Denial of Service Attacks and IP Traceback

Ben Zhao
Oct 29, 2018
CS 232/332

THE UNIVERSITY OF CHICAGO

Today

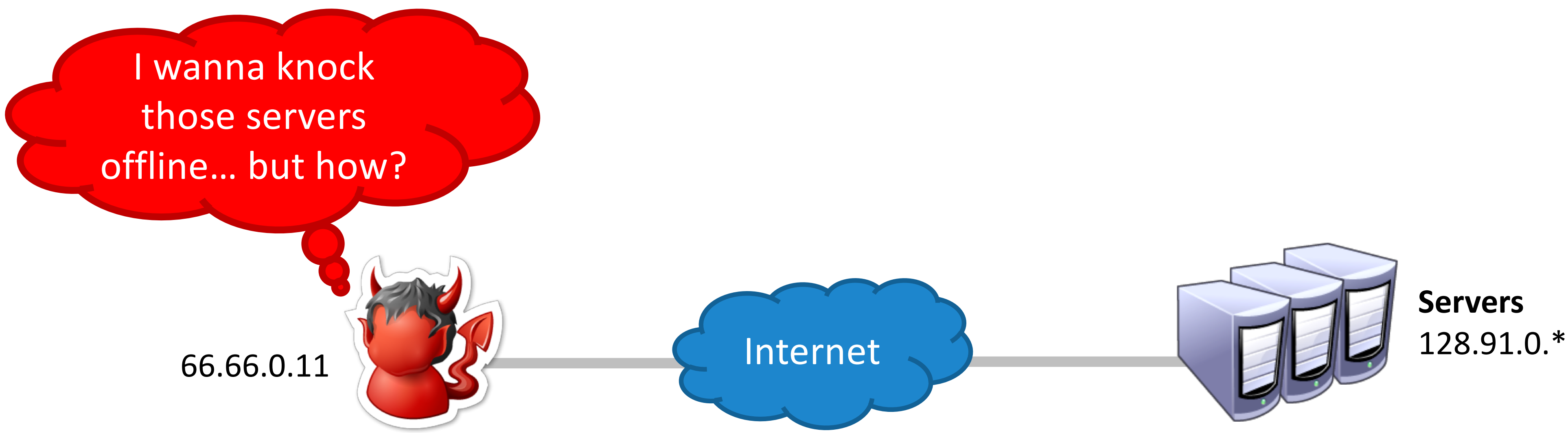
- Denial of Service Attacks (DoS)
- Defenses
  - Traceback (assignment 3)
  - CDNs

Denial of Service (DoS)

- Prevent users from being able to access a specific computer, service, or piece of data
- In essence, an attack on availability
- Possible vectors:
  - Exploit bugs that lead to crashes
  - Exhaust the resources of a target
- Often very easy to perform…
- … and fiendishly difficult to mitigate

DoS Attacker Goals & Threat Model

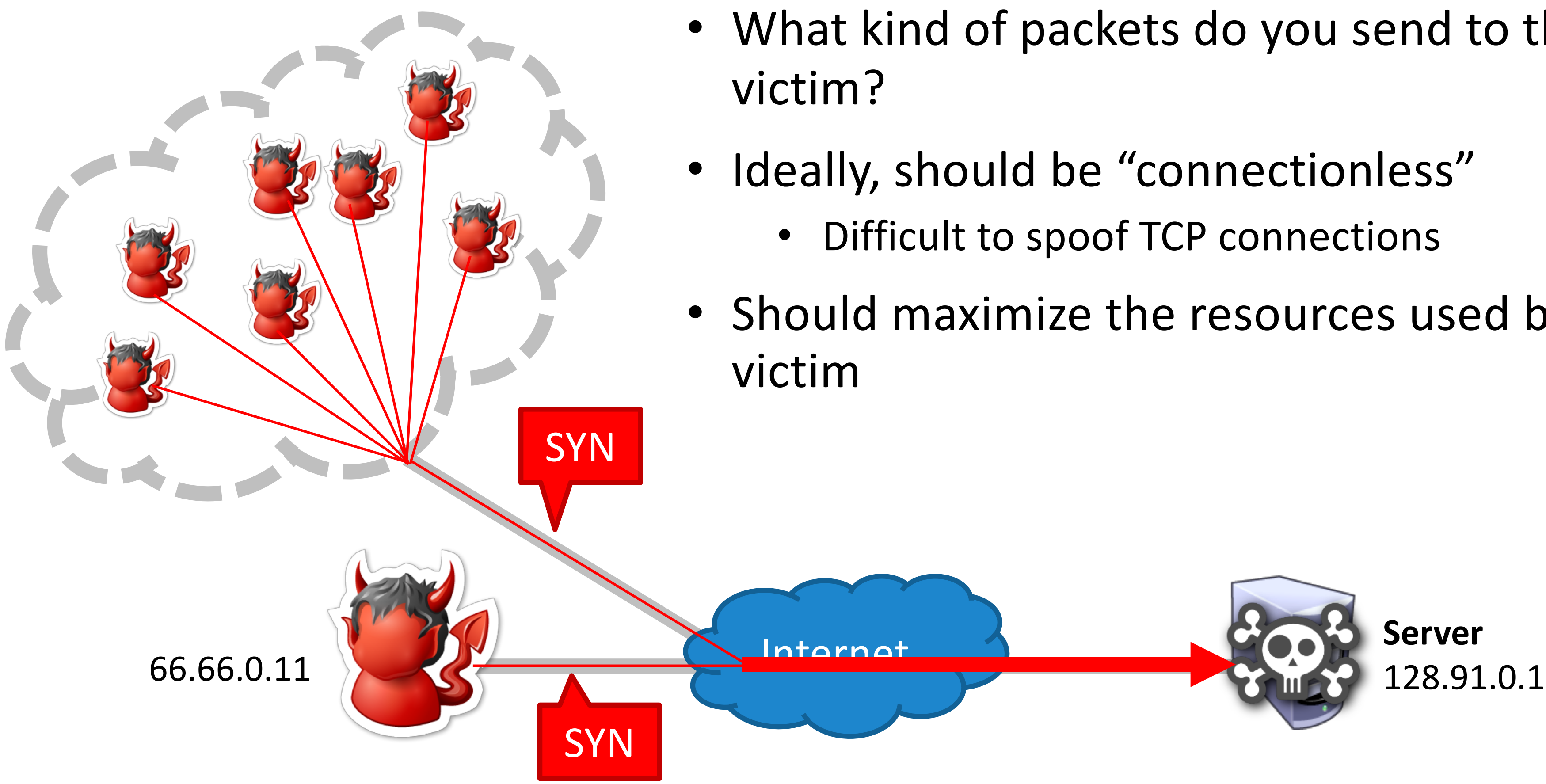
- Active attacker who may send arbitrary packets
- Goal is to reduce the availability of the victim



DoS Attack Parameters

- How much bandwidth is available to the attacker?
  - Can be increased by controlling more resources…
  - Or tricking others into participating in the attack
- What kind of packets do you send to victim?
  - Minimize effort and risk of detection for attacker…
  - While also maximizing damage to the victim

Standard DDoS, Revisited

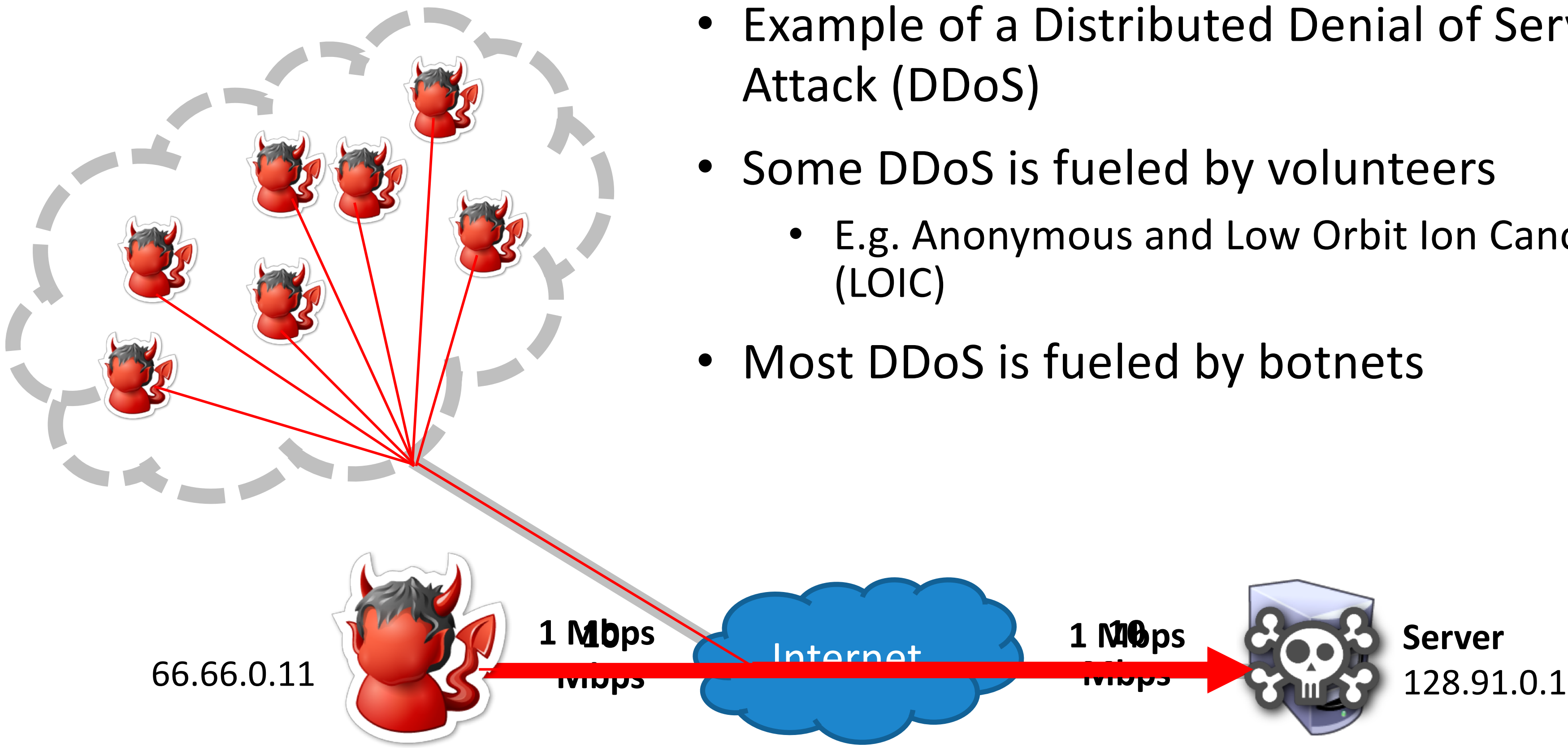


- What kind of packets do you send to the victim?
- Ideally, should be “connectionless”
  - Difficult to spoof TCP connections
- Should maximize the resources used by the victim

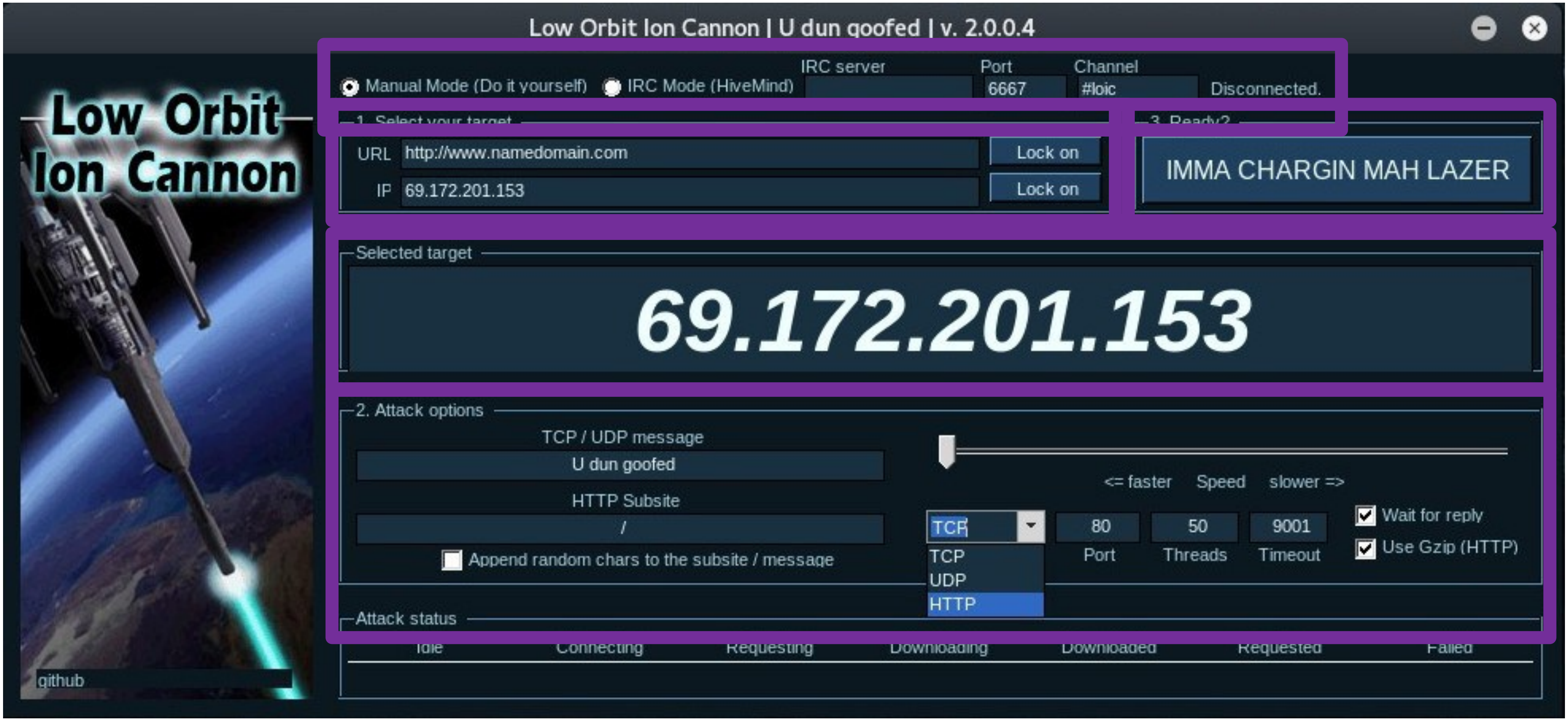
TCP SYN Flood

- TCP stack keeps track of connection state in data structures called Transmission Control Blocks (TCBs)
  - New TCB allocated by the kernel whenever a listen socket receives a SYN
  - TCB must persist for at least one RTO
- Attack: flood the victim with SYN packets
  - Exhaust available memory for TCBs, prevent legitimate clients from connecting
  - Crash the server OS by overflowing kernel memory
- Advantages for the attacker
  - No connection – each SYN can be spoofed, no need to hear responses
  - Asymmetry – attacker does not need to allocate TCBs

Exploiting Asymmetry



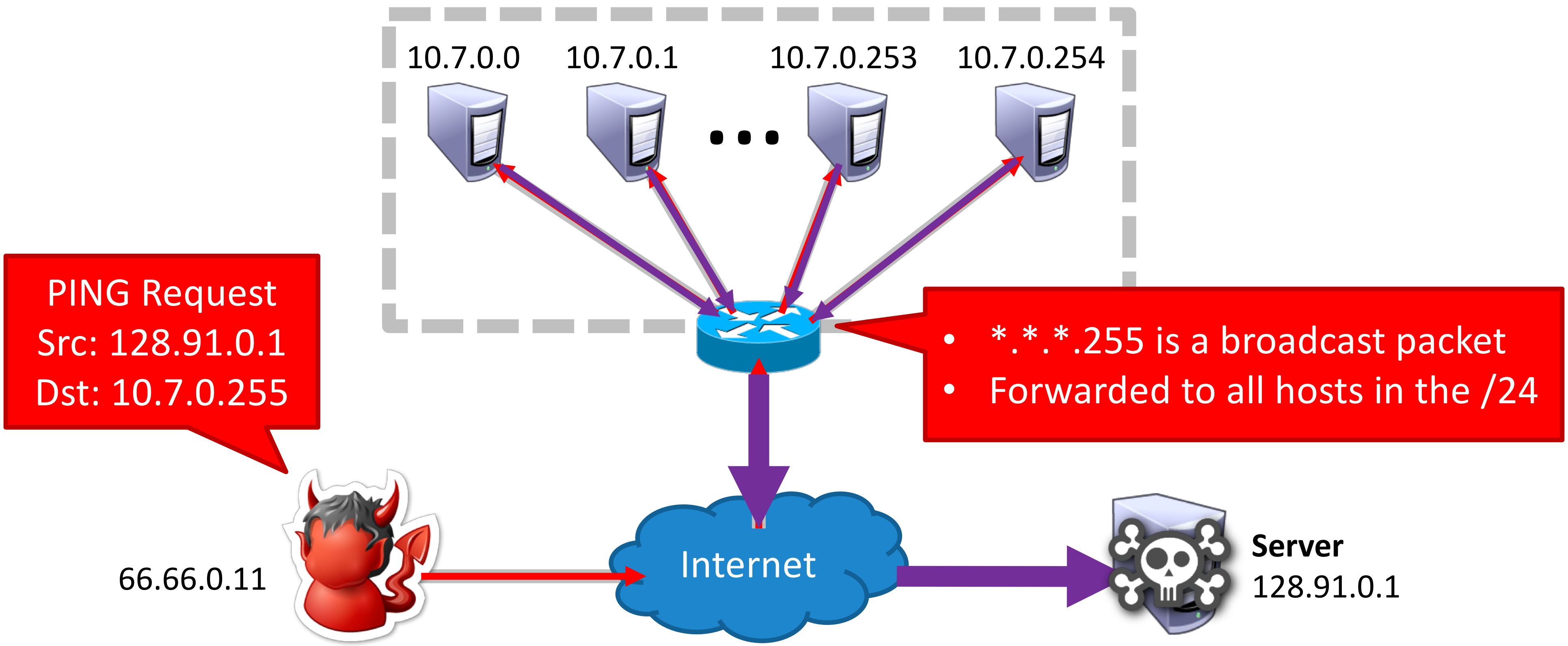
- Example of a Distributed Denial of Service Attack (DDoS)
- Some DDoS is fueled by volunteers
  - E.g. Anonymous and Low Orbit Ion Canon (LOIC)
- Most DDoS is fueled by botnets



Dumbest tool ever: doesn’t spoof your IP address

Guarantees that you will be caught by law enforcement

The Smurf Attack



Why Does Smurfing Work?

1. ICMP protocol does not include authentication
   - No connections
   - Receivers accept messages without verifying the source
   - Enables attackers to spoof the source of messages
2. Attacker benefits from an amplification factor

$$amp\ factor = \frac{total\ response\ size}{request\ size}$$

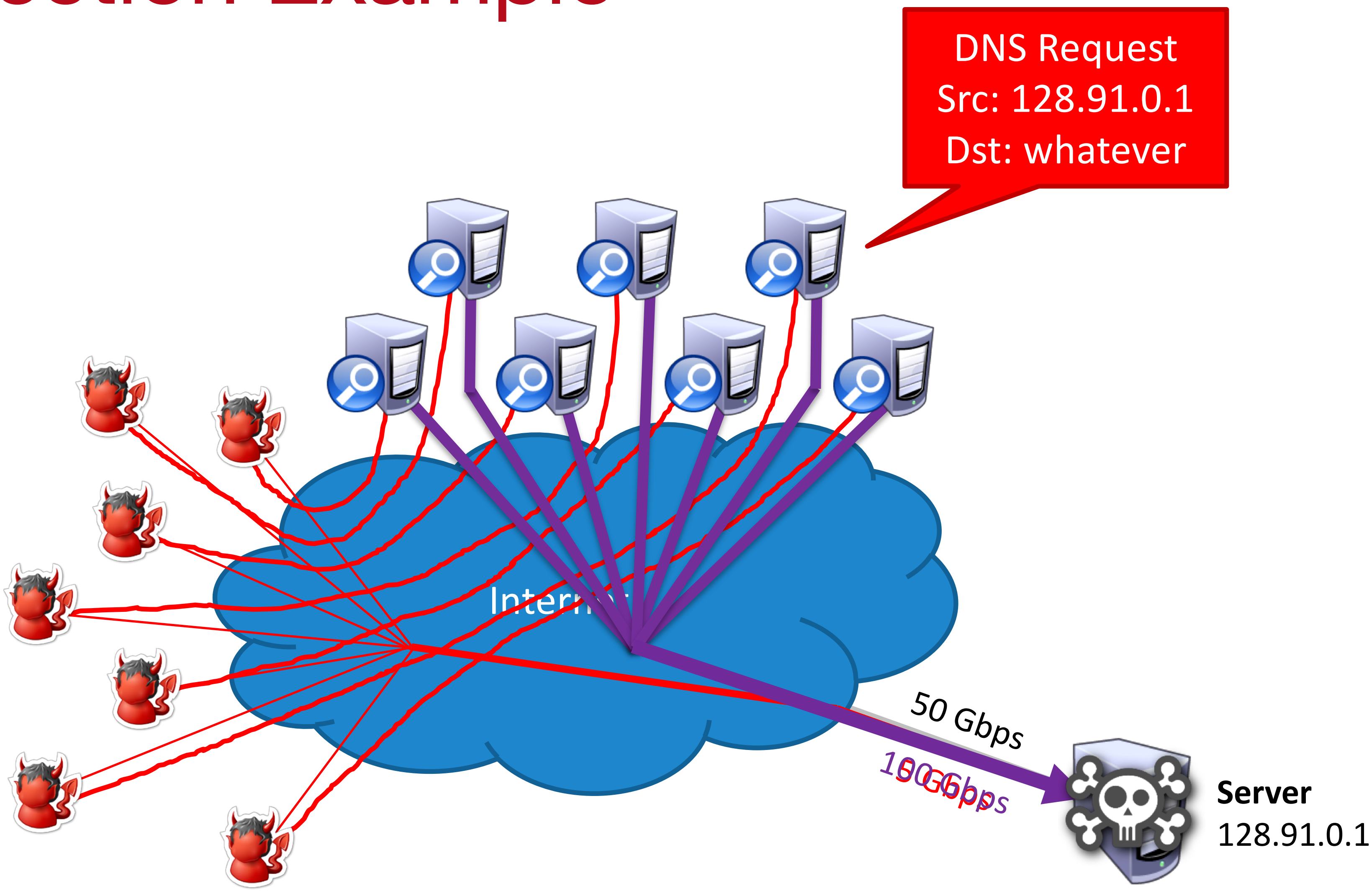
Reflection/Amplification Attacks

- Smurfing is an example of a reflection or amplification DDoS attack
- Fraggle attack also relies on broadcasts for amplification
  - Send spoofed UDP packets to IP broadcast addresses on port 7 (*echo*) and 13 (*chargen*)
    - *echo* – 1500 bytes/pkt requests, equal size responses
    - *chargen* -- 28 bytes/pkt request, 10K-100K bytes of ASCII in response
  - Amp factor
    - *echo – [number of hosts responding to the broadcast]:1*
    - *chargen – [number of hosts responding to the broadcast]*360:1*

DNS Reflection Attack

- Spoof DNS requests to many **open** DNS resolvers
  - DNS is a UDP-based protocol, no authentication of requests
  - Open resolvers accept requests from any client
    - E.g. 8.8.8.8, 8.8.4.4, 1.1.1.1, 1.0.0.1
  - February 2014 – 25 million open DNS resolvers on the internet
- 64 byte DNS queries generate large responses
  - Old-school "A" record query → maximum 512 byte response
  - EDNS0 extension "ANY" record query → 1000-6000 byte response
    - E.g. $ dig ANY isc.org
  - Amp factor – *180:1*
- Attackers have been known to register their own domains and install very large records just to enable reflection attacks!

Reflection Example



NTP Reflection Attack

- Spoof requests to open Network Time Protocol (NTP) servers
  - NTP is a UDP-based protocol, no authentication of requests
  - May 2014 – 2.2 million open NTP servers on the internet
- 234 byte queries generate large responses
  - *monlist* query: server returns a list of all recent connections
  - Other queries are possible, i.e. *version* and *showpeers*
  - Amp factor – from *10:1* to *560:1*

memcached Reflection Attack

- Spoof requests to open memcached servers
  - Popular <key:value> server used to cache web objects
  - memcached uses a UDP-based protocol, no authentication of requests
  - February 2018 – 50k open memcached servers on the internet
- 1460 byte queries generate large responses
  - A single query can request multiple 1MB <key:value> pairs from the database
  - Amp factor – up to *50000:1*

Reflection Amplification

Protocol	Amplification Factor
memcached	50000
NTP	557
chargen	359
DNS	179
QOTD	140
BitTorrent	54
SSDP	31
SNMPv2	6
Steam	6
NetBIOS	4

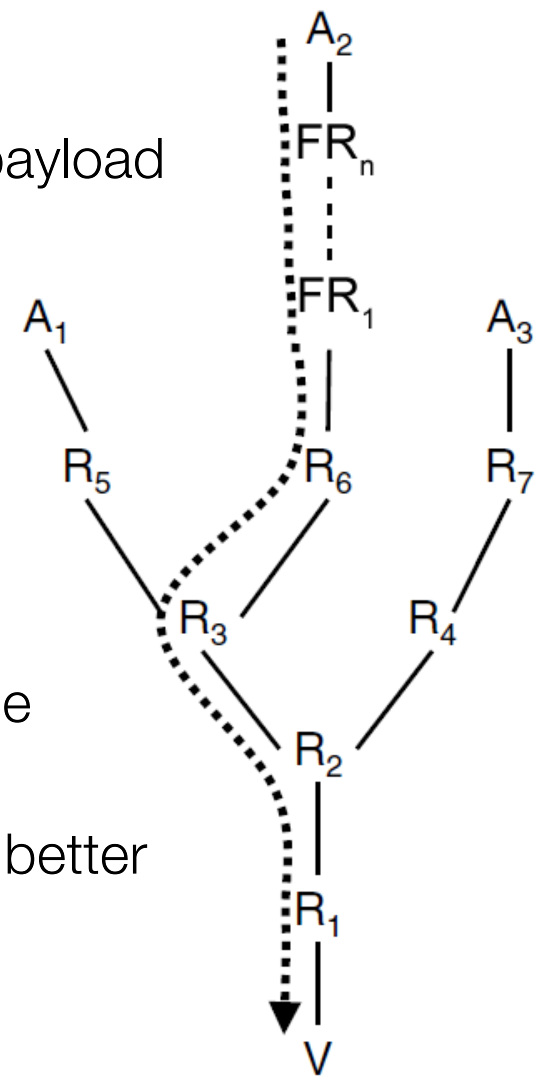
Infamous DDoS Attacks

When	Against Who	Size	How
March 2013	Spamhaus	120 Gbps	Botnet + DNS reflection
February 2014	Cloudflare	400 Gbps	Botnet + NTP reflection
September 2016	Krebs	620 Gbps	Mirai
October 2016	Dyn (major DNS provider)	1.2 Tbps	Mirai
March 2018	Github	1.35 Tbps	Botnet + memcached reflection

Mitigation: *IP Traceback*

- IP includes a Record Route option
  - If enabled, each router inserts its IP into packet payload (but off by default)

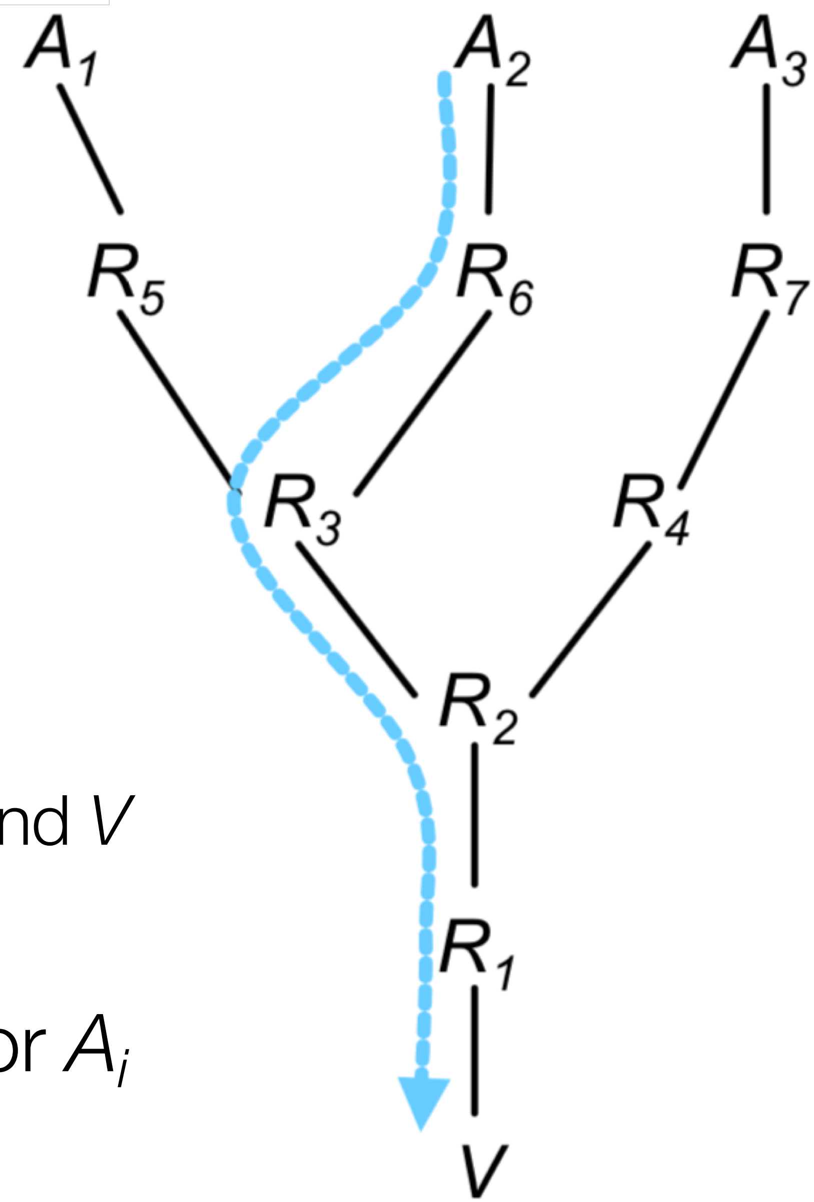
- Proposals for Packet marking
  - Practical IP traceback, Stefan Savage, 2000
  - Probabilistic marking by routers
  - Novel compression/sampling algorithms to enable victim to reconstruct entire path
  - Extended by Song/Perrig in 2001 (INFOCOM) to better handle DDoS and minimize false positives



Savage et al, SIGCOMM 2000

Practical Network Support for IP Traceback

- First *practical* proposal for
- Assumptions
  - Set of attackers A_i
  - Set of routers R_i
  - Victim V
- Attack path for A_i
  - Ordered list of routers betw A_i and V
  - e.g. $\{R_6, R_3, R_2, R_1\}$
- Goal: determine attack path for A_i



Basic Idea: Packet Marking

- Routers “mark” packets with path state
- Naïve approach
  - Routers add their addr to each packet
  - Expensive, not enough “space”
- Use edge sampling instead
  - Edge: two adjacent router addresses (start&end)
  - Distance: # edges traversed since marked
- Probabilistically mark packets in routers
  - DoS all about volume: many packets ==> path reconstruction

Mark & Reconstruct

- Marking a packet (assuming *start* & *end* & *distance* fields)

```
with probability p,
   write R into start field
   write 0 into distance field
else
   if distance == 0 then
      write R into end field
   increment distance field
```

(worry about space later)

- Path reconstruction at victim
  - Collect all attack packets
  - Each (start,end,dist) is single edge
  - Traverse edge from root to find attack path

packets needed to reconstruct path

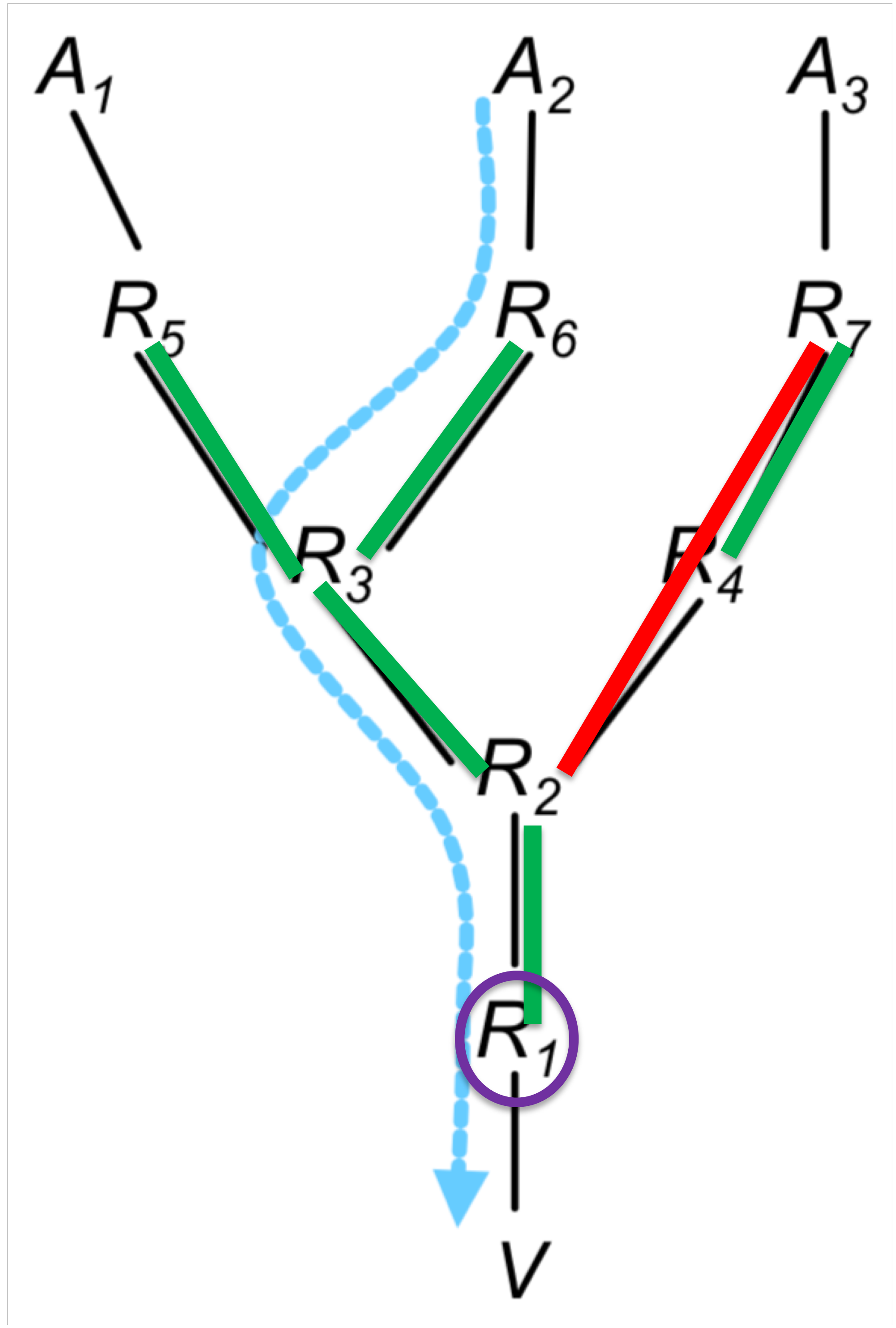
$$E(X) < \frac{\ln(d)}{p(1-p)^{d-1}}$$

p: marking probability
d: length of path

Example

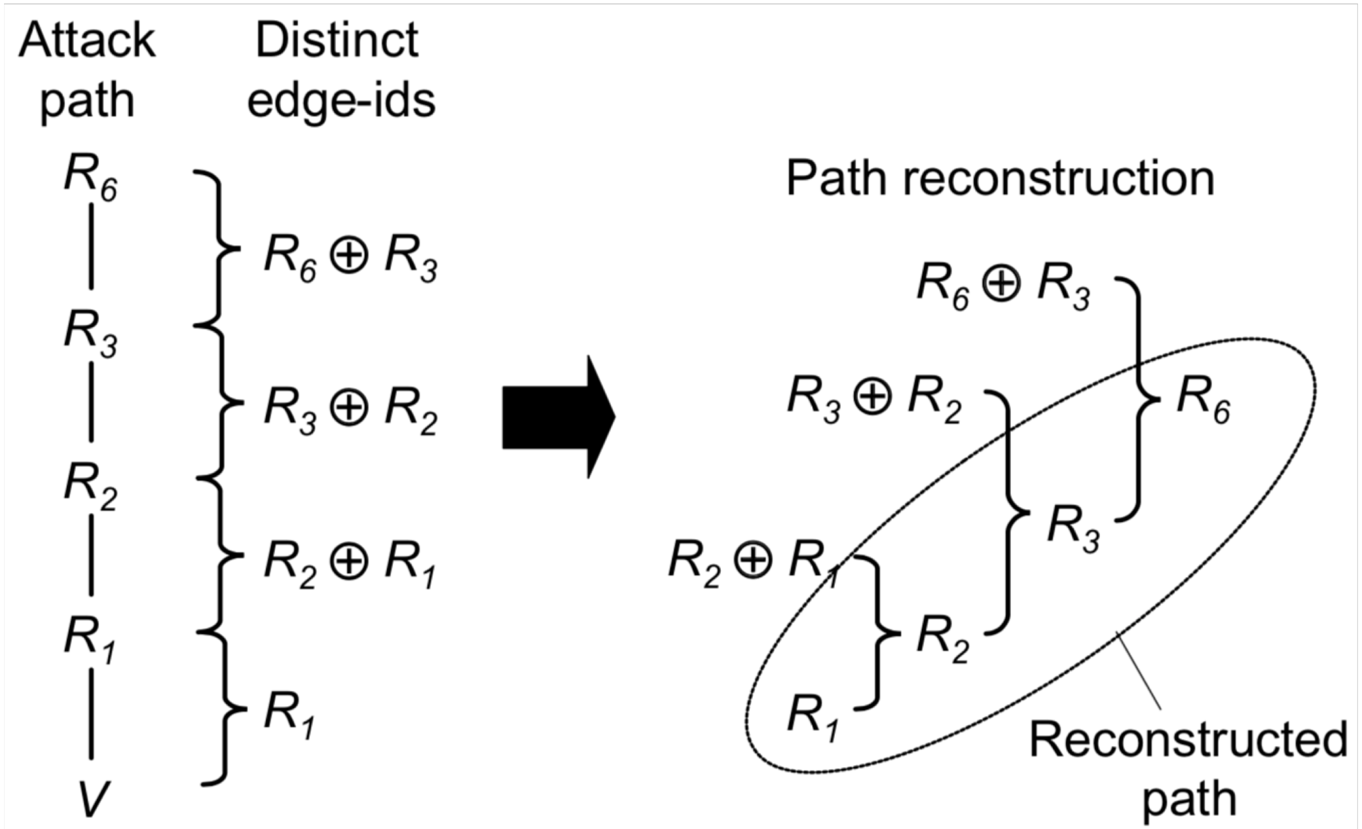
- Packets at *V* *(count backwards from R_1)*
  - <R6, R3, 3>
  - <R3, R2, 2>
  - <R5, R3, 3>
  - <R2, R1, 1>
  - <R7, R4, 3>

  - <R7, R2, 2> ??
  - <R9, R6, 4> ??



Reality Sets In…

- Don't have space for 3-tuple (32+32+8bits)
- Overload IP-identification field (16bits total)
- Compress!!!

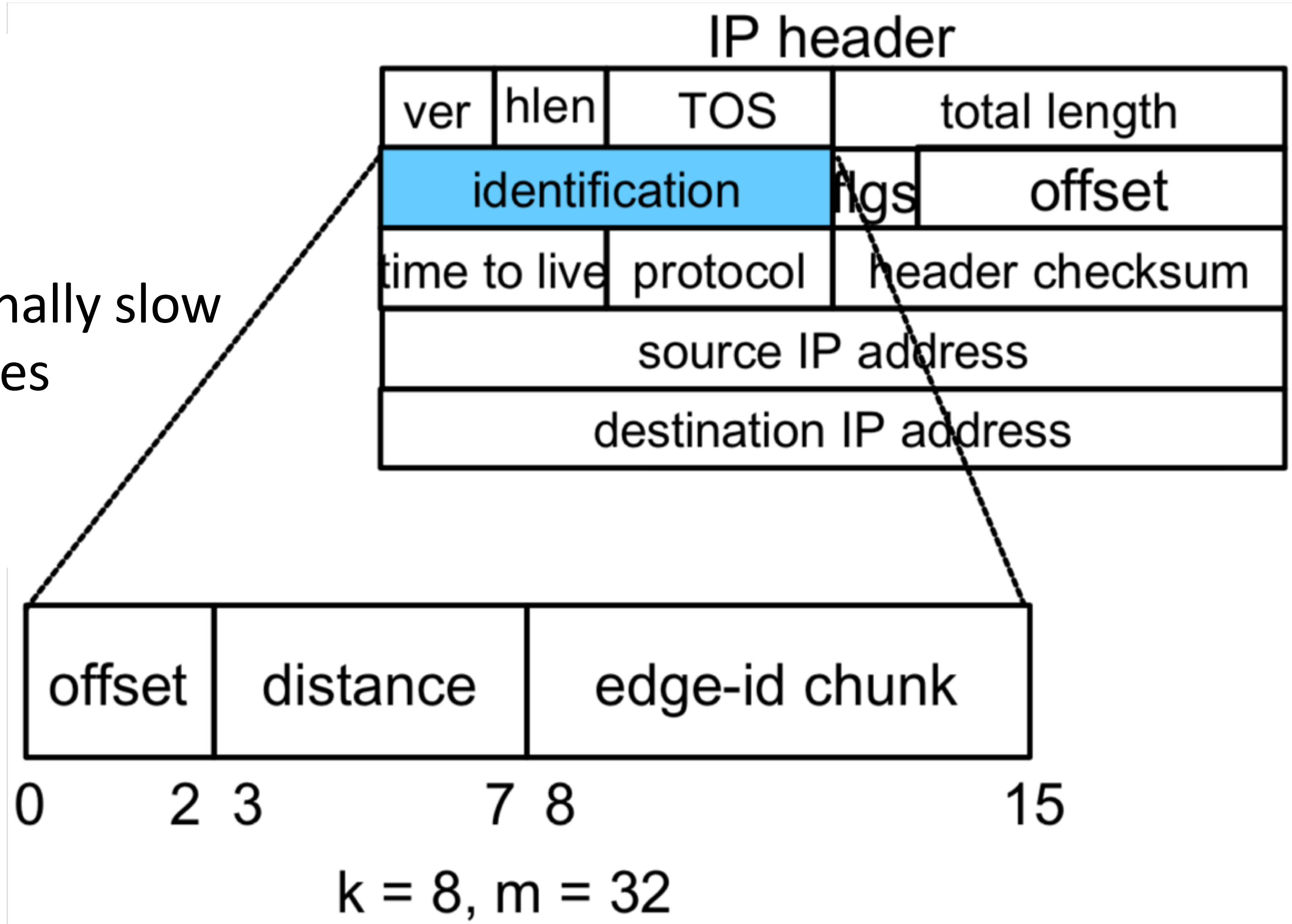


Still Not Enough Space…

- Can't store whole edge-id
- Settle for one of *k* chunks of edge-id
  - Mark random chunk & offset into packet
- Chunks may not be unique
  - Augment edge-id with hash of *m* bits
  - Validate chunk combinations at reconstruction

Result

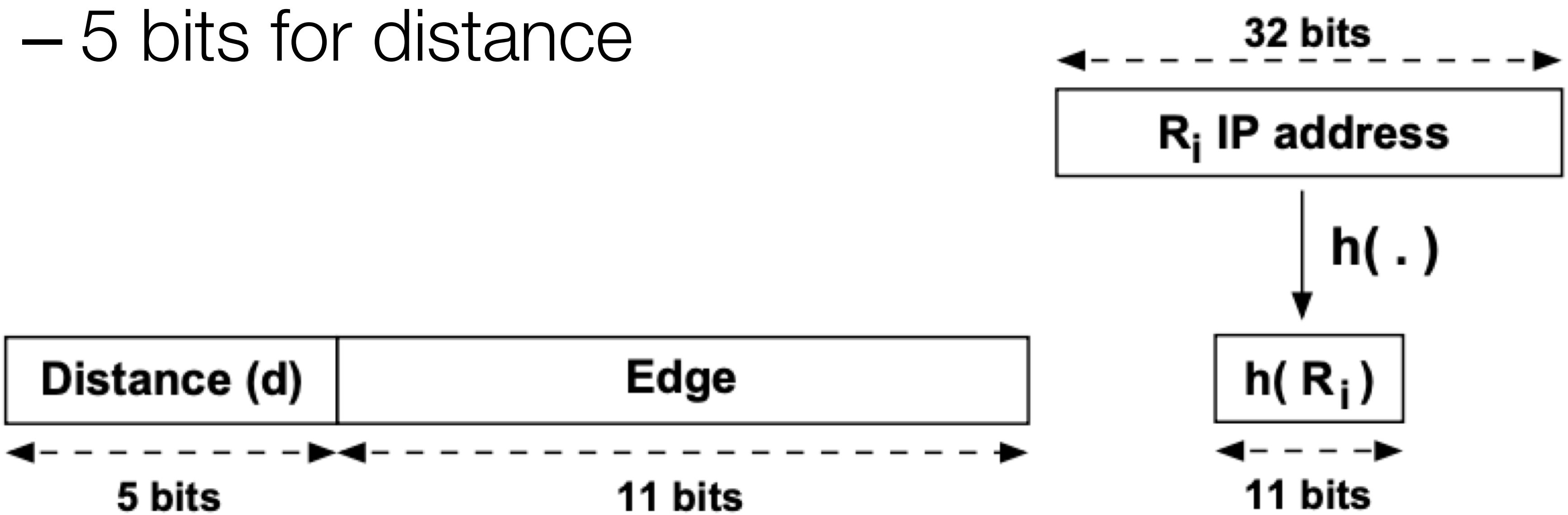
Not ideal
A Bit messy
Computationally slow
False positives



Song & Perrig, INFOCOM 2001

Advanced and Authenticated Marking Schemes for IP Traceback

- Can we do better with more information?
  - Assume map of upstream routers is known
- Encoding:
  - 11 bit for XOR of hashes of IP addresses
  - 5 bits for distance



Mark & Reconstruct

- Marking a packet (assuming map of upstream routers)

```
for each packet P
   let u be a random number from [0, 1)
   if u ≤ q then
      P.distance ← 0
      P.edge ← h(R_i)
   else
      if (P.distance == 0) then
         P.edge ← P.edge ⊕ h'(R_i)
      P.distance ← P.distance + 1
```

- Path reconstruction at victim
  - Use upstream router map
  - Guess last router, confirm by computing hash
  - Otherwise, same as before (XOR encoding…)

Finally, Your Assignment 3

- Implement *either* the Savage2000 or Song2001 IP Traceback scheme
- Implement
  - Packet marking routine
  - Path reconstructor program
  - Two need to work together
- Takes place of 2 assignments
  Due November 9, 11:59PM

Levels of Correctness

1. Basic unlimited header space, 1 attacker
2. Compact header space, 1 attacker
3. Additional *features*
   - Dropped packets
   - Premarking by attackers
   - Collisions with IP fragmentation
   - Traceback for large attacker groups

Song scheme must support multiple attackers
Savage scheme gets bonus pts for multiple attackers