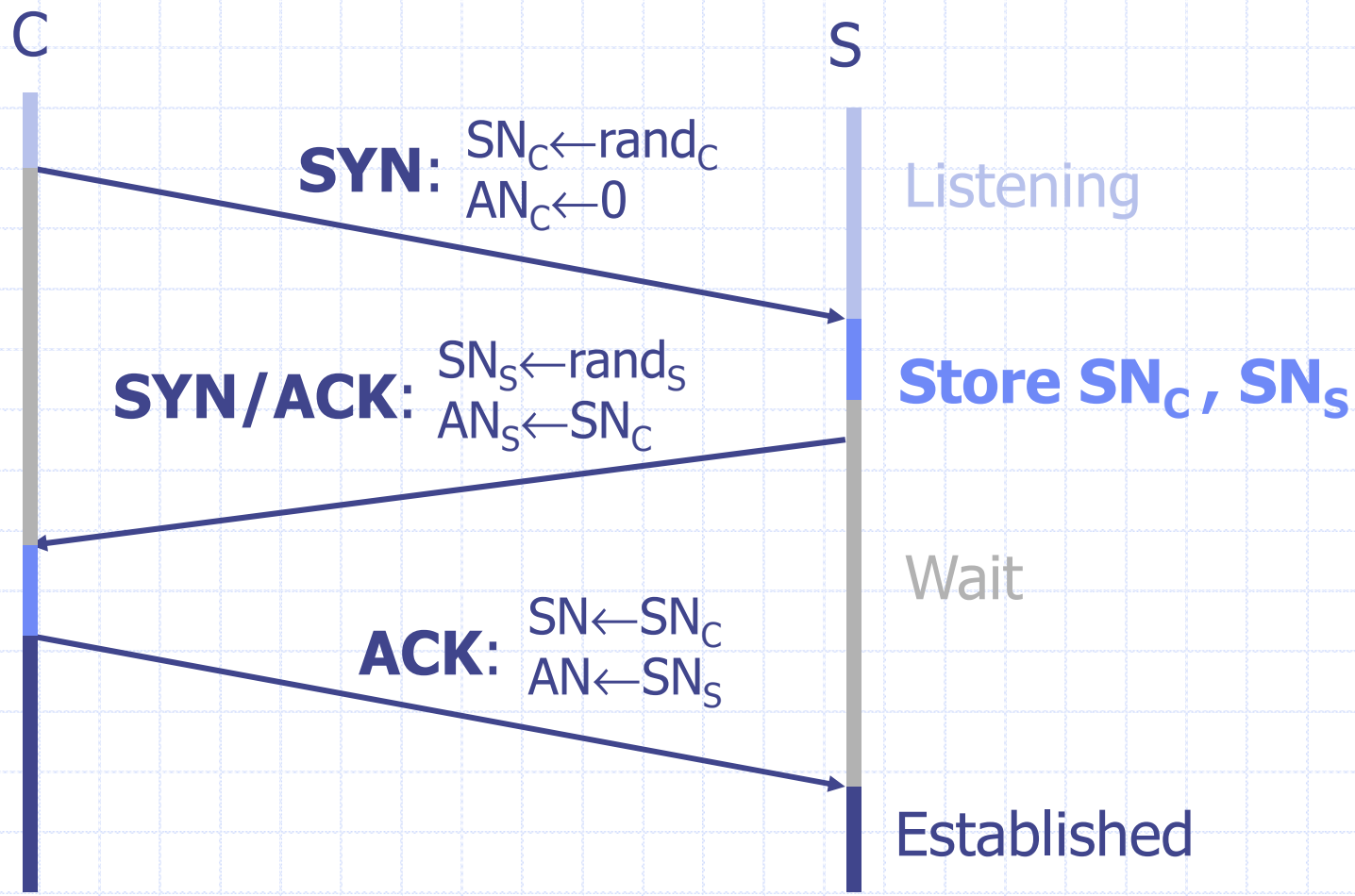


Review: TCP Header format

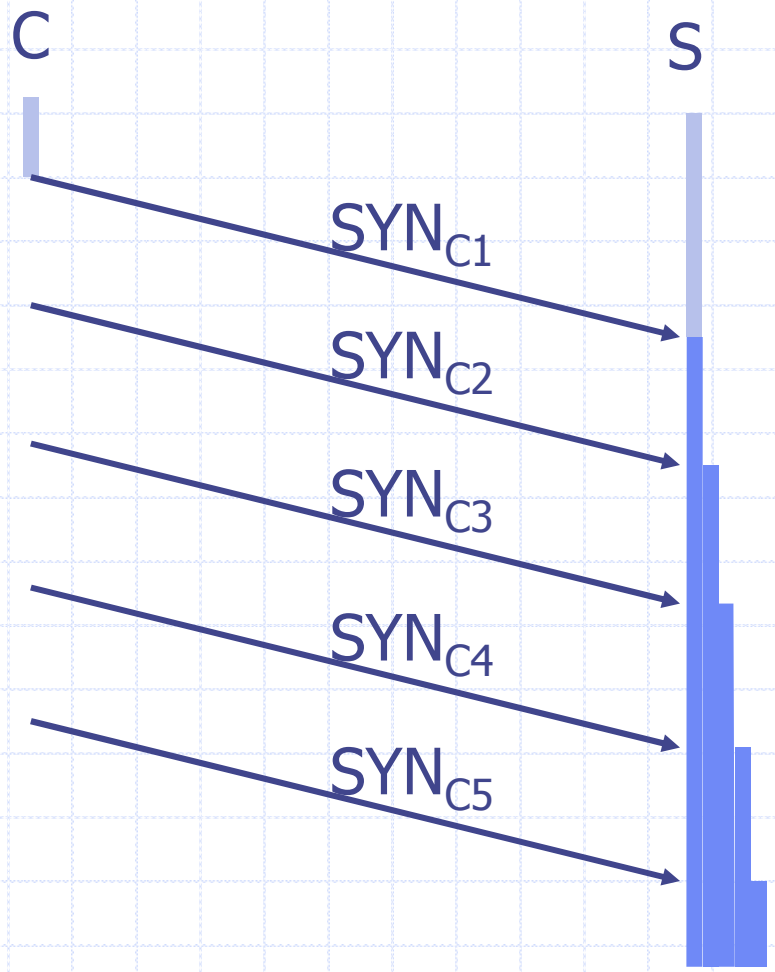
- TCP:
  - Session based
  - Congestion control
  - In order delivery



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

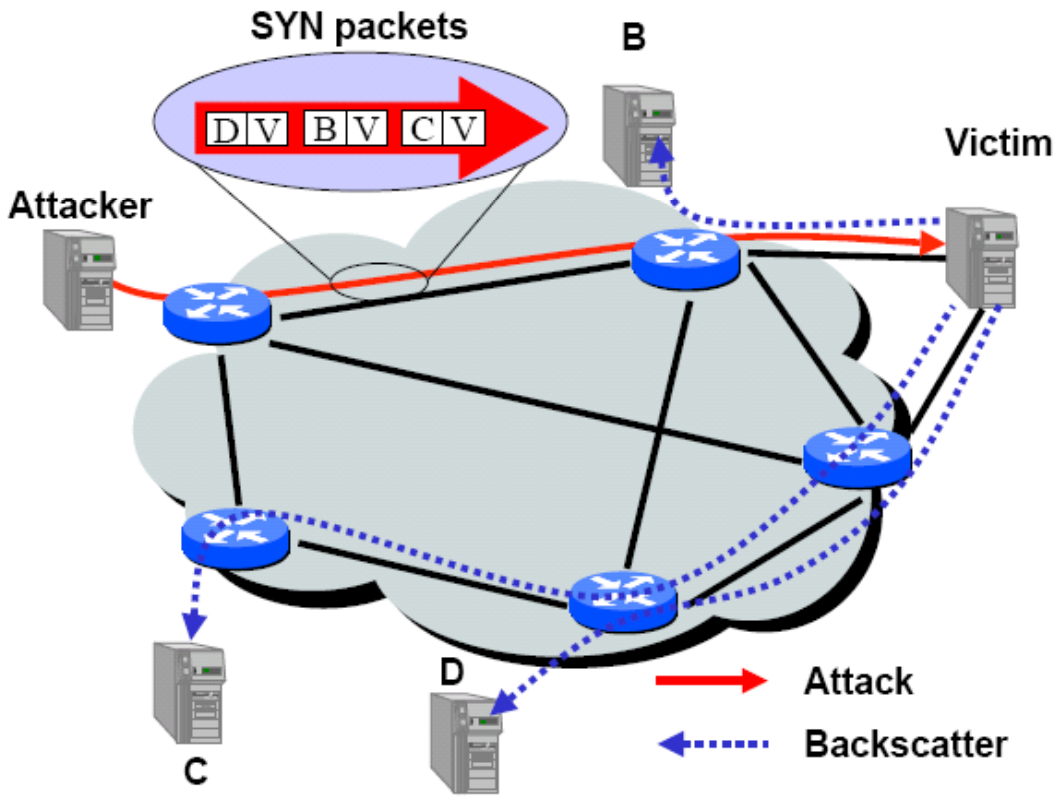
Syncookies

[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 ⇒ 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
- 2008: **4425** SYN attacks/24 hours (arbor networks ATLAS)

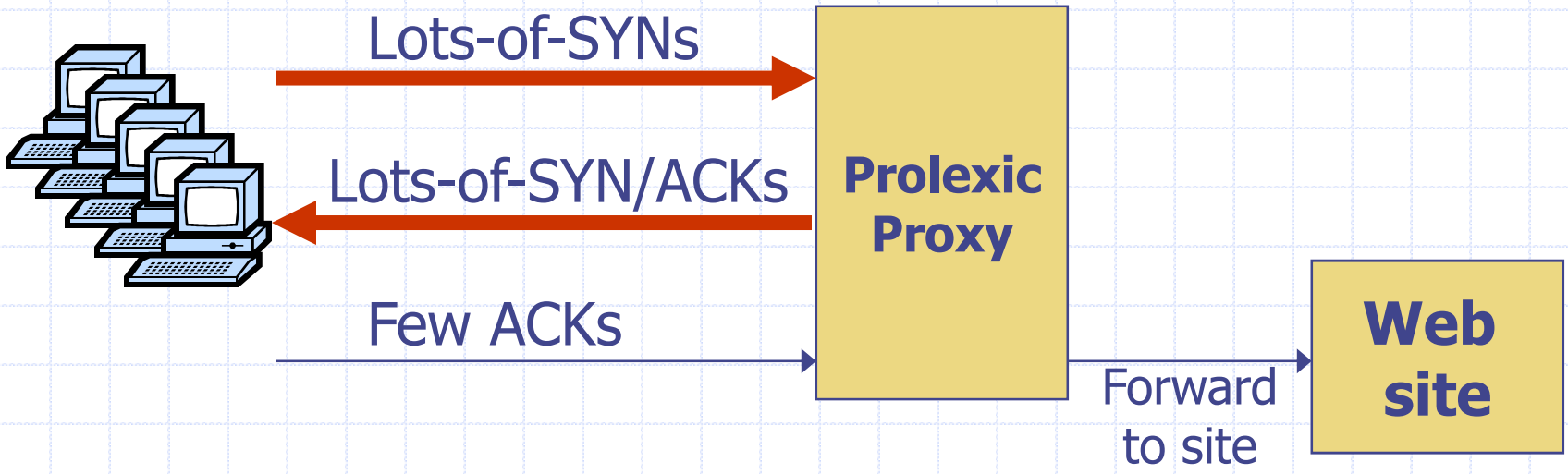
  - Larger experiments: (monitor many ISP darknets)
    - Arbor networks
    - Network telescope (UCSD)

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

Prolexic

- Idea: only forward established TCP connections to site



- Prolexic capacity: 20Gb/sec link

  can handle $40 \cdot 10^6$ SYN/sec

Other junk packets

Attack Packet	Victim Response	Rate (2008) [ATLAS]
TCP SYN to open port	TCP SYN/ACK	4425
TCP SYN to closed port	TCP RST	
TCP ACK or TCP DATA	TCP RST	
TCP RST	No response	276
TCP NULL	TCP RST	2821
ICMP ECHO Request	ICMP ECHO Response	**8352**
UDP to closed port	ICMP Port unreachable	

Proxy must keep floods of these away from web site

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

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 (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 ???

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 (DoD), l-root (ICANN)
    - Most other root servers use anycast

Attack in Oct. 2002 took out 9 of the 13 TLD servers

DNS DoS solutions

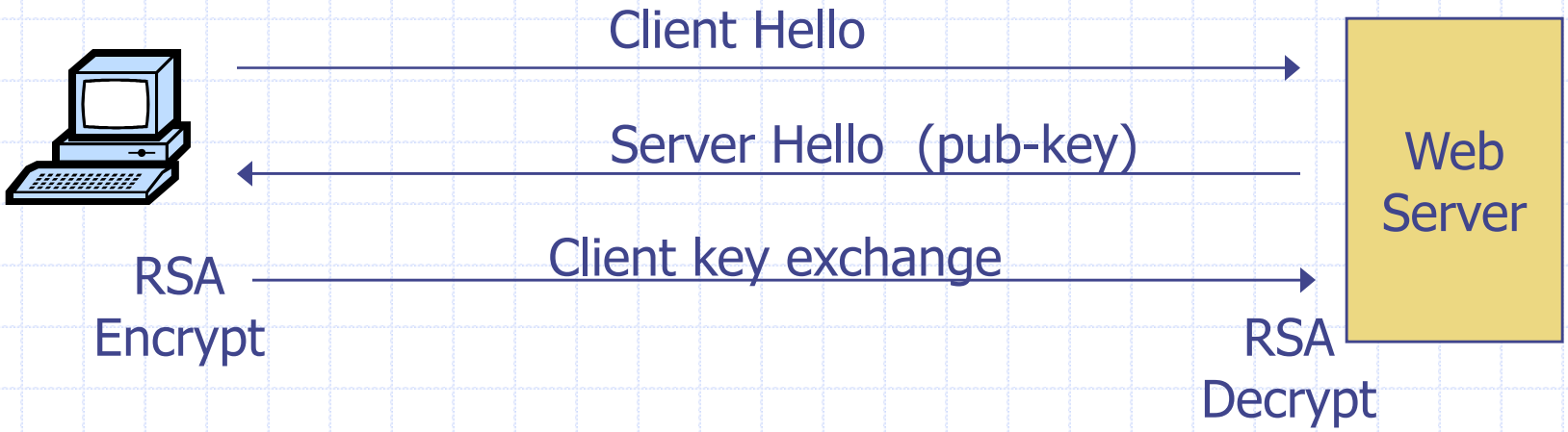
- Generic DDoS solutions:
  - Later on. Require major changes to DNS.
- DoS resistant DNS design:
  - **CoDoNS**: [Sirer'04]
    - Cooperative Domain Name System
  - P2P design for DNS system:
    - DNS nodes share the load
    - Simple update of DNS entries
    - Backwards compatible with existing DNS

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]



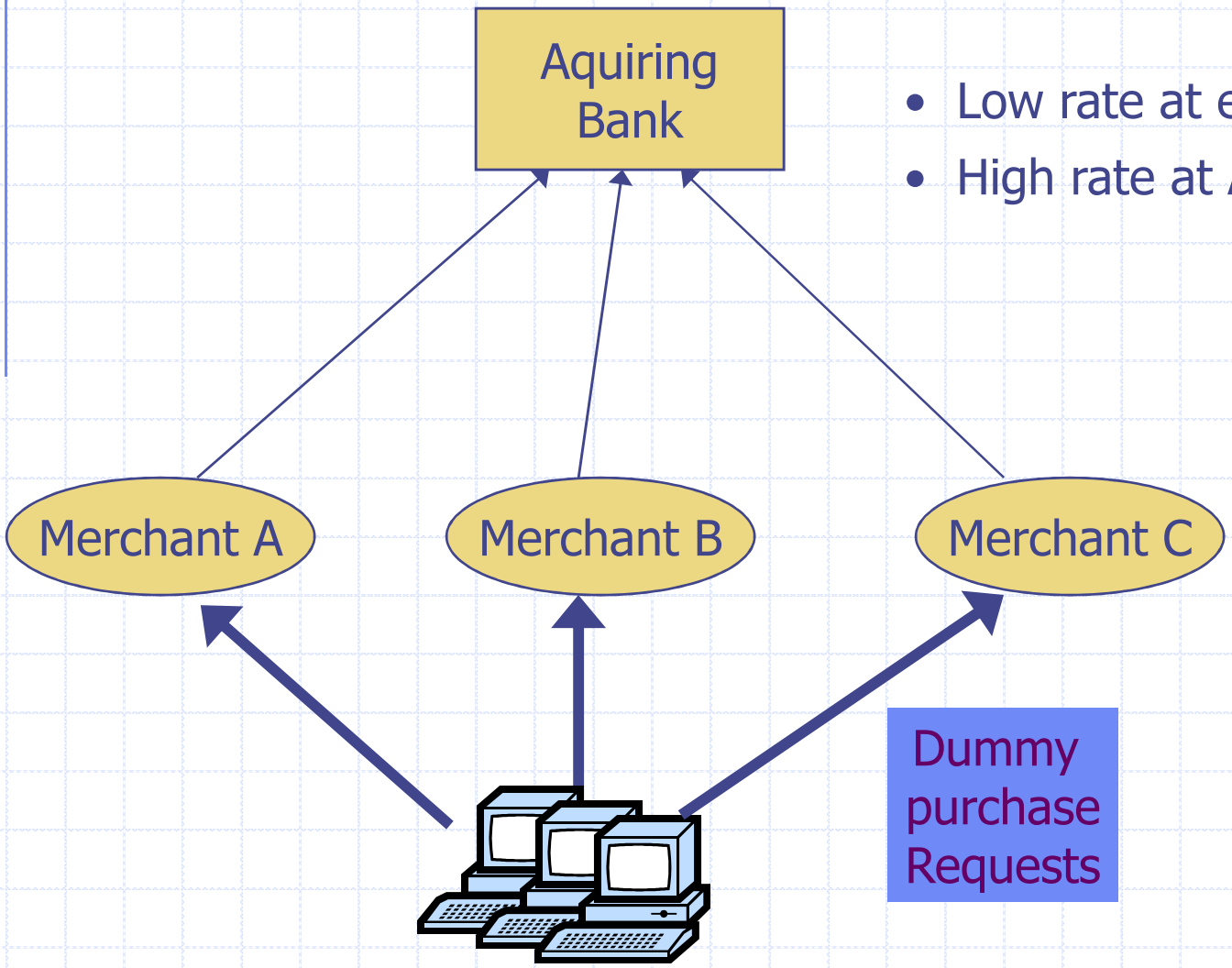
  - RSA-encrypt speed ≈ 10× RSA-decrypt speed

  ⇒ 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.

Payment DDoS

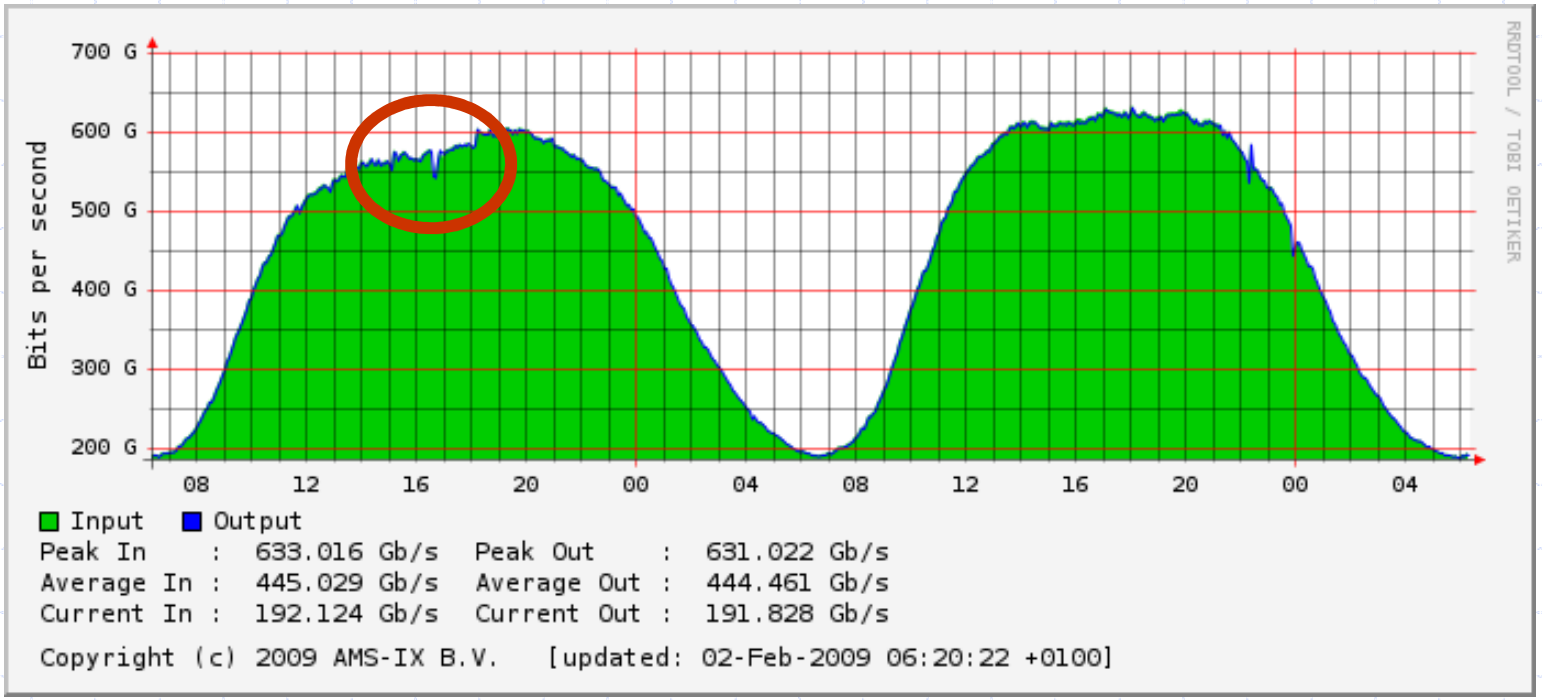


- Low rate at each Merchant
- High rate at Aquiring bank

Google DoS

- Firefox phishing/malware protection:
  - Browser downloads blacklisted list from Google
    http://safebrowsing.clients.google.com/safebrowsing/gethash
  - List contains hashes of (prefixes) of badware sites
  - Firefox consults list before following a URL
- Jan. 31, 2009: Google adds "/" to blacklist
  - For 55 minutes, all web sites marked as malware
  - Reason: human error
- Browser bug: Firefox no longer checks for "/" on list

Google DoS: results



Amsterdam peering point

DoS Mitigation

1. Client puzzles

- Idea: slow down attacker

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

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

  - 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, PDAs 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.

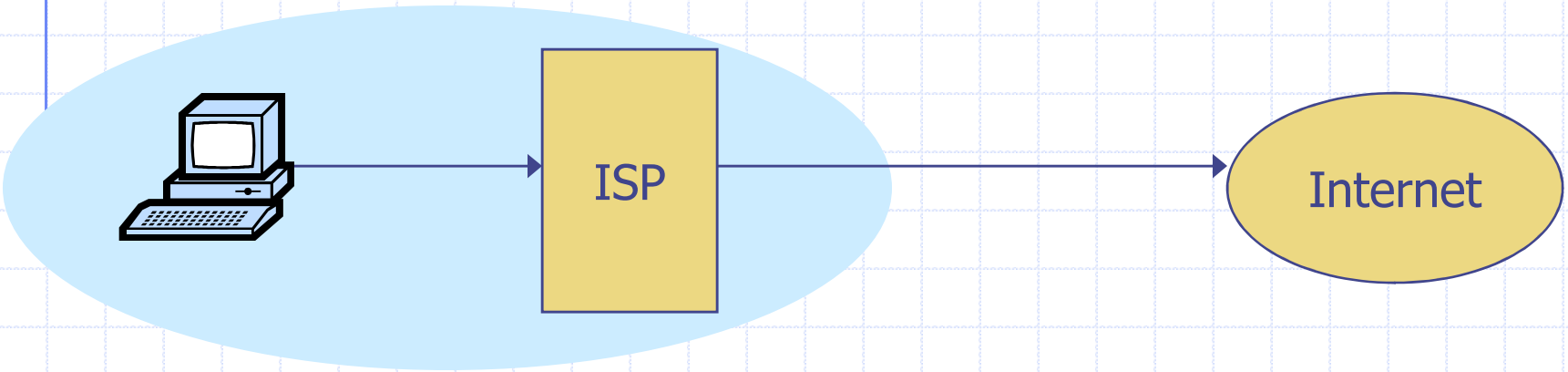
3. Source identification

Goal: identify packet source

Ultimate goal: block attack at the source

1. Ingress filtering (RFC 2827, 2000)

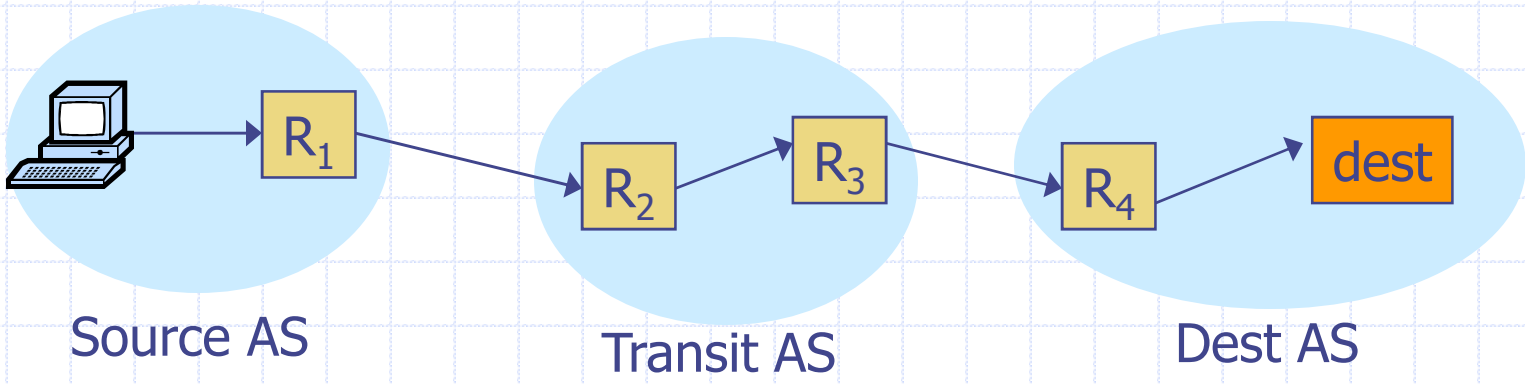
- Big problem: DDoS with spoofed source IPs
- Question: how to find packet origin?



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

Implementation problems

- ALL ISPs must do this. Requires global trust.
  - If 10% of ISPs do not implement ⇒ no defense
- Another non-solution: enforce source IP at peer AS



- Can transit AS validate packet source IP? No …

2. Traceback [Savage et al. '00]

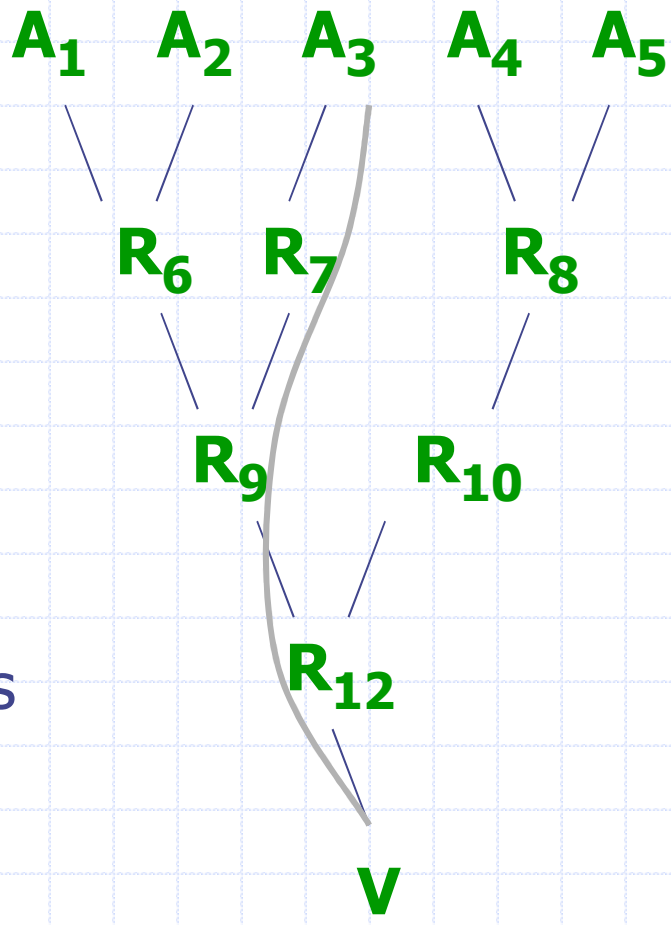
- Goal:
  - Given set of attack packets
  - Determine path to source
- How: change routers to record info in packets
- Assumptions:
  - Most routers remain uncompromised
  - Attacker sends many packets
  - Route from attacker to victim remains relatively stable

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

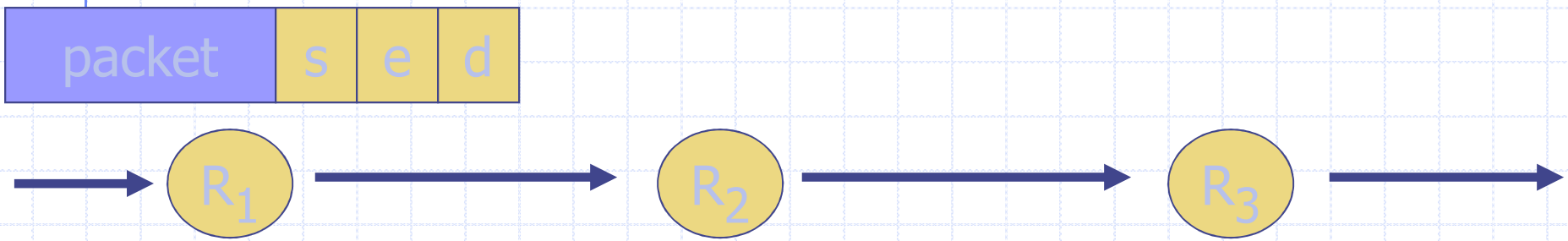
- Data fields written to packet:
  - Edge: *start* and *end* IP addresses
  - Distance: number of hops since edge stored

- 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
```

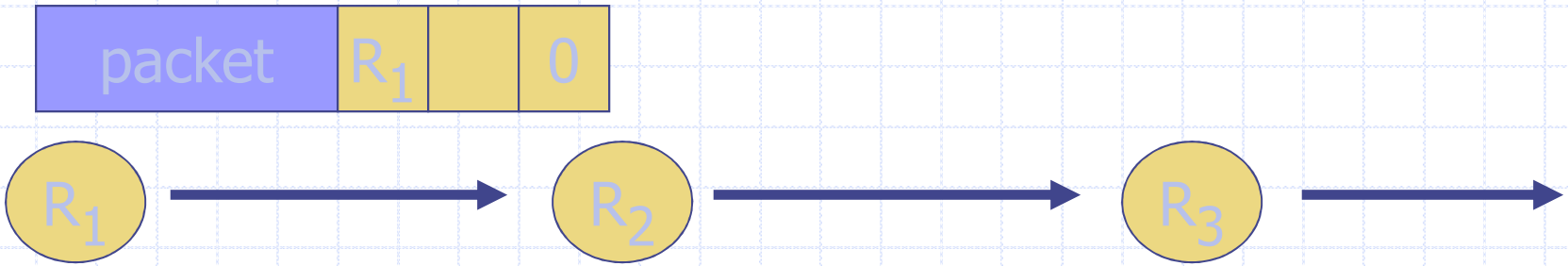
Edge Sampling: picture

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



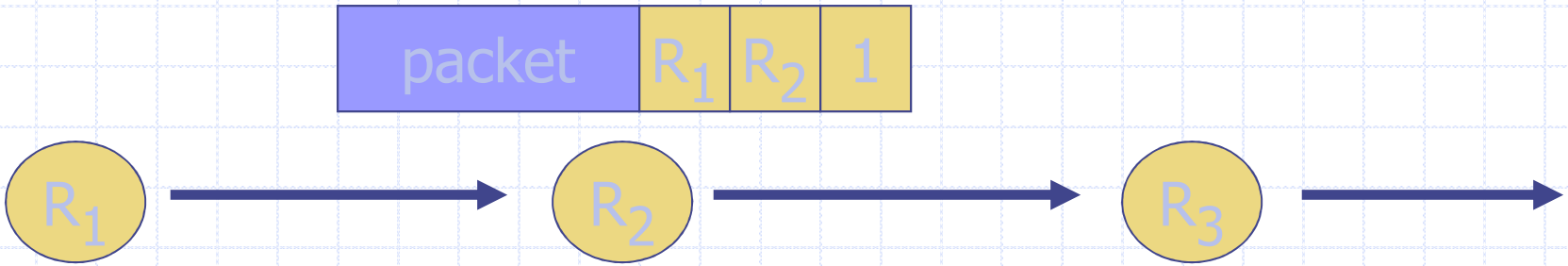
Edge Sampling: picture

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



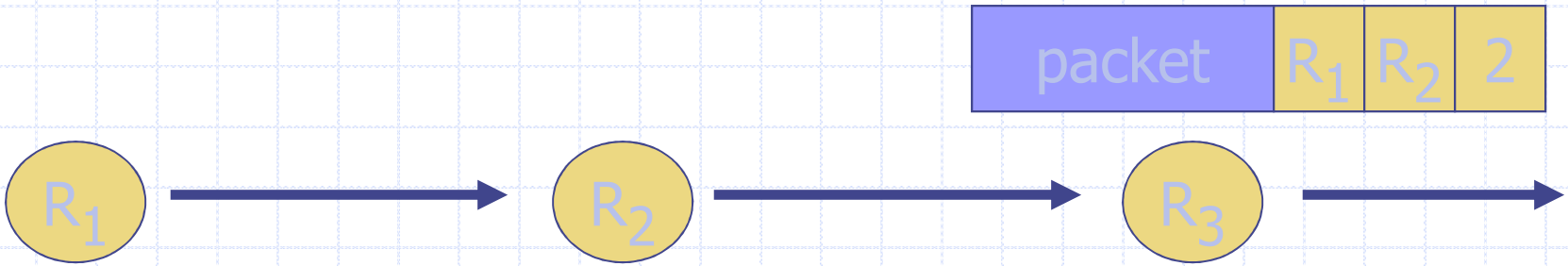
Edge Sampling

- 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

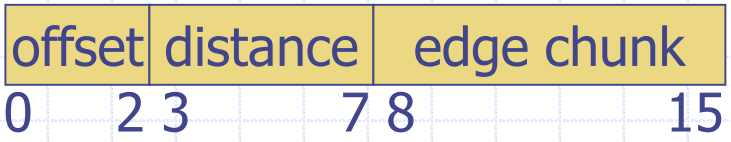
- Extract information from attack packets
- Build graph rooted at victim
  - Each (start,end,distance) tuple provides an edge
- # packets needed to reconstruct path

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

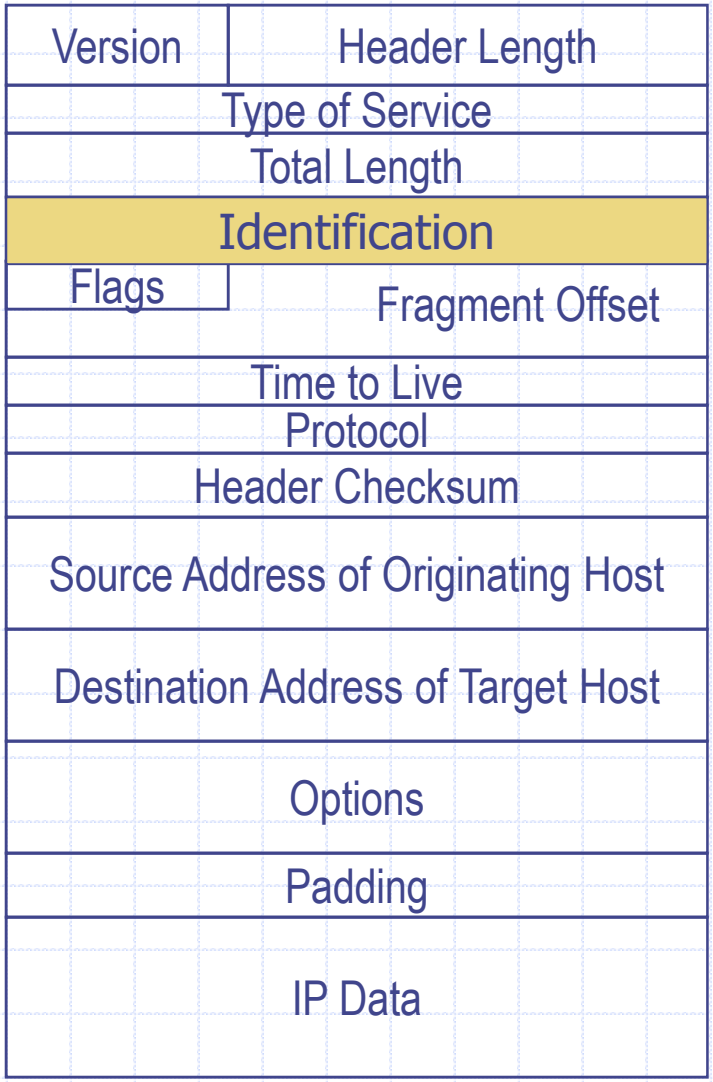
where p is marking probability, d is length of path

Details: where to store edge

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



  - Break into chunks
  - Store start ⊕ end



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

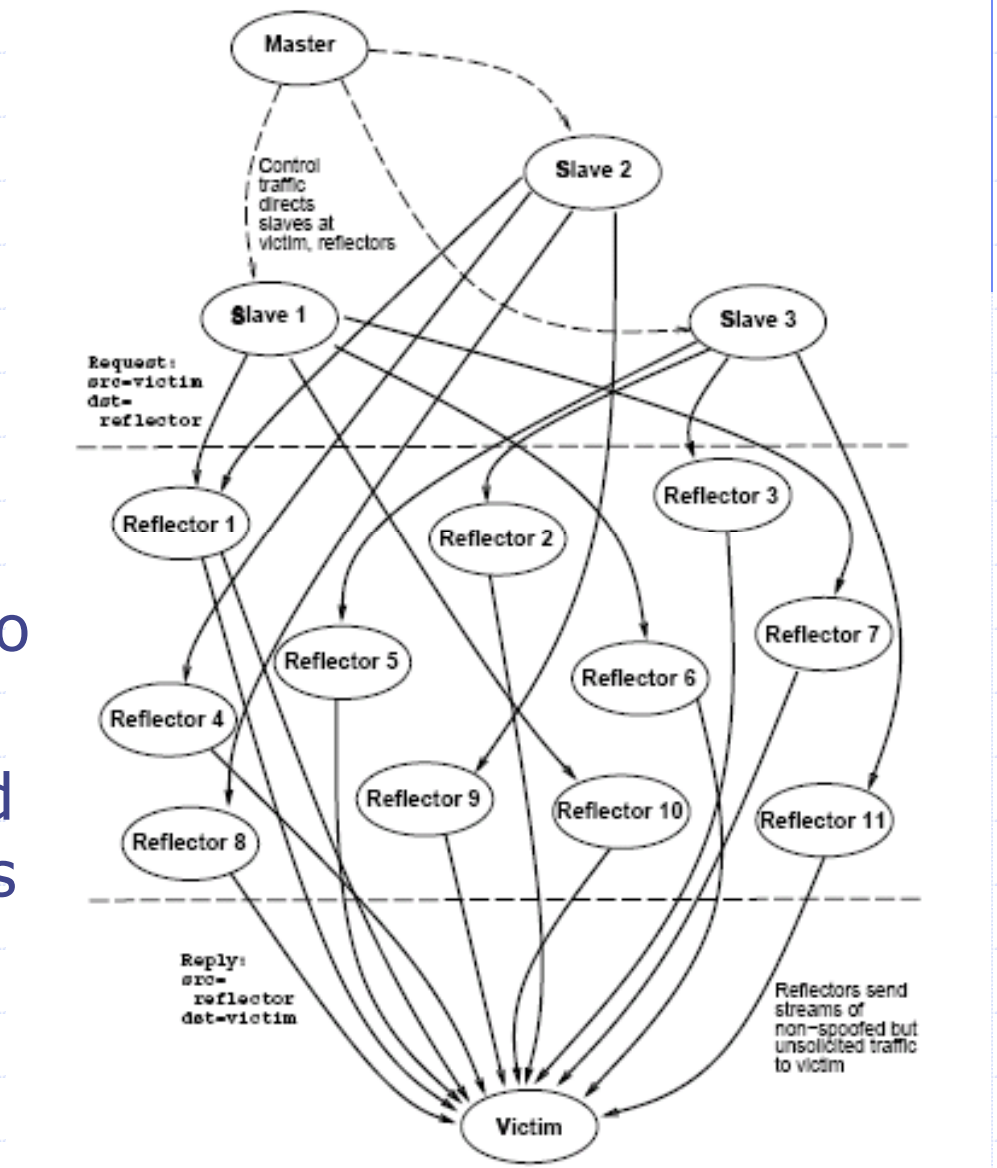
Problem: Reflector attacks [Paxson '01]

- **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

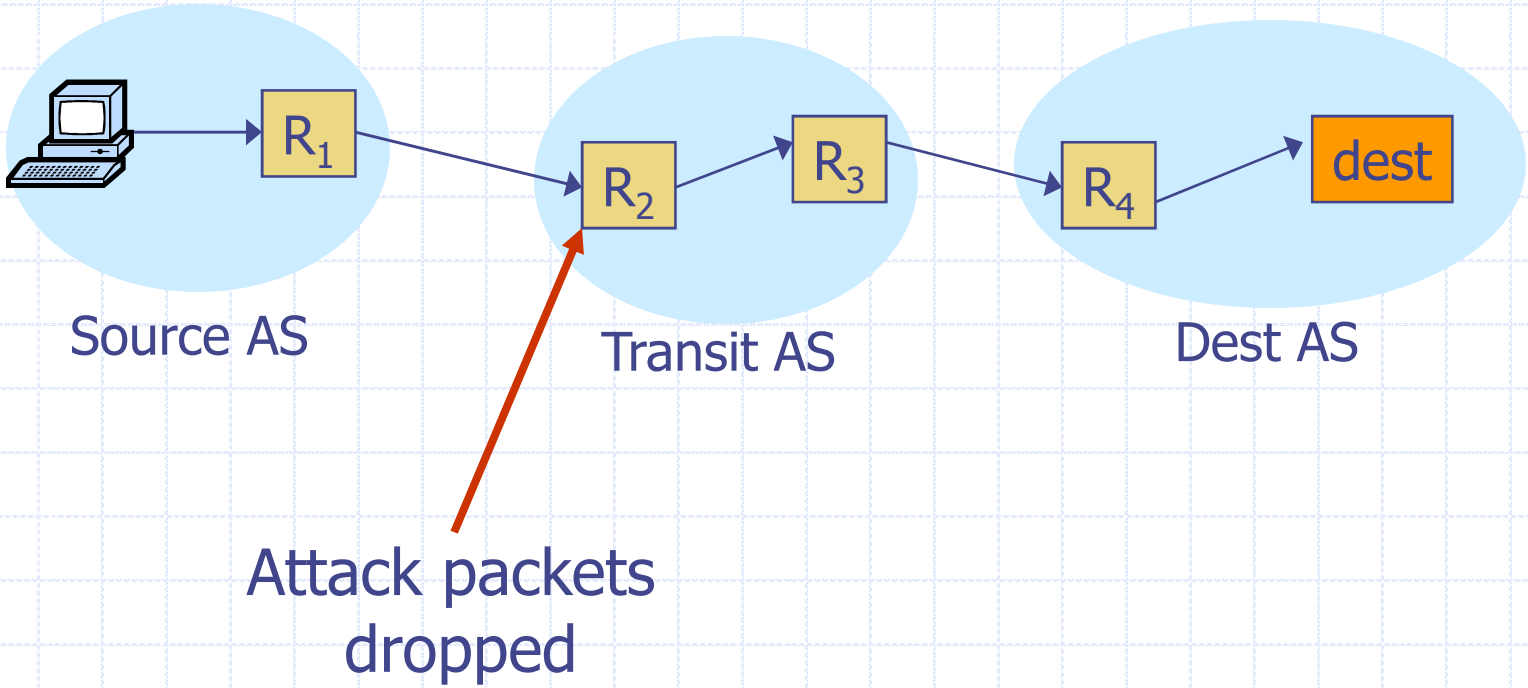
- Anderson, Roscoe, Wetherall.
  - Preventing internet denial-of-service with capabilities. SIGCOMM '04.
- Yaar, Perrig, and Song.
  - Siff: A stateless internet flow filter to mitigate DDoS flooding attacks. IEEE S&P '04.
- Yang, Wetherall, Anderson.
  - A DoS-limiting network architecture. SIGCOMM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

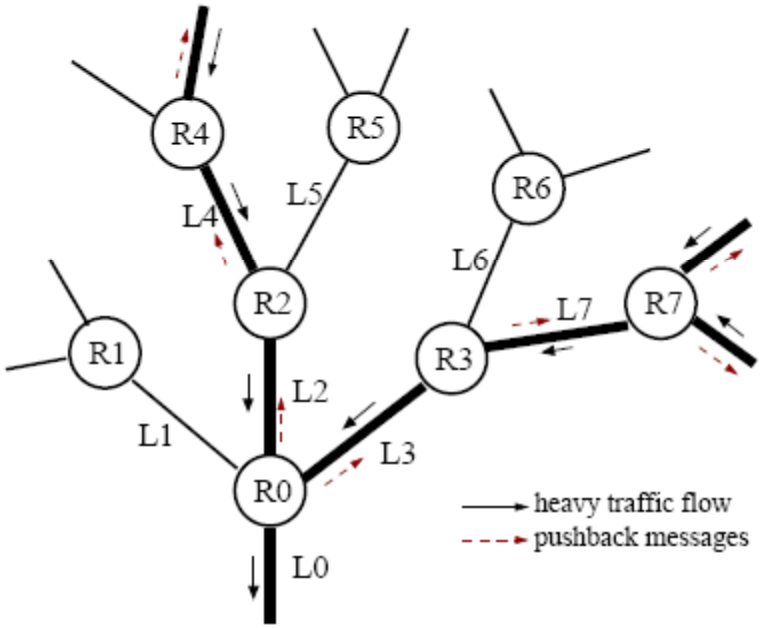
- Mahajan, Bellovin, Floyd, Ioannidis, Paxson, Shenker. Controlling High Bandwidth Aggregates in the Network. *Computer Communications Review* '02.

- 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:
  - Current Internet is ill-equipped to handle DDoS attacks
- Many good proposals for core redesign.

Spam Email

How email works: **SMTP** (RFC 821, 1982)

- Some SMTP Commands:

  MAIL FROM: <reverse-path>

  Repeated for each recipient:

  RCPT TO: <forward-path>

  RCPT TO: <forward-path>

  If unknown recipient: response "550 Failure reply"

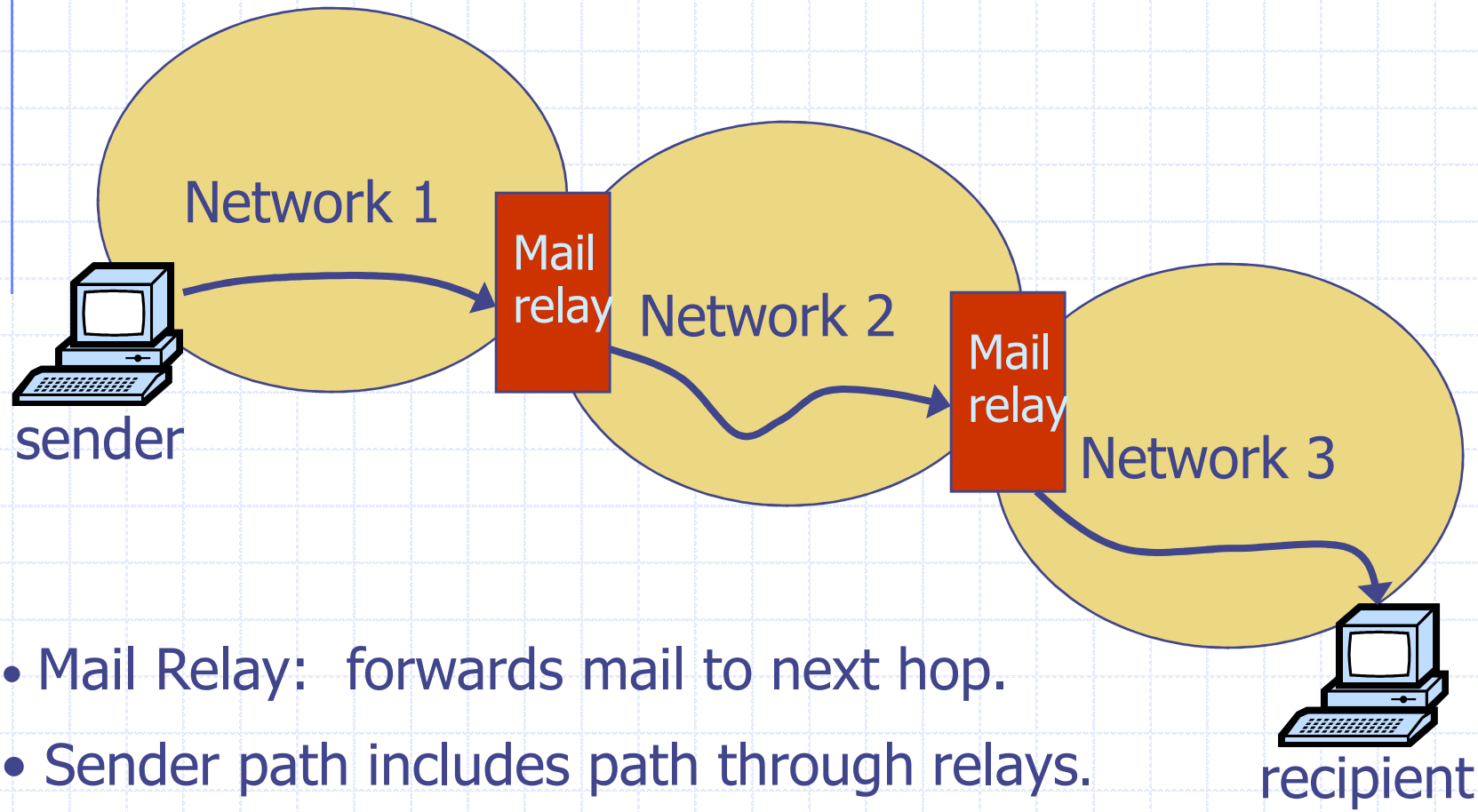
  DATA

  email headers and contents

  .

- **VRFY** username (Often disabled)
  - 250 (user exists) or 550 (no such user)

Email in the early 1980's



- Mail Relay: forwards mail to next hop.
- Sender path includes path through relays.

Spoofed email

- SMTP: designed for a trusting world …
- Data in **MAIL FROM** totally under control of sender
  - … an old example of improper input validation
- Recipient's mail server:
  - Only sees IP address of direct peer
  - Recorded in the first **From** header

The received header

- Sending spoofed mail to myself:

From someone@somewhere.com (172.24.64.20) ...

From relays:

Received: from cs-smtp-1.stanford.edu
Received: from smtp3.stanford.edu
Received: from cipher.Stanford.EDU

- Received header inserted by relays --- untrustworthy
- From header inserted by recipient mail server

Spam Blacklists

- RBL: Realtime Blackhole Lists
  - Includes servers or ISPs that generate lots of spam
  - **spamhaus.org , spamcop.net**
- Effectiveness (stats from spamhaus.org):
  - RBL can stop about 15-25% of incoming spam at SMTP connection time,
  - Over 90% of spam with message body URI checks
- Spammer goal:
  - Evade blacklists by hiding its source IP address.

Spamming techniques

Open relays

- SMTP Relay forwards mail to destination
  1. Bulk email tool connects via SMTP (port 25)
  2. Sends list of recipients (via RCPT TO command)
  3. Sends email body --- once for all recipients
  4. Relay delivers message

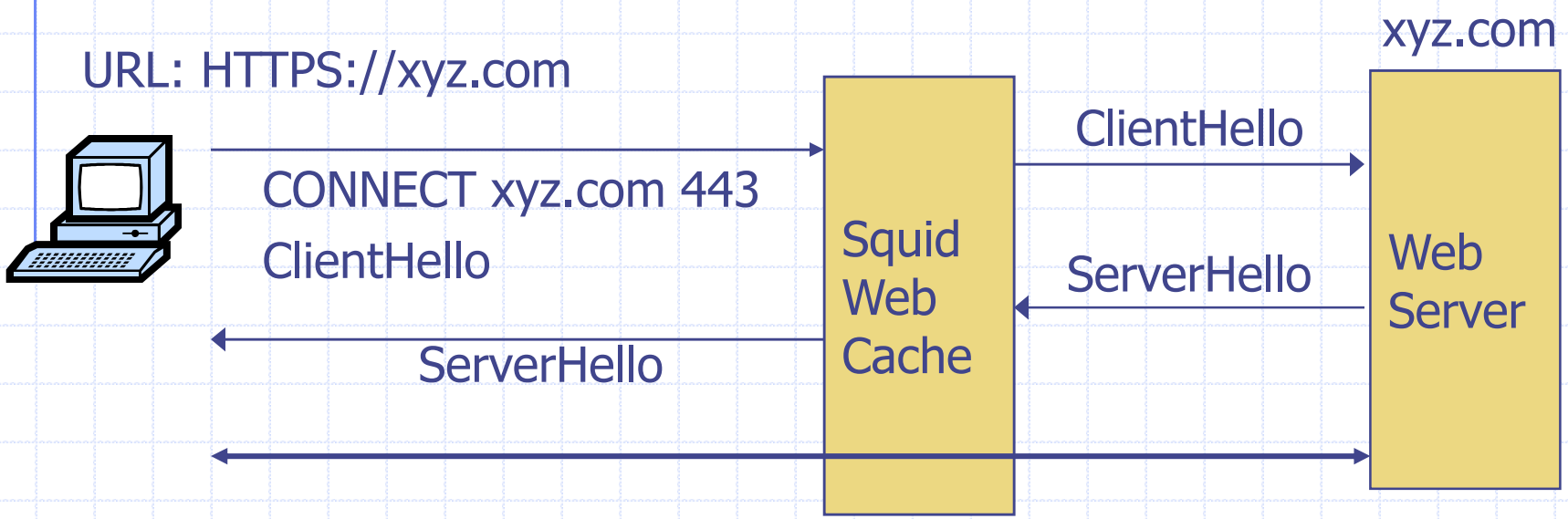
- Honest relay:
  - Adds Received header revealing source IP
  - Hacked relay does not

Example: bobax worm

- Infects machines with high bandwidth
  - Exploits MS LSASS.exe buffer overflow vulnerability
- Slow spreading:
  - Spreads on manual command from operator
  - Then randomly scans for vulnerable machines
- On infected machine: (spam zombie)
  - Installs hacked open mail relay. Used for spam.
  - Once spam zombie added to RBL:
    - Worm spreads to other machines

Open HTTP proxies

- Web cache (HTTP/HTTPS proxy) -- e.g. squid



- To spam: CONNECT SpamRecipient-IP 25
  SMTP Commands

Squid becomes a mail proxy ...

Finding proxies

- Squid manual: (squid.conf)

  acl Safe_ports port 80 443

  http_access deny !Safe_ports

  - URLs for other ports will be denied

- Similar problem with SOCKS proxies

- Some open proxy and open relay listing services:
  - http://www.multiproxy.org/
    http://www.stayinvisible.com/
    http://www.blackcode.com/proxy/
    http://www.openproxies.com/ (20$/month)

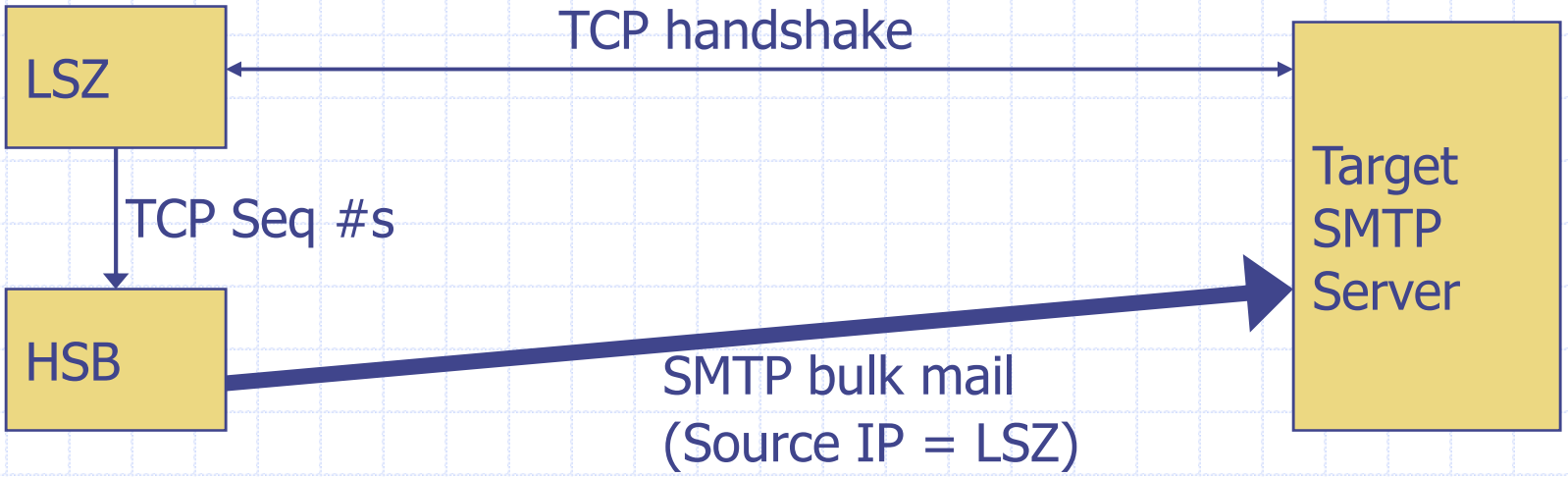
Open Relays vs. Open Proxies

- Relay vs. proxy:
  - Relay takes list of address and send msg to all
  - Proxy: spammer must send msg body to each recipient through proxy.

  $\Rightarrow$ zombies typically provide hacked mail relays.

Thin pipe / Thick pipe method

- Spam source has
  - High Speed Broadband connection (HSB)
  - Controls a Low Speed Zombie (LSZ)



  - Assumes no ingress filtering at HSB's ISP
  - Hides IP address of HSB. LSZ is blacklisted.

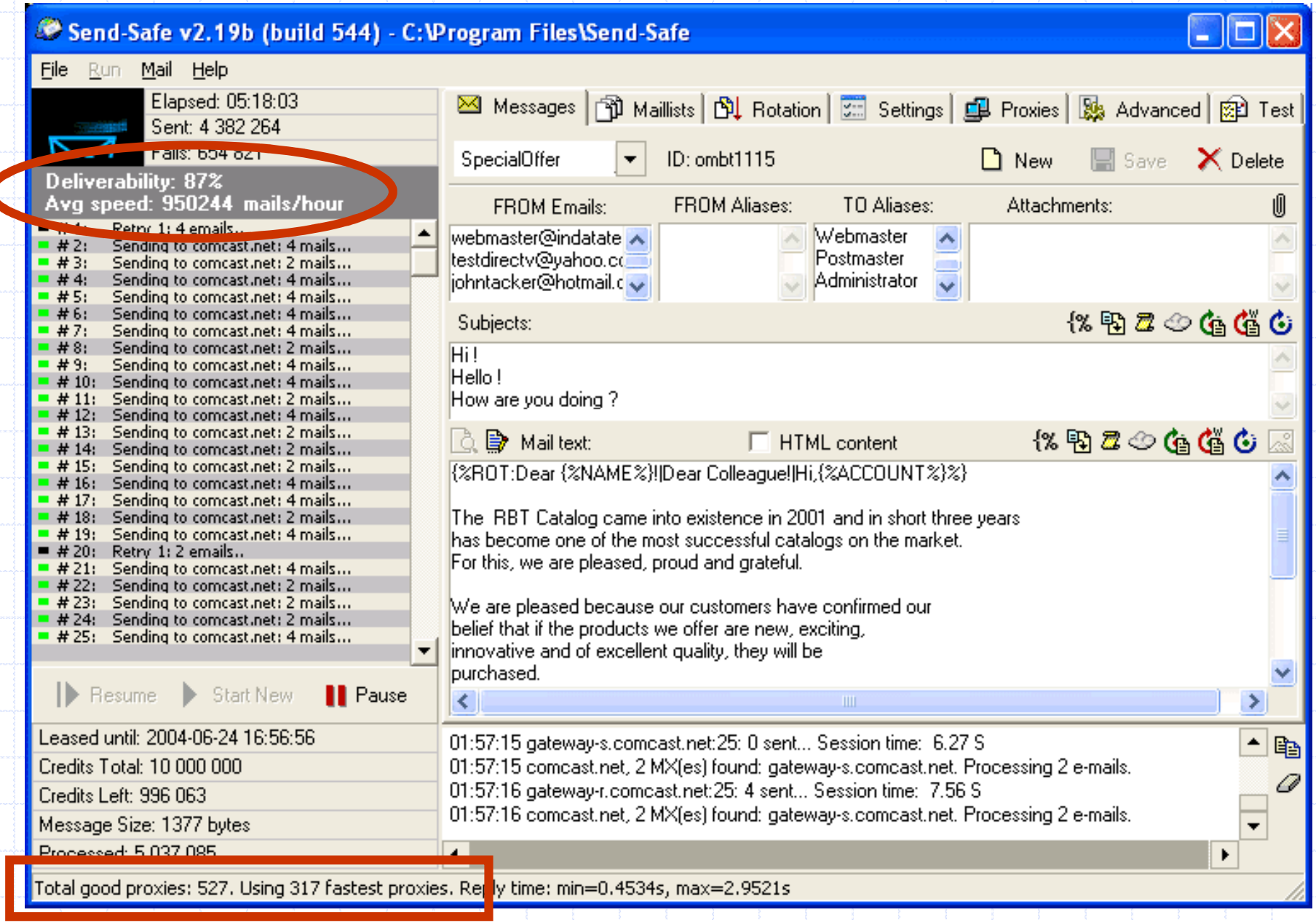
Harvesting emails

- Will not discuss here ...
- Lots of ways:
  - majordomo who command
  - SMTP VRFY command
  - Web pages
  - Dictionary harvesting
- Obvious lesson:
  - Systems should protect user info

Bulk email tools (spamware)

- Automate:
  - Message personalization
    - Also test against spam filters (e.g. spamassassin)
  - Mailing list and proxy list management

Send-Safe bulk emailer



Anti-spam methods

Will not discuss filtering methods ...

The law: CAN-SPAM act (Jan. 2004)

- Bans false or misleading header information
  - To: and From: headers must be accurate
- Prohibits deceptive subject lines
- Requires an opt-out method
- Requires that email be identified as advertisement
  - ... and include sender's physical postal address
- Also prohibits various forms of email harvesting and the use of proxies

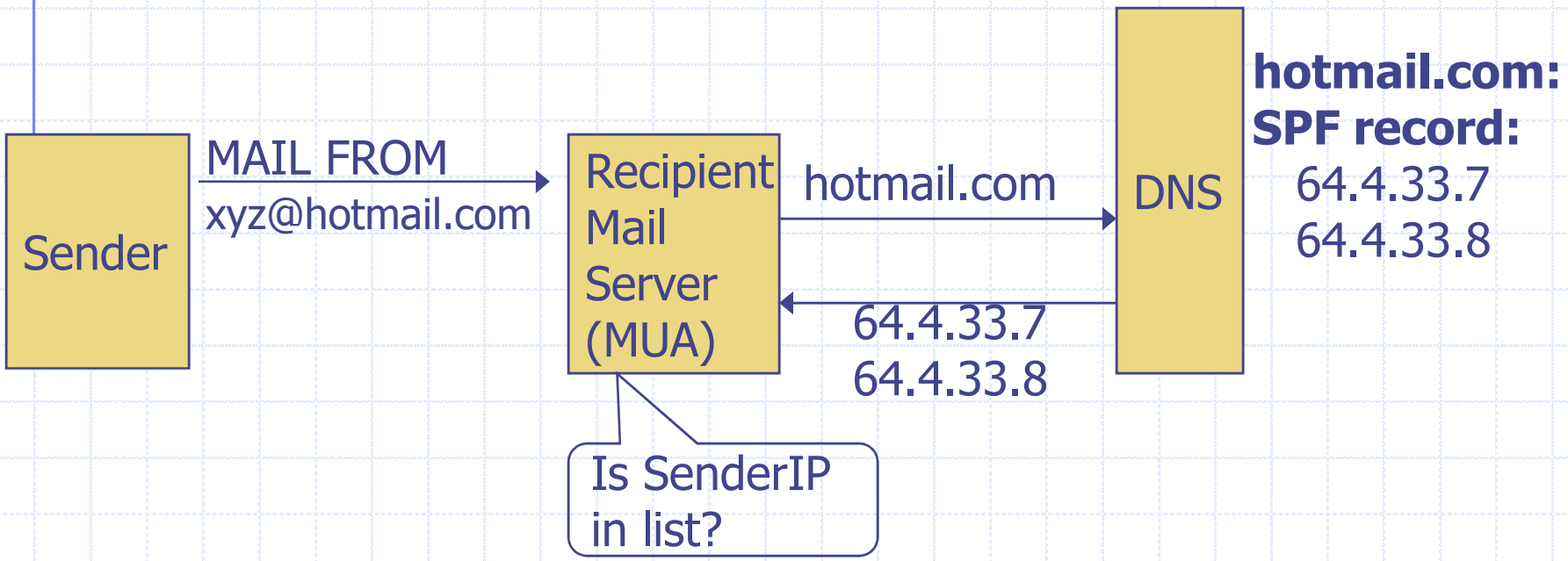
Effectiveness of CAN-SPAM

- Enforced by the FTC:
  - FTC spam archive **spam@uce.gov**
  - Penalties: 11K per act
- Dec ’05 FTC report on effectiveness of CAN-SPAM:
  - 50 cases in the US pursued by the FTC
  - No impact on spam originating outside the US
  - Open relays hosted on bot-nets make it difficult to collect evidence

http://www.ftc.gov/spam/

Sender verification I: SPF

- Goal: prevent spoof email claiming to be from HotMail
  - Why? Bounce messages flood HotMail system



More precisely: **hotmail.com TXT v=spf1 a:mailers.hotmail.com -all**

Sender verification II: DKIM

- Domain Keys Identified Mail (DKIM)
  - Same goal as SPF. Harder to spoof.
- Basic idea:
  - Sender's MTA signs email
    - Including body and selected header fields
  - Receiver's MUA checks sig
    - Rejects email if invalid
  - Sender's public key managed by DNS
    - Subdomain: **_domainkey.hotmail.com**

DKIM header example

DKIM-Signature: a=rsa-sha1; q=dns;
d=hotmail.com (domain)
s=may2006; c=relaxed/simple; (selector)
t=1117574938; x=1118006938; (time/exp)
h=from:to:subject:date; (header)
b=dzdVyOfAKCdLXdJOc9G2q8LoXSlEniSb (sig)
av+yuU4zGeeruD00lszZVoG4ZHRNiYzR

- Recipient's MUA will query for DNS TXT record of may2006._domainkey.hotmail.com

Graylists

- Recipient's mail server records triples:
  - (sender email, recipient email, peer IP)
  - Mail server maintains DB of triples
- First time: triple not in DB:
  - Mail server sends **421 reply: "I am busy"**
  - Records triple in DB
- Second time (after 5 minutes): allow email to pass
- Triples kept for 3 days (configurable)
- Easy to defeat but currently works well.

Whitelisting: DOEmail

- User specifies list of allowable senders
  - All other senders must solve CAPTCHA to enable email delivery
  - Simple UI to add incoming senders to whitelist

THE END