

Module 7: Network Security II

IPSEC, S-BGP, DNS-SEC etc

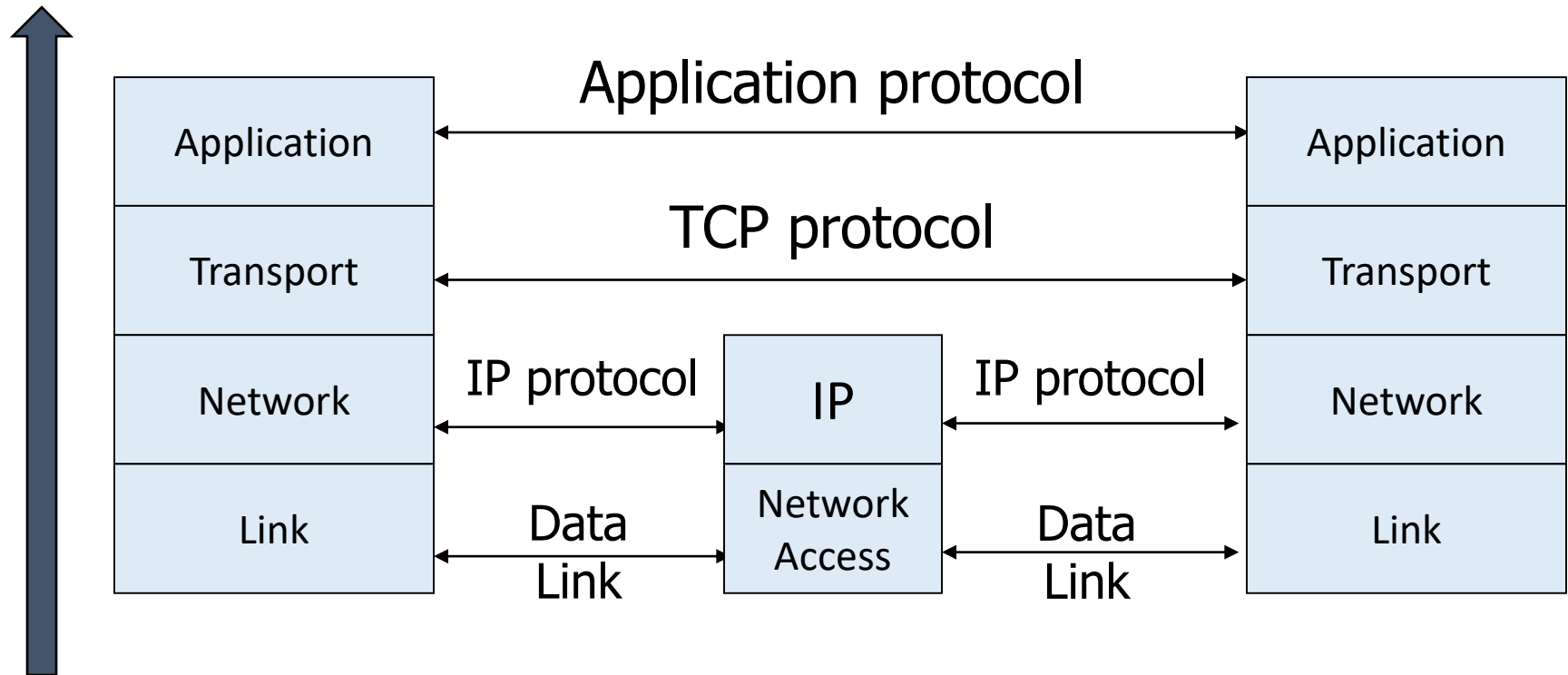
Plan for this module

- Network protocol security
 - IPSEC
 - BGP instability and secure BGP proposals
 - DNS rebinding and DNSSEC
- Standard network defenses
 - Firewall
 - Packet filter (stateless, stateful), Application layer proxies
 - Intrusion detection
 - Anomaly and misuse detection

Last Module

- Basic network protocols
 - IP, TCP, UDP, BGP, DNS
- Problems with them
 - TCP/IP
 - No SRC authentication: can't tell where packet is from
 - Packet sniffing
 - Connection spoofing, sequence numbers
 - BGP: advertise bad routes or close good ones
 - DNS: cache poisoning, rebinding
 - Web security mechanisms rely on DNS

Network Protocol Stack

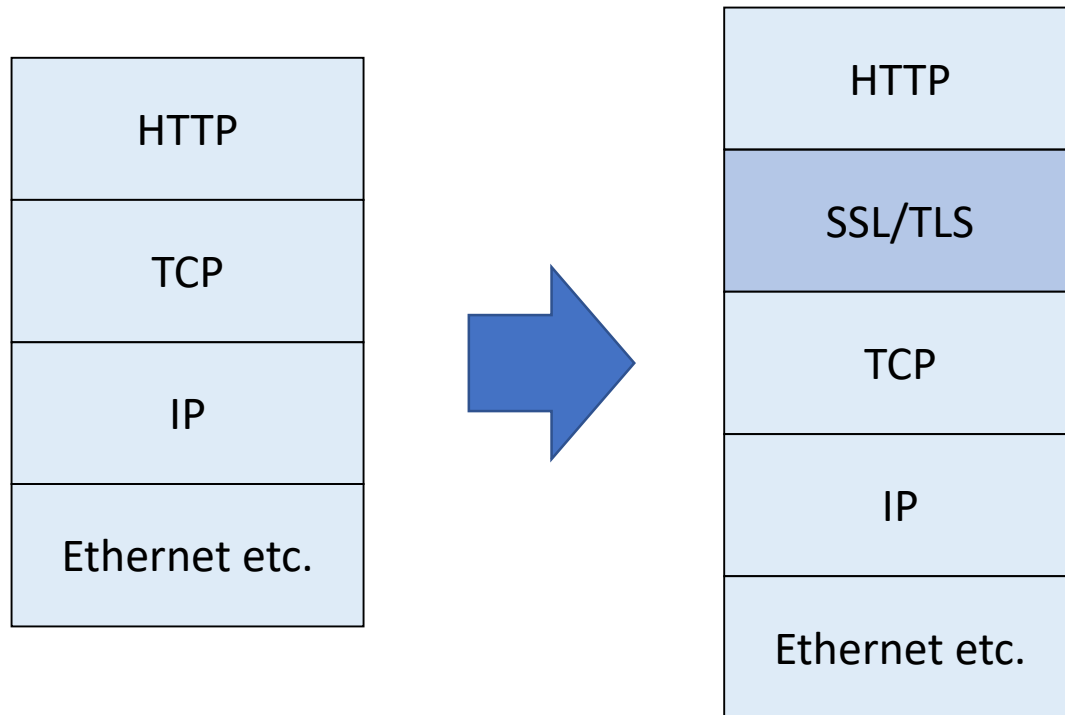


TCP/IP connectivity

Basic Layer 2-3 Security Problems

- Network packets pass by untrusted hosts
 - Eavesdropping, packet sniffing
 - Especially easy when attacker controls a machine close to victim
- TCP state can be easy to guess
 - Enables spoofing and session hijacking

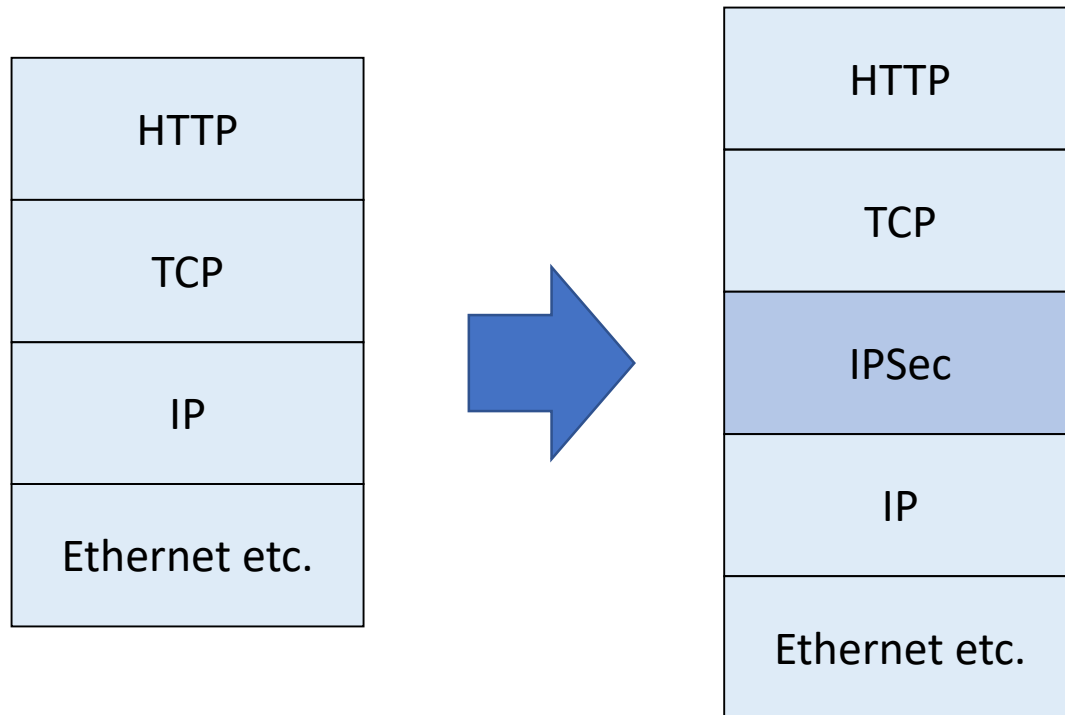
How to ensure security on top of IP?



Option 1: Add at the application level

- Needs application updates
- Specific to each application

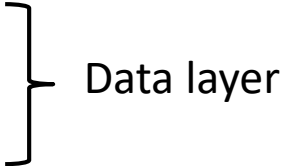
How to ensure security on top of IP?



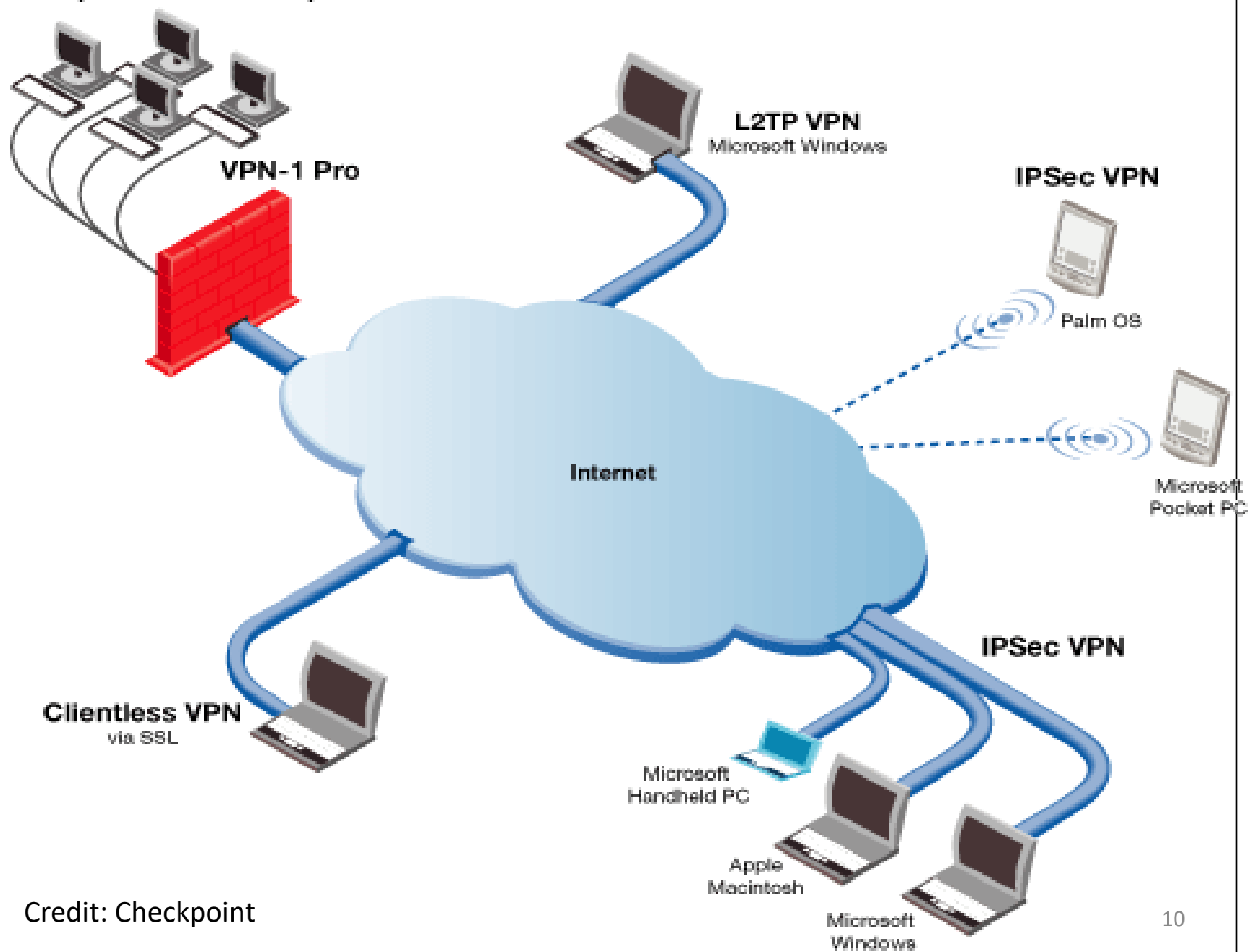
Option 2: IPsec

- Works for all applications
- Adopted widely in practice for creating VPNs

Virtual Private Network (VPN)

- Three different modes of use:
 - Remote access client connections
 - LAN-to-LAN internetworking
 - Controlled access within an intranet
 - Several different protocols
 - PPTP – Point-to-point tunneling protocol
 - L2TP – Layer-2 tunneling protocol
 - IPsec (Layer-3: network layer)
- 
- A right-facing curly bracket groups the three protocols listed under 'Several different protocols'. To the right of the bracket is the text 'Data layer'.

LAN (Trusted Network)

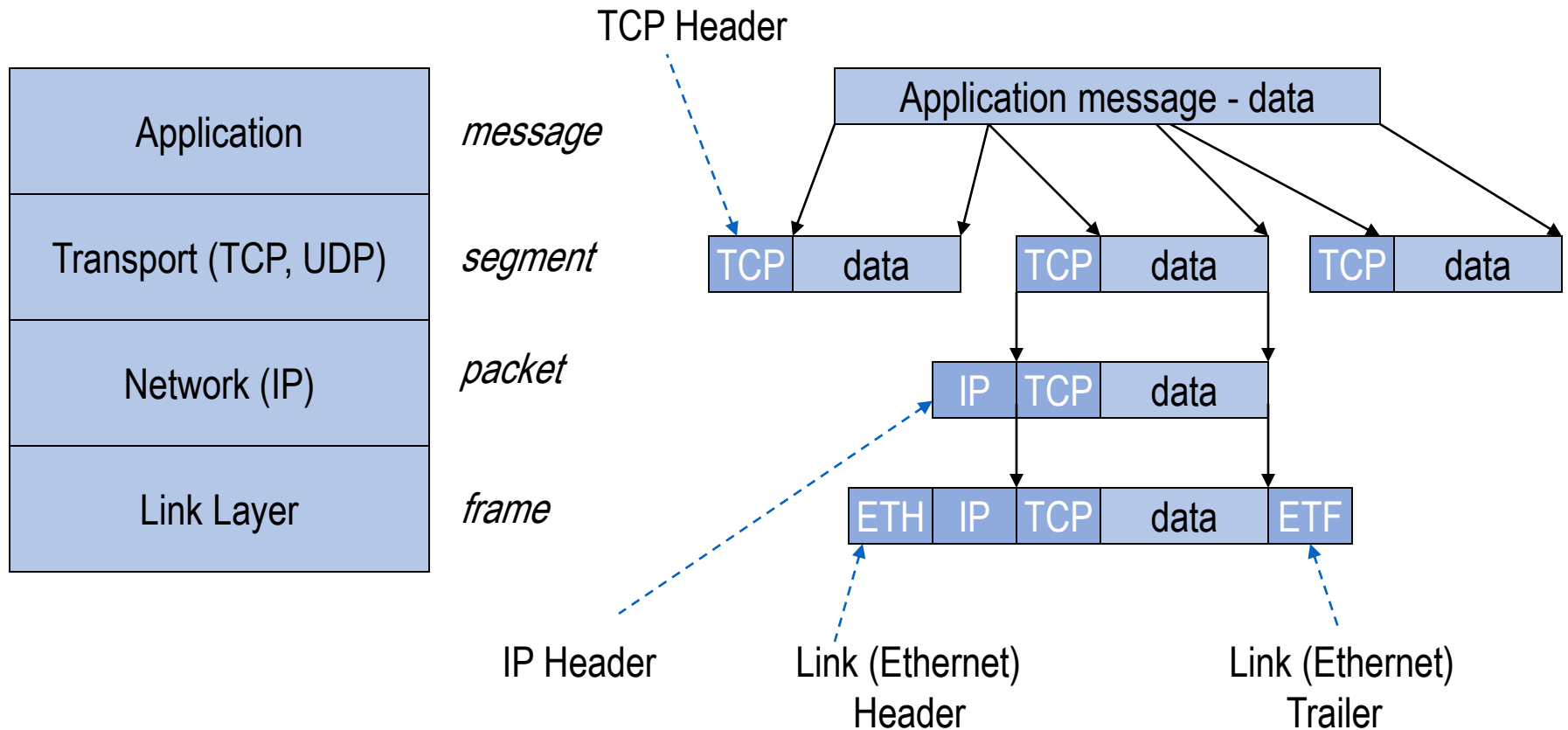


Credit: Checkpoint

IPSEC

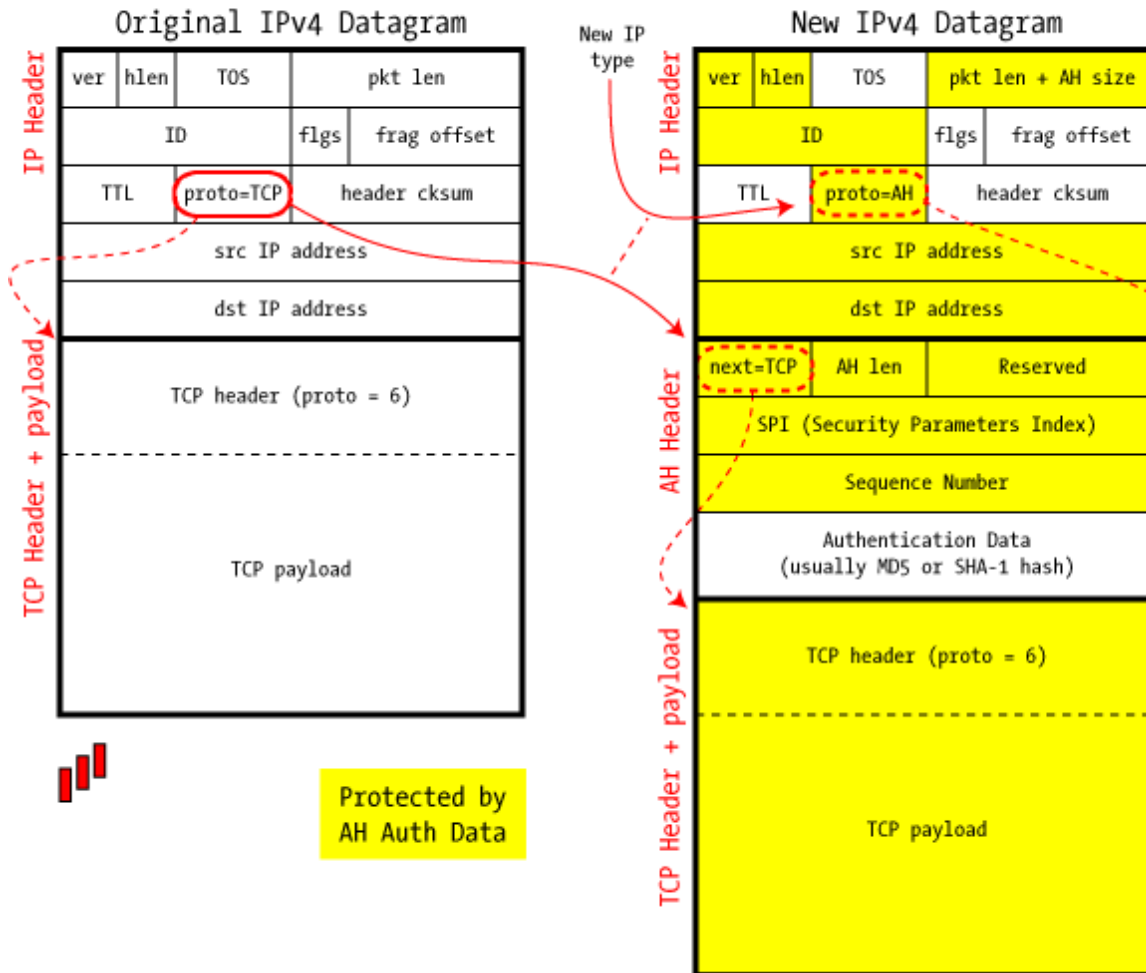
- Security extensions for IPv4 and IPv6
- IP Authentication Header (AH)
 - Authentication and integrity of payload and header
- IP Encapsulating Security Protocol (ESP)
 - Confidentiality of payload
- ESP with optional ICV (integrity check value)
 - Confidentiality, authentication and integrity of payload

Recall packet formats and layers



IPSec Authentication Header

IPSec in AH Transport Mode

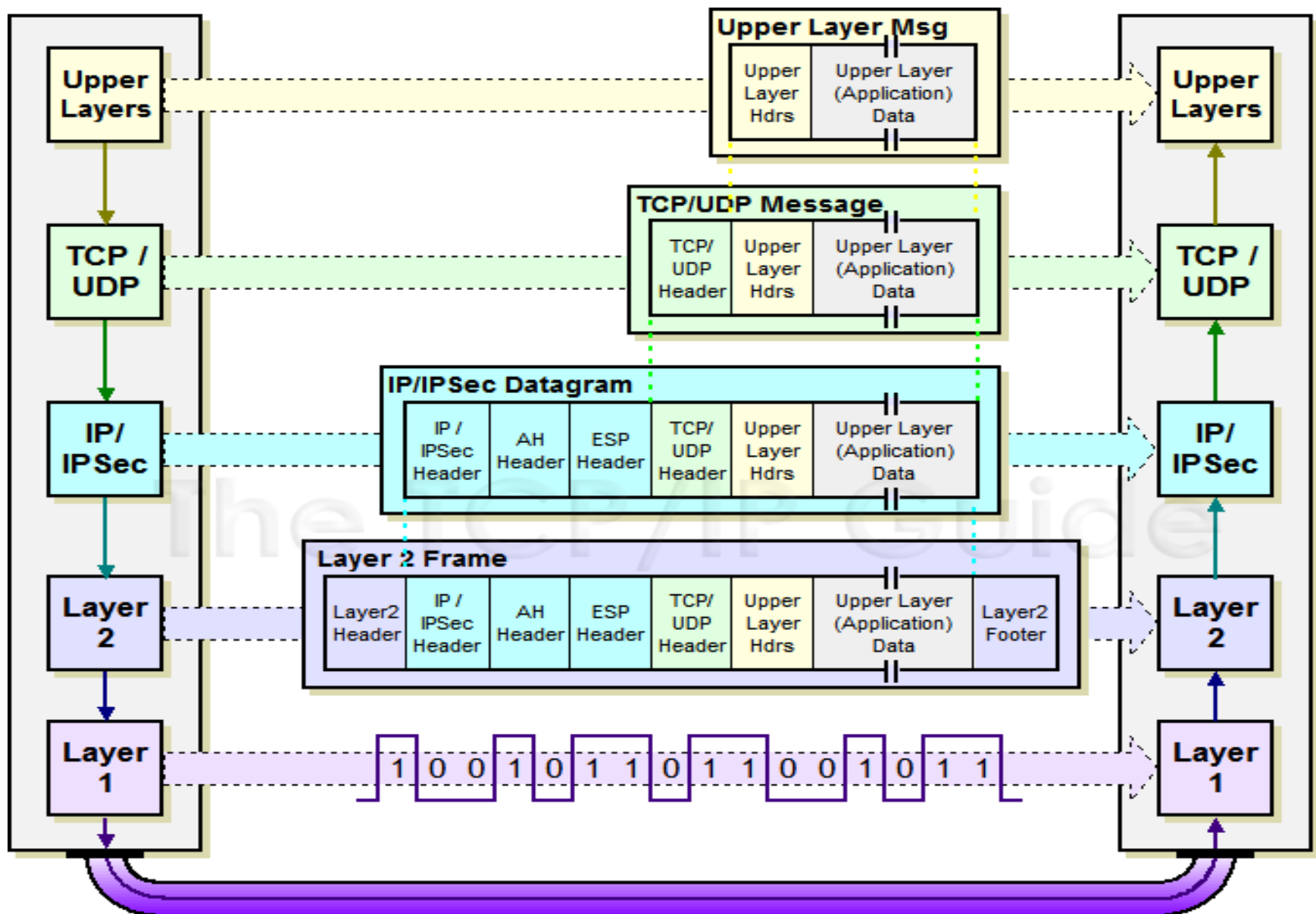


Goal: integrity

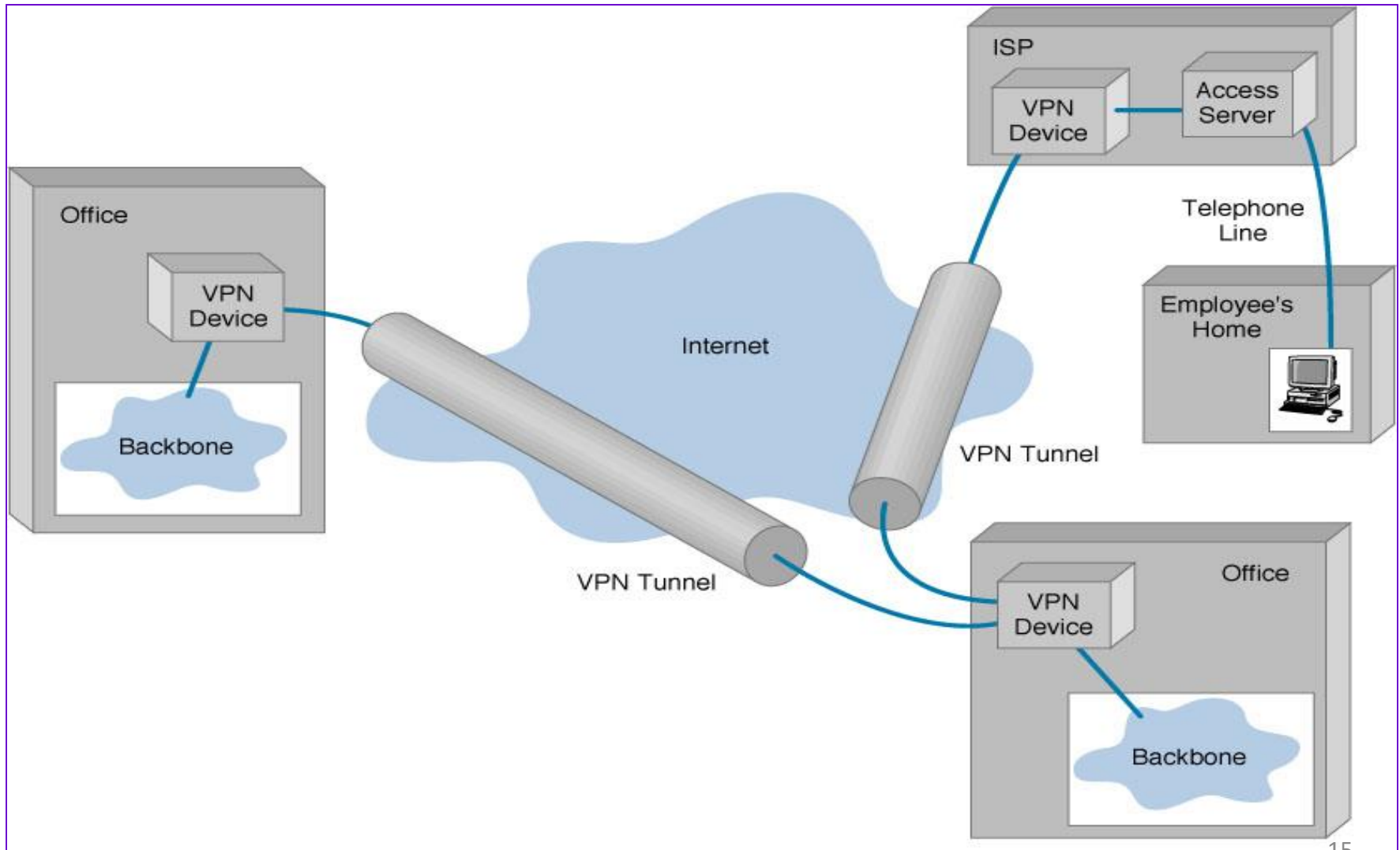
Main idea: wrap the original IP packet inside the AH with a MAC

Requires routers to be setup with a shared secret key

IPSec Transport Mode: IPSEC instead of IP header

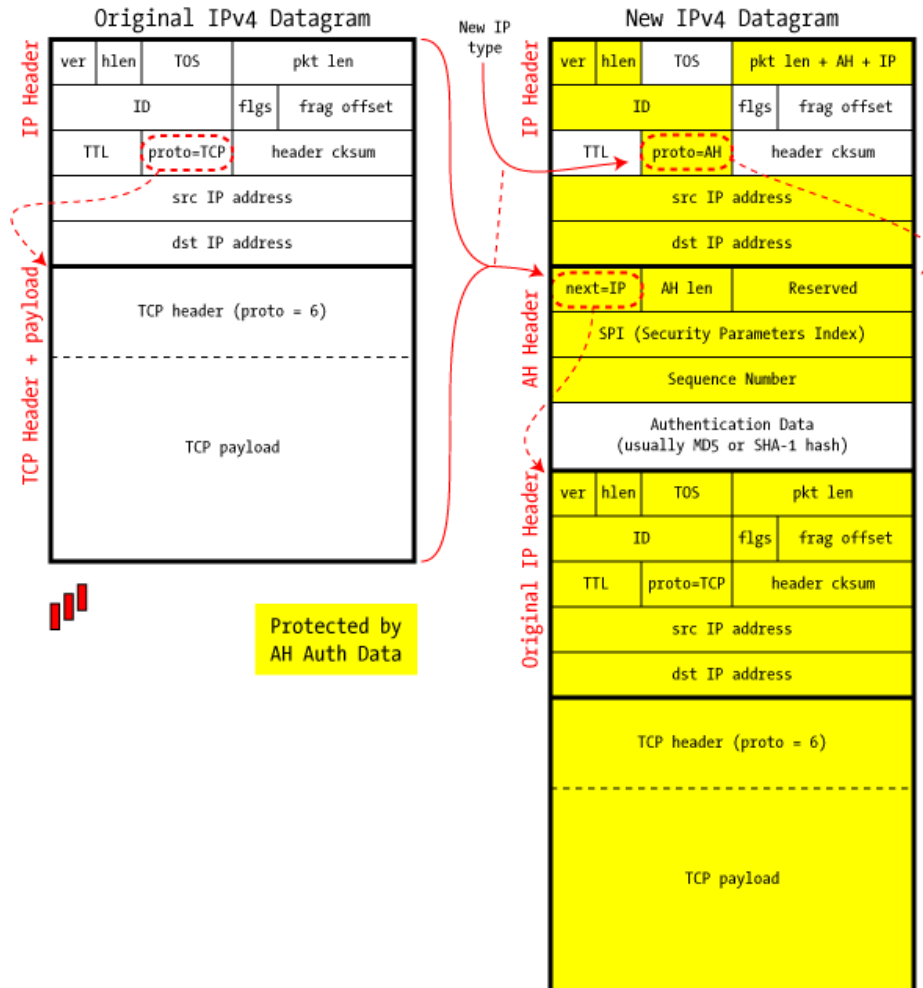


IPSEC Tunnel Mode



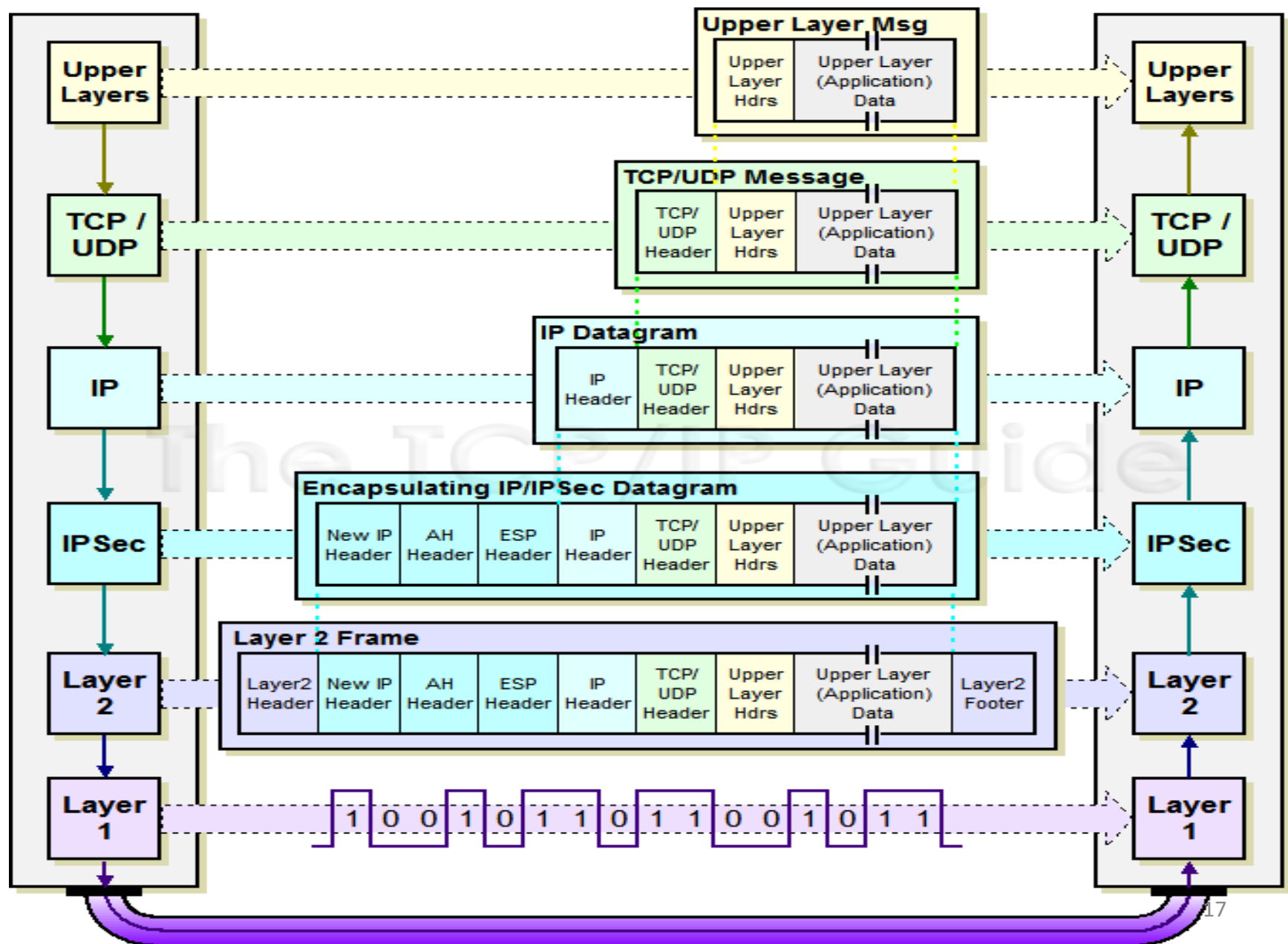
IPSec AH in Tunnel Mode

IPSec in AH Tunnel Mode

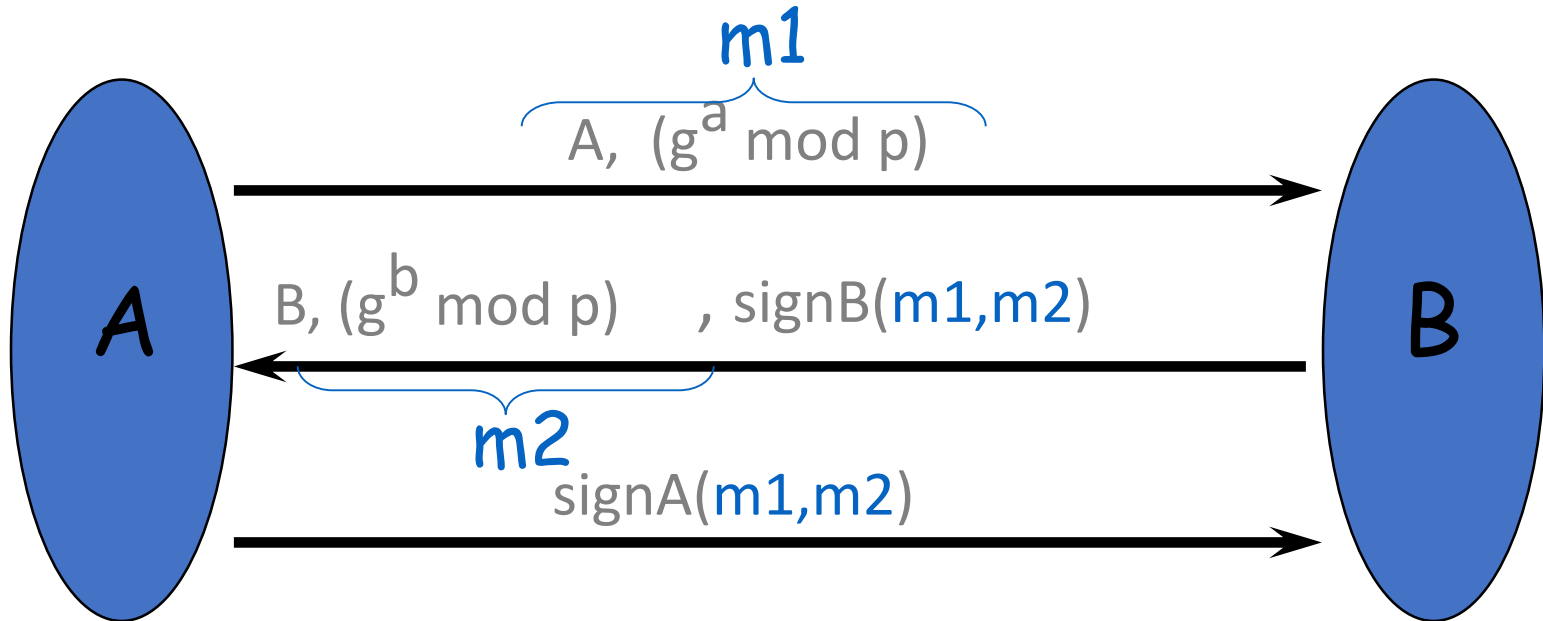


- Now there is a packet inside the packet!
- When tunnel mode packet arrives at dest, it's unwrapped and sent through usual routing process

IPSec Tunnel Mode: IPSEC header + IP header



IKE subprotocol from IPSEC



Result: A and B share secret $g^{ab} \bmod p$

Mobile IPv6 Architecture

Mobile Node (MN)



Direct connection via binding
update



Corresponding Node (CN)



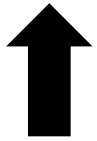
Home Agent (HA)



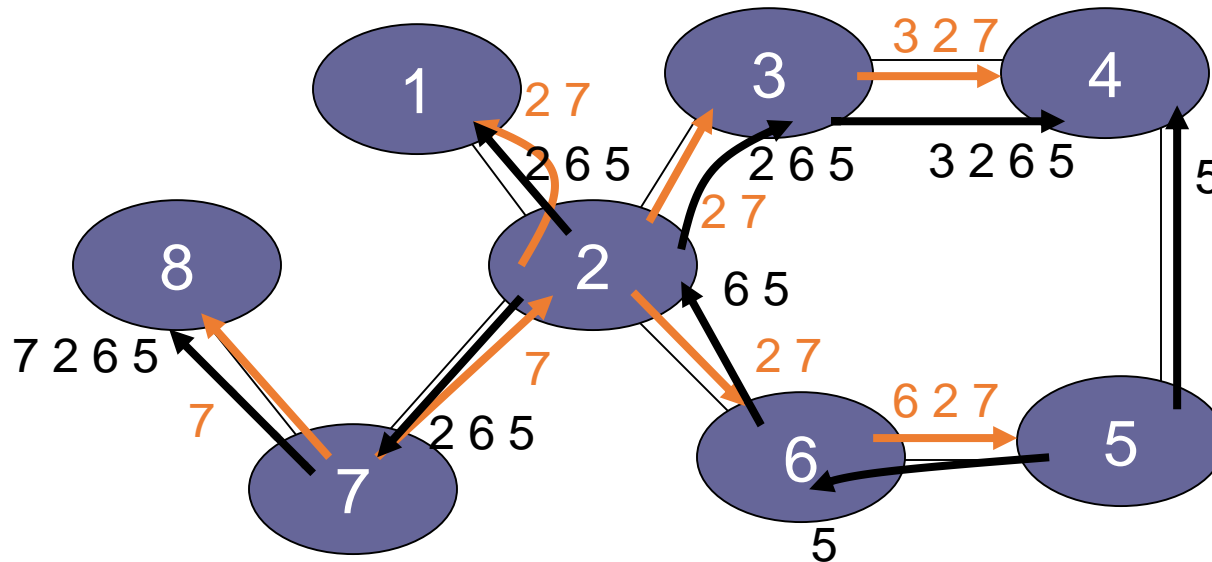
- Authentication is a requirement
- Early proposals weak

Infrastructure protocols:

BGP, DNS



BGP example



- Transit: 2 provides transit for 7
- Algorithm seems to work OK in practice
 - BGP does not respond well to frequent node outages

BGP Security Issues

- BGP is used for all inter-ISP routing
- Benign configuration errors affect about 1% of all routing table entries at any time
- Highly vulnerable to human errors, malicious attacks
 - Actual routing policies can be very complicated
- MD5 MAC is rarely used, perhaps due to lack of automated key management, addresses only one class of attacks

Laundry list of BGP attacks

- Peer spoofing and TCP resets
 - Attacker needs to be able to guess TCP seq no.
 - Special case: reset attack
 - Target router withdraws learned routes from peer
 - Can also be done using ICMP destination unreachable
- Route flapping
 - Advertise a route and then withdraw
 - Causes lots of propagations between routers
- DoS via Resource Exhaustion
- Link cutting attack (physical or through DoS)

Laundry list of BGP attacks

- Route deaggregation
 - Announce a smaller prefix
 - Consequences: overload, eavesdropping
- Malicious route injection
 - Inject a route you don't have
 - Consequences: eavesdropping, dropped packets
- Unallocated route injection
 - Regularly used by botnets
 - Make attacks untraceable

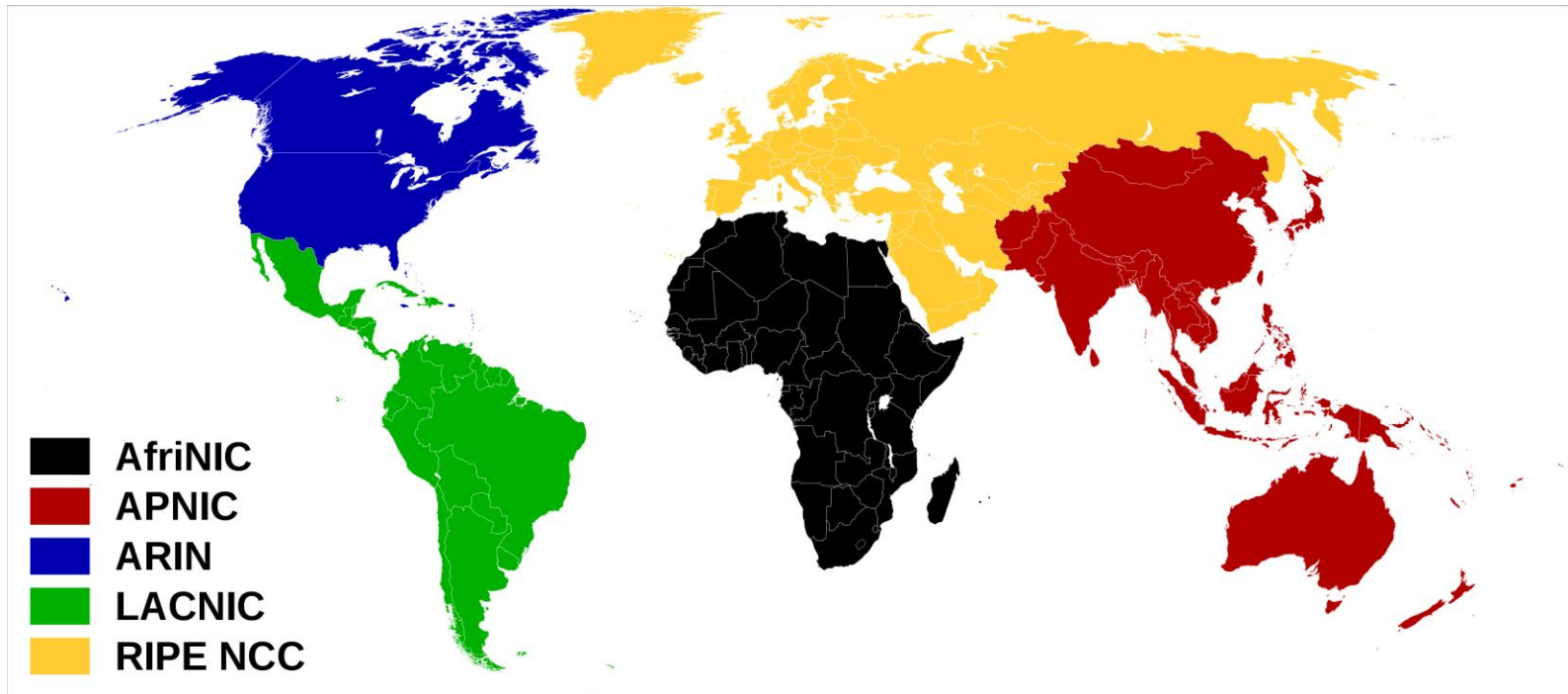
BGP attacks are serious business

- <https://arstechnica.com/information-technology/2010/11/how-china-swallowed-15-of-net-traffic-for-18-minutes/>
- <https://www.wired.com/2014/08/isp-bitcoin-theft/>
- And there is always a plausible excuse
“Sorry, I misconfigured my router!”

Many proposals to secure BGP

- Starting with S-BGP, NDSS'99
- BGPsec
- And several others
- Very little adoption in practice (unfortunately)

A bit about address allocation



- Five regional internet registries (RIRs)
- Own all addresses on the internet
- Sell/allocate these addresses to other entities (ISPs, govts)

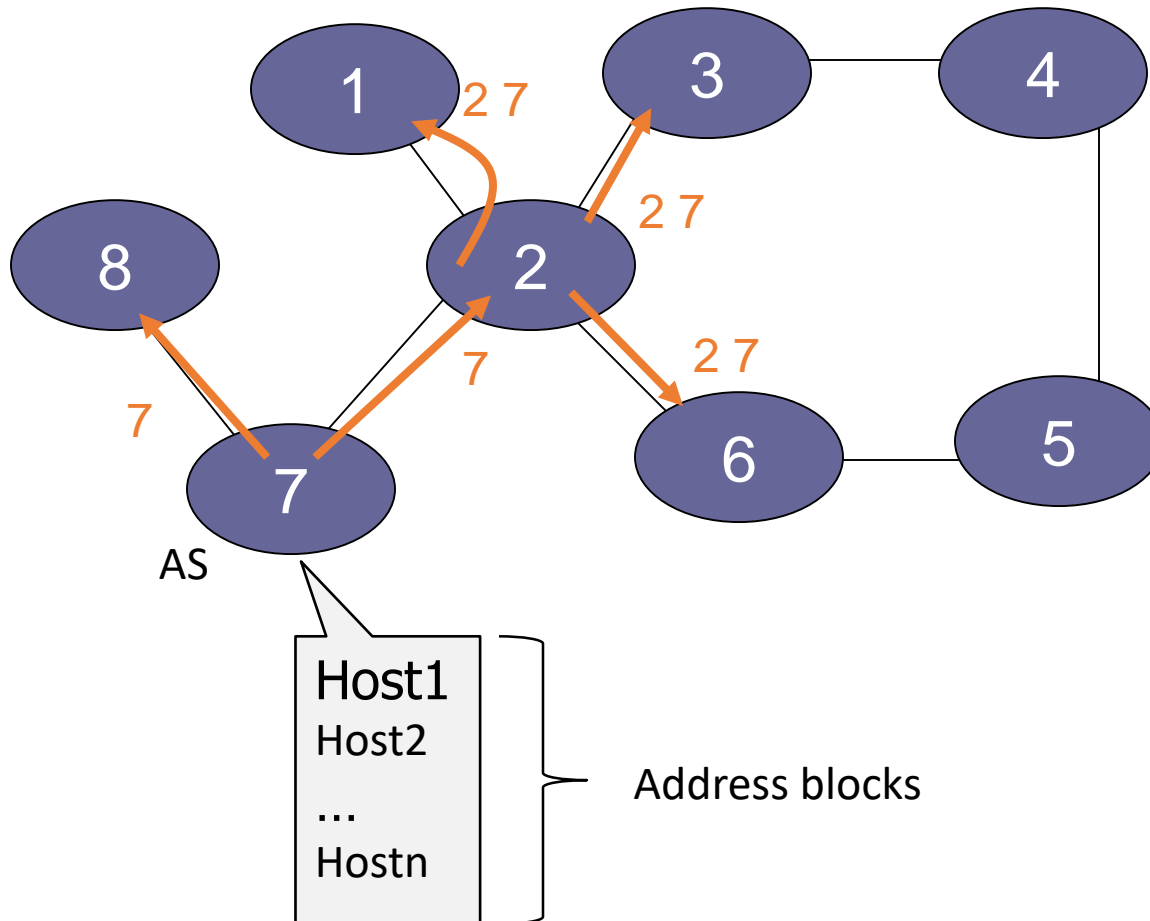
2 problems secure BGP tries to solve

- If I announce a route to an IP range, do I actually own it?
 - Called address attestation in S-BGP
 - Resource certificates + Route origin authorizations in BGPsec
- If I pass along a route to an IP range, am I allowed to pass that route along
 - Called route attestation in S-BGP
 - More difficult problem

S-BGP Design Overview

- IPsec: secure point-to-point router communication
- Public Key Infrastructure: authorization for all S-BGP entities
- Attestations: digitally-signed authorizations
 - Address: authorization to advertise specified address blocks
 - Route: Validation of UPDATES based on a new path attribute, using PKI certificates and attestations
- Repositories for distribution of certificates, CRLs, and address attestations
- Tools for ISPs to manage address attestations, process certificates & CRLs, etc.

BGP example



Address Attestation

- Indicates that the final AS listed in the UPDATE is authorized by the owner of those address blocks to advertise the address blocks in the UPDATE
- Includes identification of:
 - owner's certificate
 - AS to be advertising the address blocks
 - address blocks
 - expiration date
- Digitally signed by owner of the address blocks
- Used to protect BGP from erroneous UPDATES (authenticated but misbehaving or misconfigured BGP speakers)

Route Attestation

- Indicates that the speaker or its AS authorizes the listener's AS to use the route in the UPDATE
- Includes identification of:
 - AS's or BGP speaker's certificate issued by owner of the AS
 - the address blocks and the list of ASes in the UPDATE
 - the neighbor
 - expiration date
- Digitally signed by owner of the AS (or BGP speaker) distributing the UPDATE, traceable to the IANA ...
- Used to protect BGP from erroneous UPDATES (authenticated but misbehaving or misconfigured BGP speakers)

Validating a Route

- To validate a route from AS_n , AS_{n+1} needs:
 - address attestation from each organization owning an address block(s) in the NLRI
 - address allocation certificate from each organization owning address blocks in the NLRI
 - route attestation from every AS along the path (AS_1 to AS_n), where the route attestation for AS_k specifies the NLRI and the path up to that point (AS_1 through AS_{k+1})
 - certificate for each AS or router along path (AS_1 to AS_n) to check signatures on the route attestations
 - and, of course, all the relevant CRLs must have been checked

Infrastructure protocols:

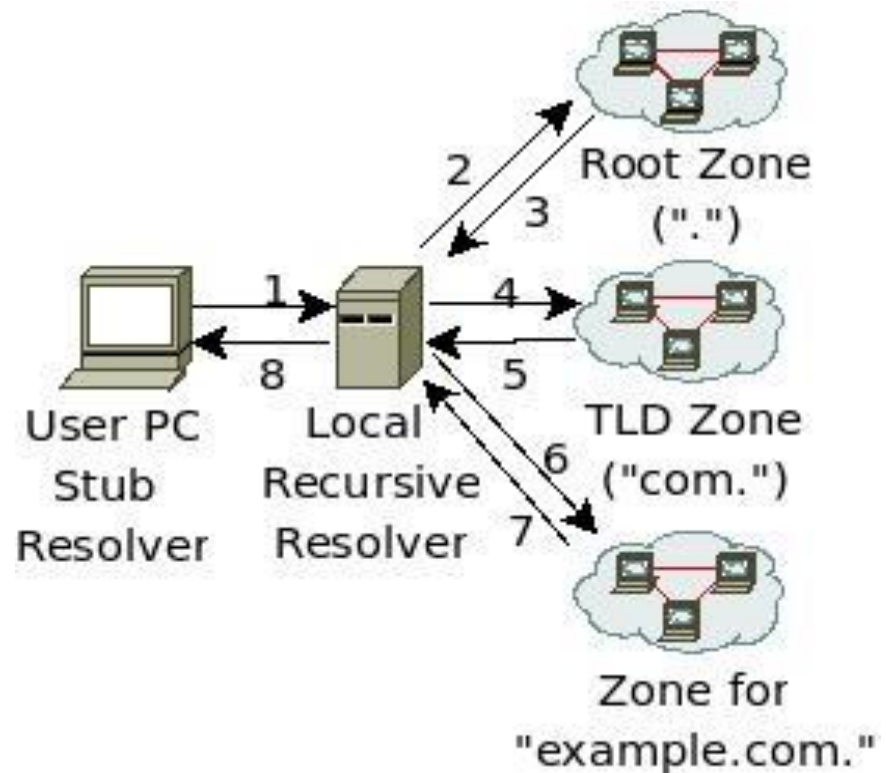
BGP, DNS



Recall: DNS Lookup

Query: "www.example.com A?"

Reply	Resource Records in Reply
3	"com. NS a.gtld.net" "a.gtld.net A 192.5.6.30"
5	"example.com. NS a.iana.net" "a.iana.net A 192.0.34.43"
7	"www.example.com A 1.2.3.4"
8	"www.example.com A 1.2.3.4"



Local recursive resolver caches these for TTL specified by RR

DNS is Insecure

- Packets sent over UDP, < 512 bytes
- 16-bit TXID, UDP Src port are only “security”
- Resolver accepts packet if above match
- Packet from whom? Was it manipulated?
- Cache poisoning
 - Attacker forges record at resolver
 - Forged record cached, attacks future lookups
 - Kaminsky (BH USA08)
 - Attacks delegations with “birthday problem”

DNSSEC Goal

“The Domain Name System (DNS) security extensions provide origin authentication and integrity assurance services for DNS data, including mechanisms for authenticated denial of existence of DNS data.”

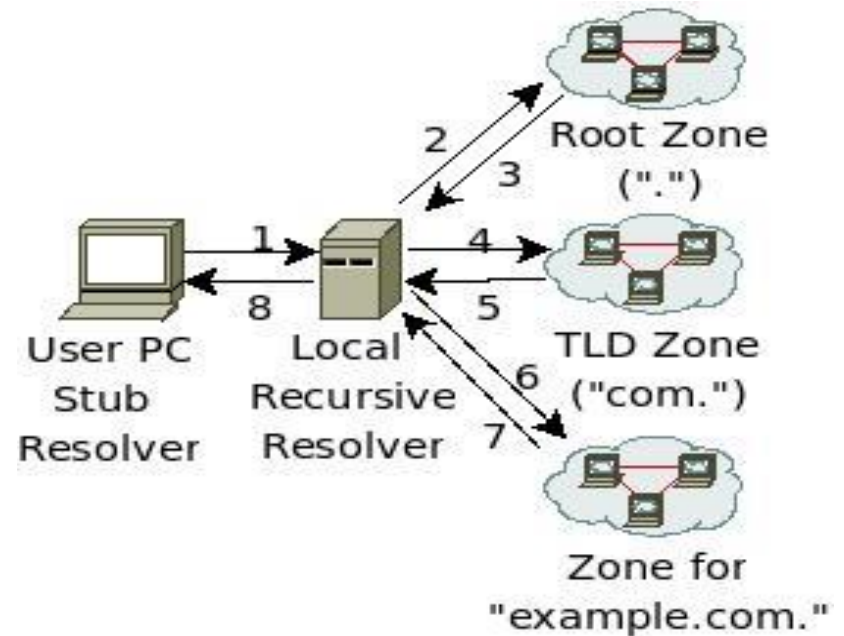
-RFC 4033

DNSSEC

- Basically no change to packet format
 - Goal is security of DNS data, not channel security
- New Resource Records (RRs)
 - RRSIG : signature of RR by private zone key
 - DNSKEY : public zone key
 - DS : crypto digest of child zone key
 - NSEC / NSEC3 authenticated denial of existence
- Lookup referral chain (unsigned)
- Origin attestation chain (PKI) (signed)
 - Start at pre-configured trust anchors
 - DS/DNSKEY of zone (should include root)
 - DS → DNSKEY → DS forms a link

DNSSEC Lookup

Query: "www.example.com A?"



Reply	RRs in DNS Reply	Added by DNSSEC
3	"com. NS a.gtld.net" "a.gtld.net A 192.5.6.30"	"com. DS" "RRSIG(DS) by ."
5	"example.com. NS a.iana.net" "a.iana.net A 192.0.34.43"	"com. DNSKEY" "RRSIG(DNSKEY) by com." "example.com. DS" "RRSIG(DS) by com."
7	"www.example.com A 1.2.3.4"	"example.com DNSKEY" "RRSIG(DNSKEY) by example.com." "RRSIG(A) by example.com."
8	"www.example.com A 1.2.3.4"	Last Hop?

Authenticated Denial-of-Existence

- Most DNS lookups result in denial-of-existence
- NSEC (Next SECure)
 - Lists all extant RRs associated with an owner name
 - Points to next owner name with extant RR
 - Easy zone enumeration
- NSEC3
 - Hashes owner names
 - Public salt to prevent pre-computed dictionaries
 - NSEC3 chain in hashed order
 - Opt-out bit for TLDs to support incremental adoption
 - For TLD type zones to support incremental adoption
 - Non-DNSSEC children not in NSEC3 chain

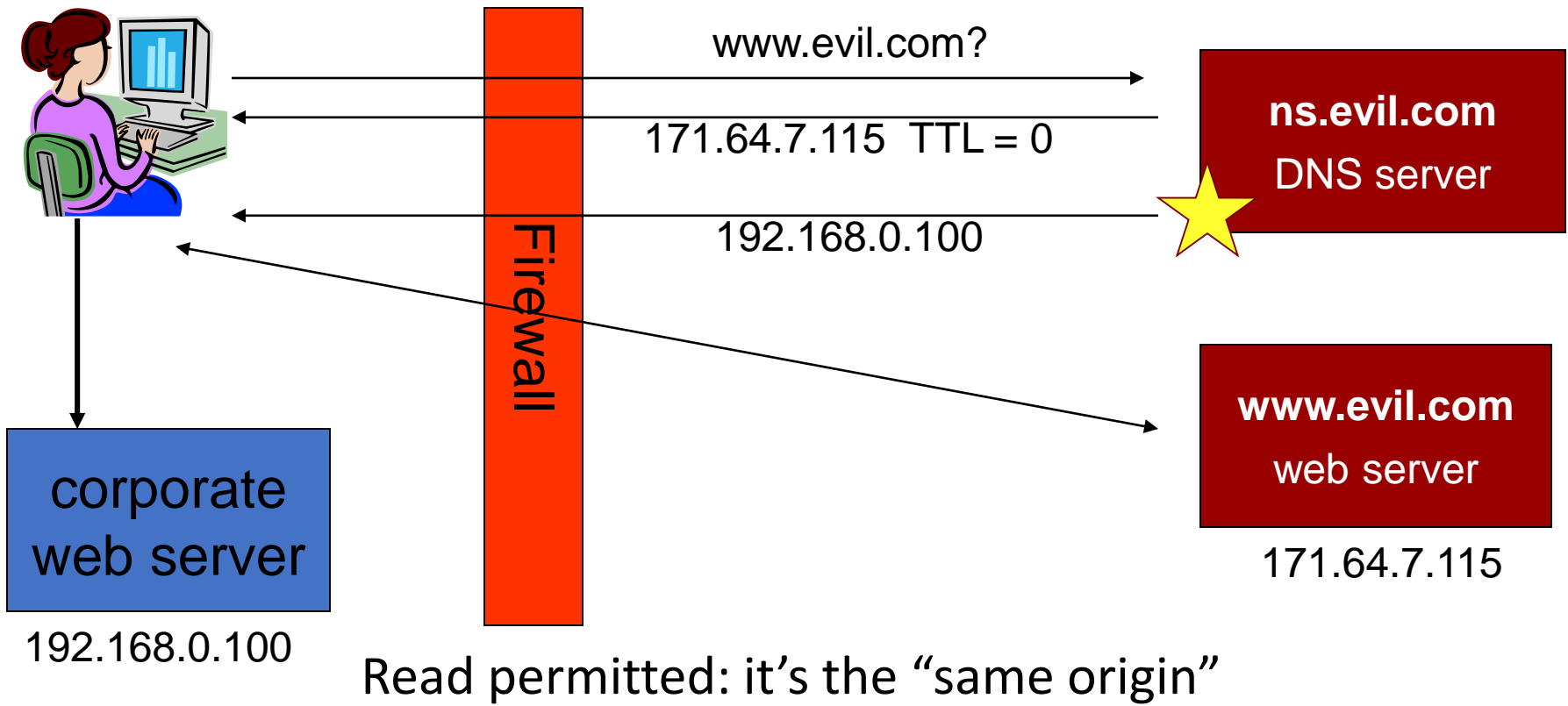
Insecure Sub-Namespace

- NSEC3 Opt-out
 - "Does not assert the existence or non-existence of the insecure delegations that it may cover" (RFC 5155)
 - Only thing asserting this is insecure glue records
- Property: Possible to insert bogus pre-pended name into otherwise secure zone. (RFC 5155)
- Insecure delegation from secure zone
 - Spoofs possible for resultant lookup results
- Acceptable for TLD, bad for enterprises

DNS Rebinding Attack

`<iframe src="http://www.evil.com">`

DNSSEC cannot
stop this attack



DNS Rebinding Defenses

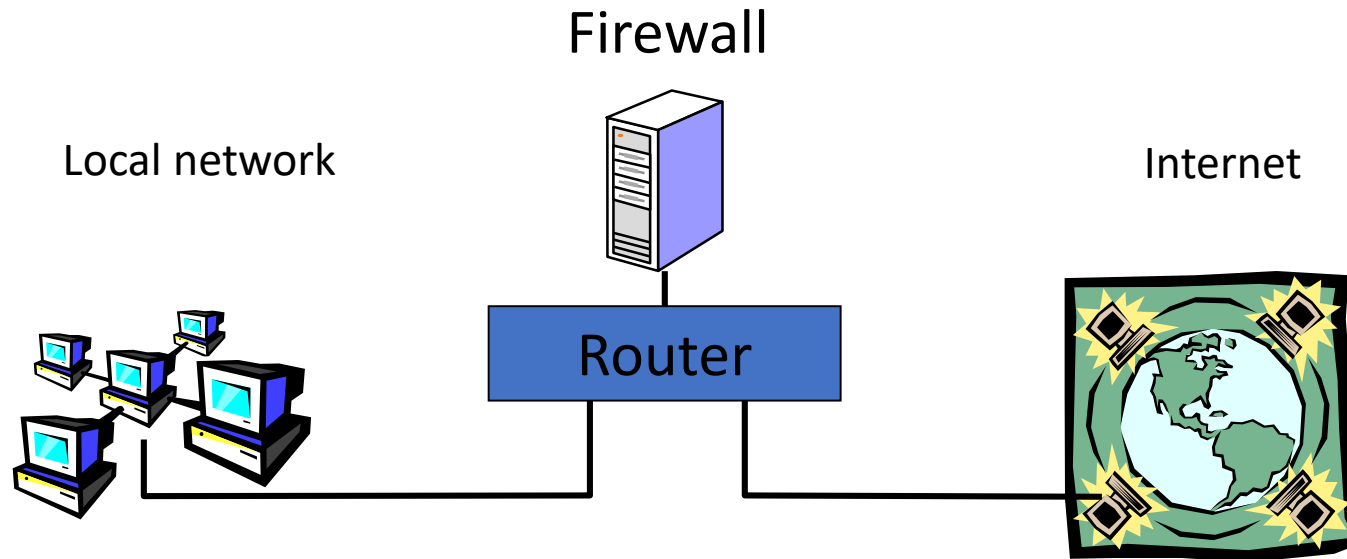
- Browser mitigation: DNS Pinning
 - Refuse to switch to a new IP
 - Interacts poorly with proxies, VPN, dynamic DNS, ...
 - Not consistently implemented in any browser
- Server-side defenses
 - Check Host header for unrecognized domains
 - Authenticate users with something other than IP
- Firewall defenses
 - External names can't resolve to internal addresses
 - Protects browsers inside the organization

Filtering network traffic

(starting at IP, transport layer ...)

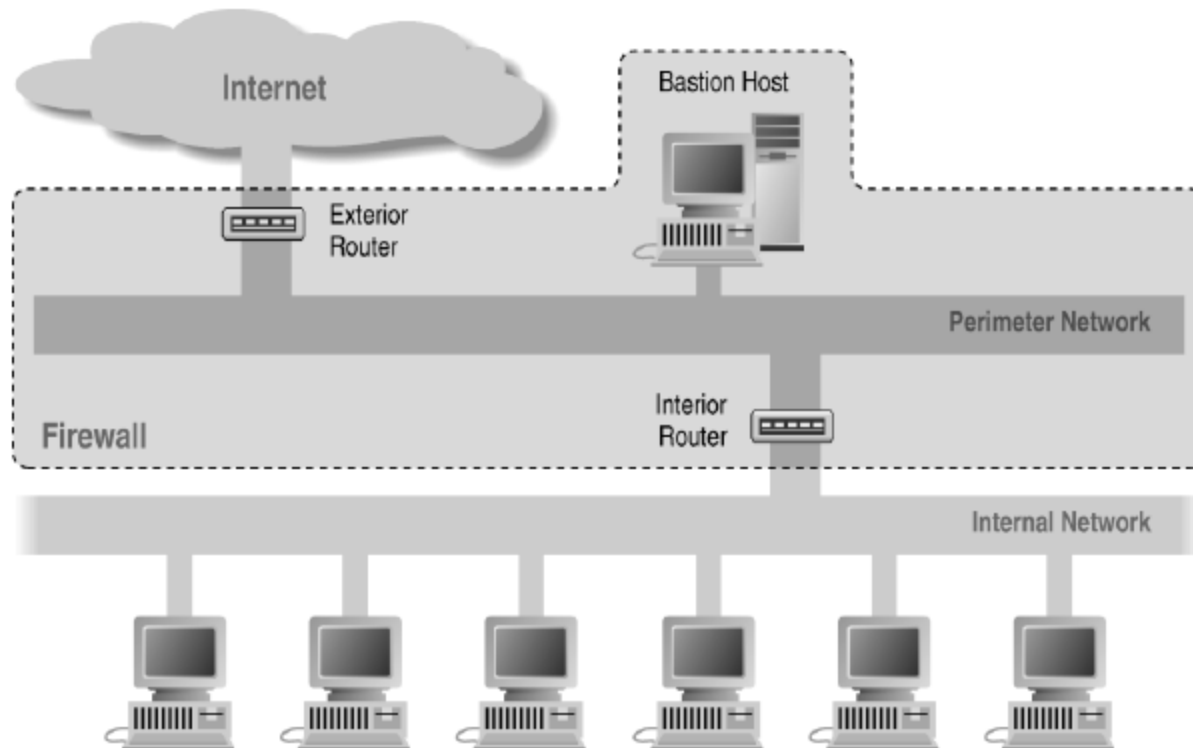
Basic Firewall Concept

- Separate local area net from internet

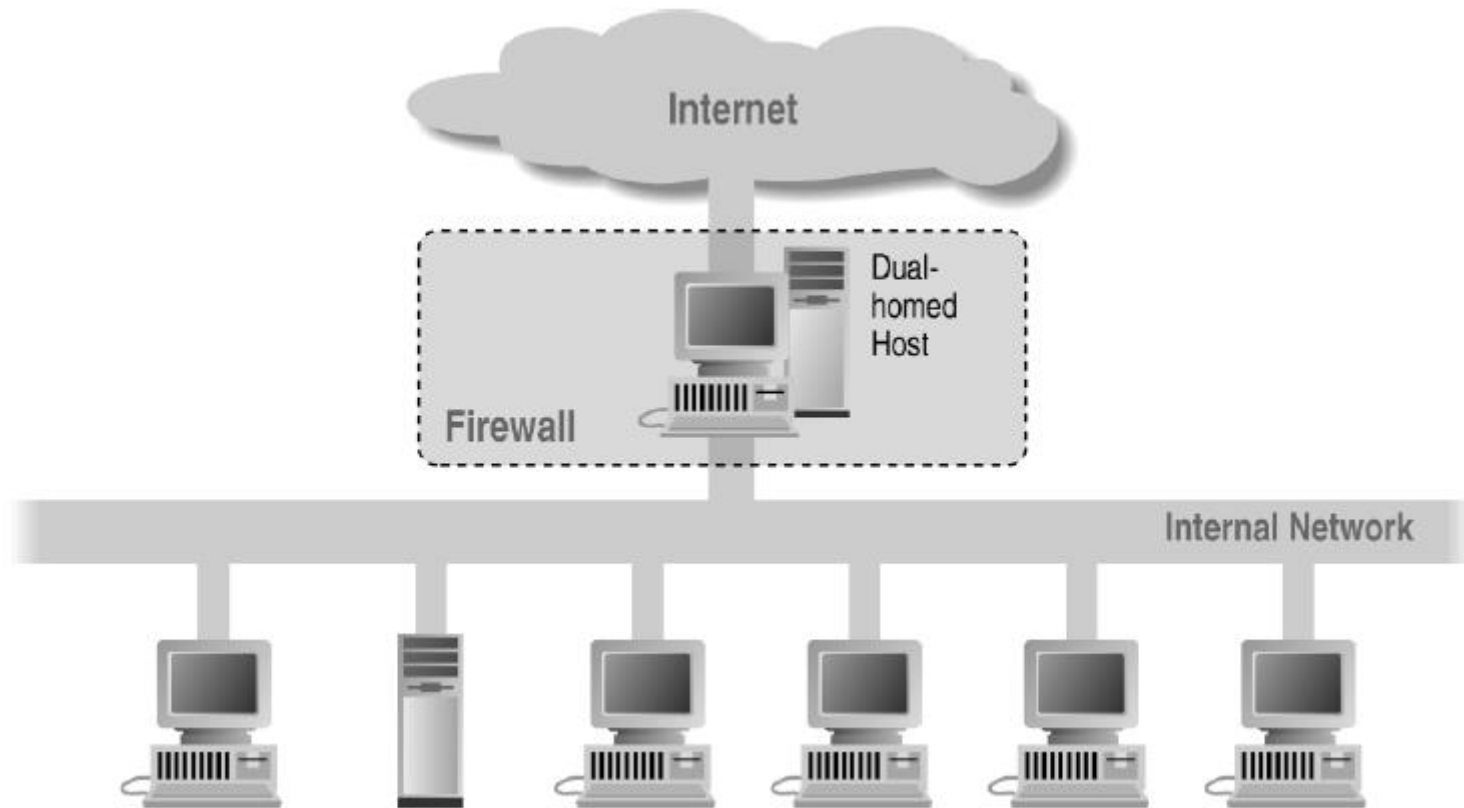


All packets between LAN and internet routed through firewall

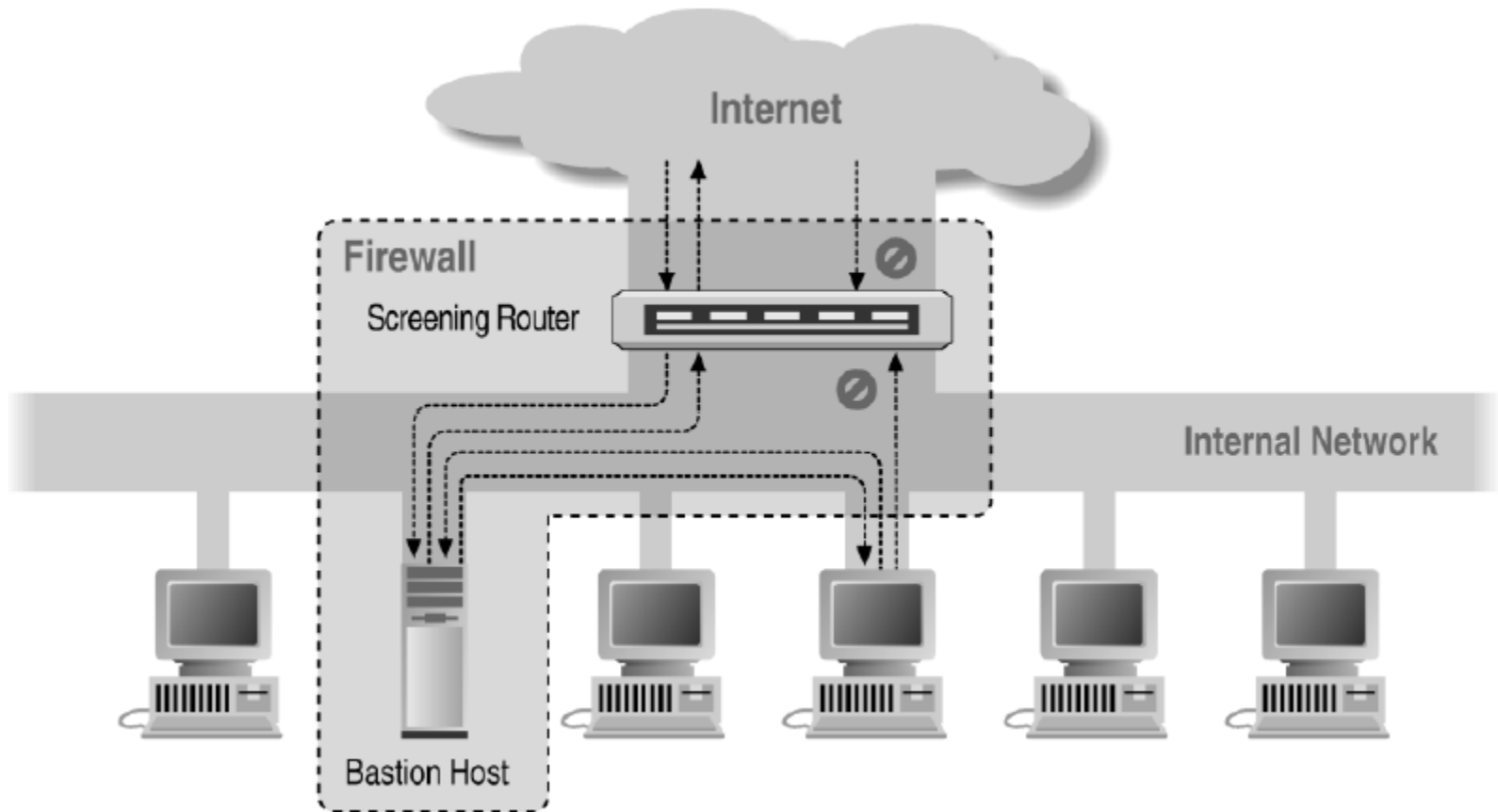
Screened Subnet Using Two Routers



Alternate 1: Dual-Homed Host



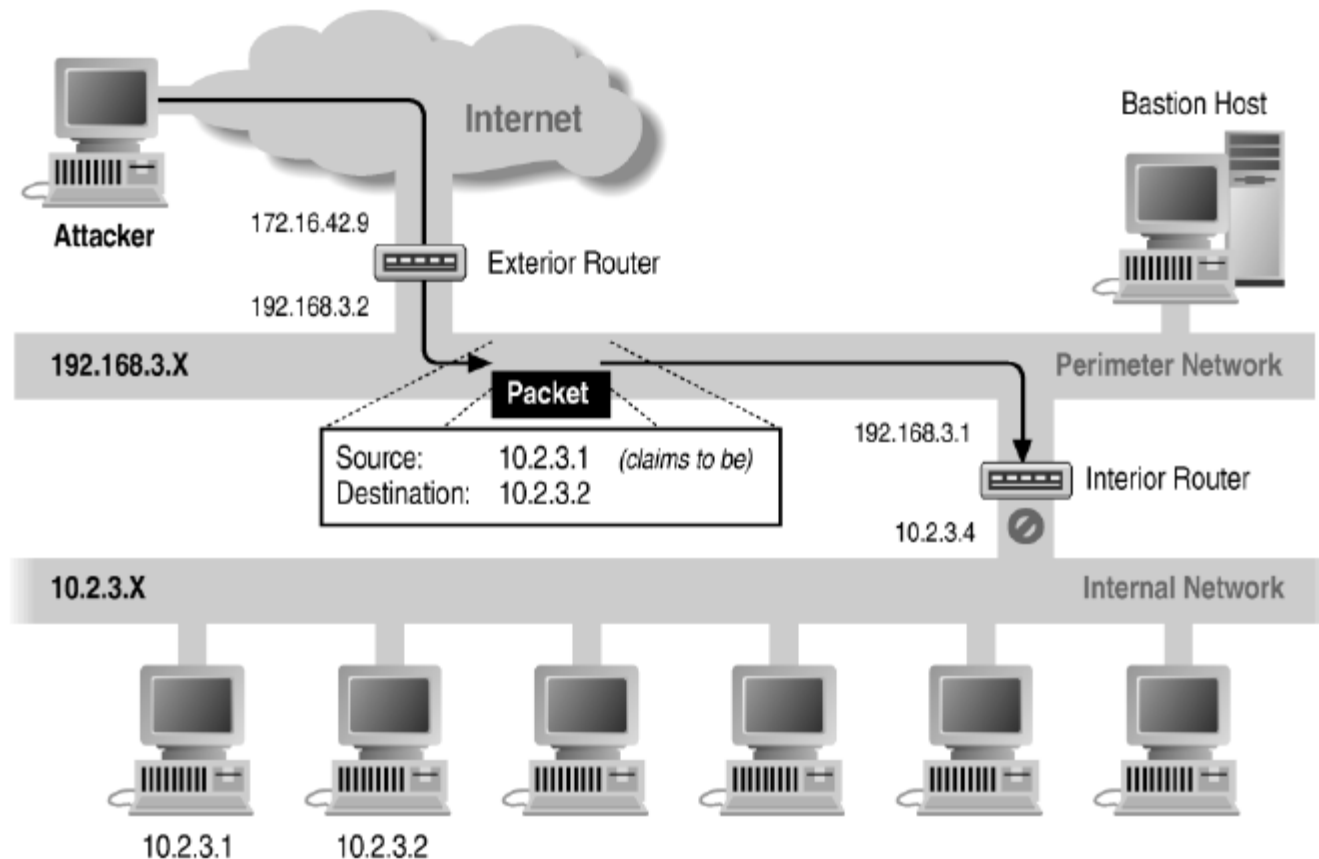
Alternate 2: Screened Host



Basic Packet Filtering

- Uses transport-layer information only
 - IP Source Address, Destination Address
 - Protocol (TCP, UDP, ICMP, etc)
 - TCP or UDP source & destination ports
 - TCP Flags (SYN, ACK, FIN, RST, PSH, etc)
 - ICMP message type
- Examples
 - DNS uses port 53
 - Block incoming port 53 packets except known trusted servers
- Issues
 - Stateful filtering
 - Encapsulation: address translation, other complications
 - Fragmentation

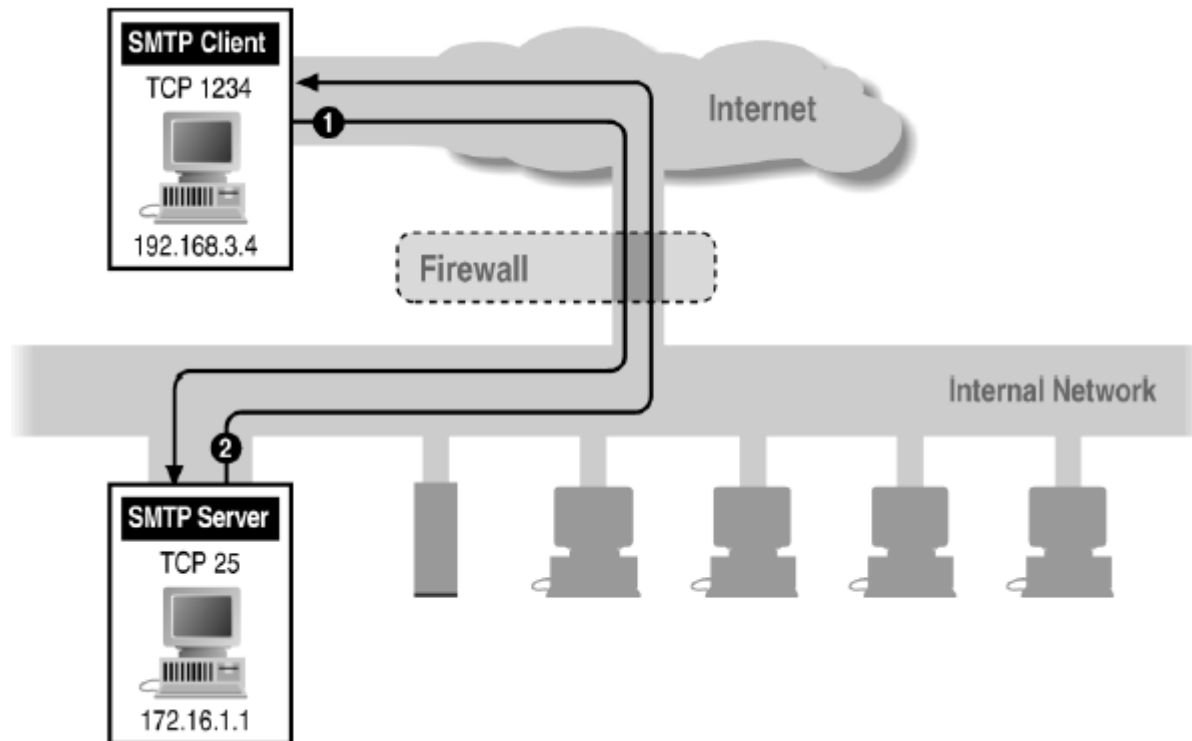
Source/Destination Address Forgery



More about networking: port numbering

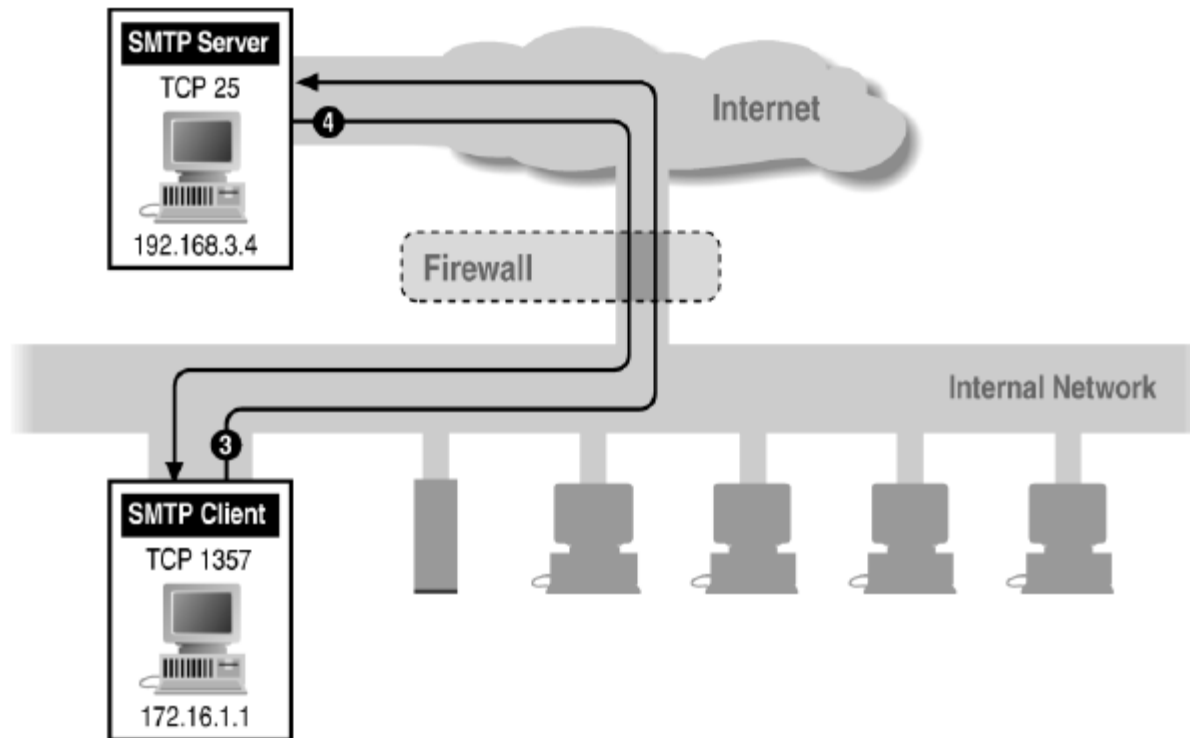
- TCP connection
 - Server port uses number less than 1024
 - Client port uses number between 1024 and 16383
- Permanent assignment
 - Ports <1024 assigned permanently
 - 20,21 for FTP 23 for Telnet
 - 25 for server SMTP 80 for HTTP
- Variable use
 - Ports >1024 must be available for client to make connection
 - Limitation for stateless packet filtering
 - If client wants port 2048, firewall must allow incoming traffic
 - Better: stateful filtering knows outgoing requests
 - Only allow incoming traffic on high port to a machine that has initiated an outgoing request on low port

Filtering Example: Inbound SMTP



Can block external request to internal server based on port number

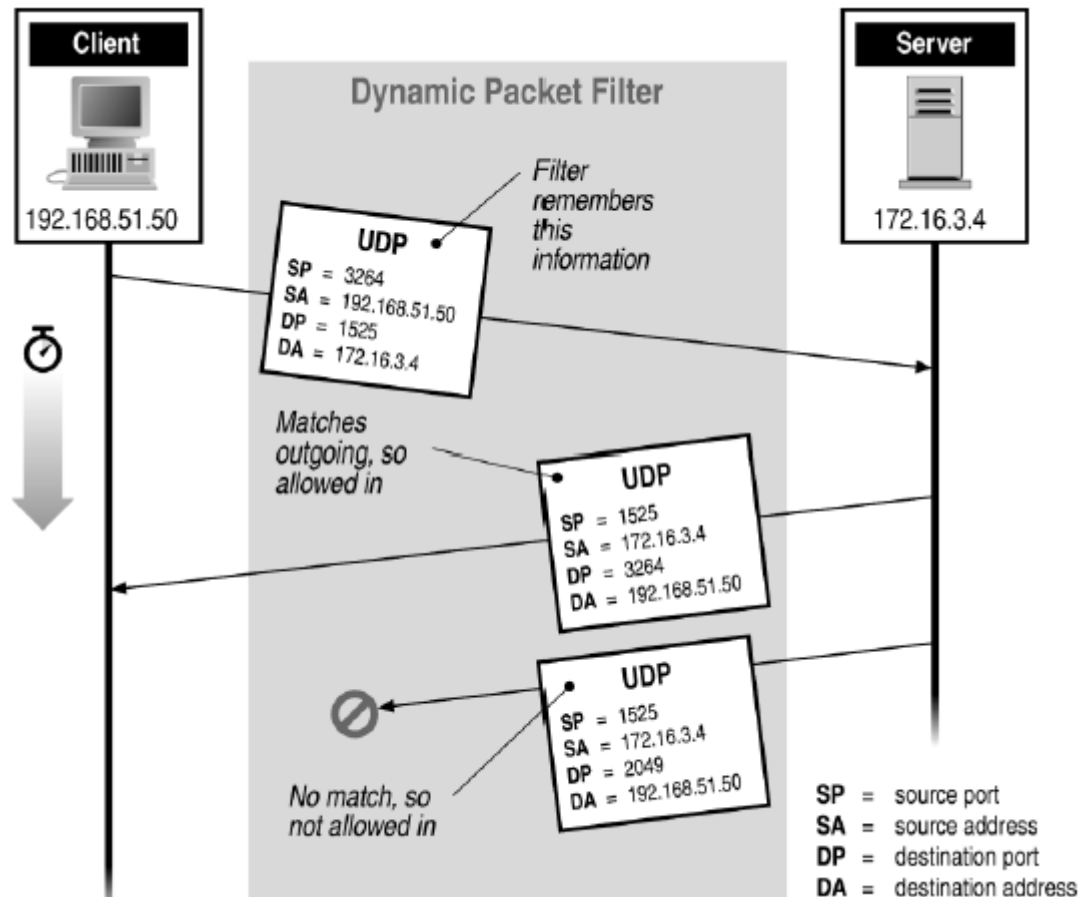
Filtering Example: Outbound SMTP



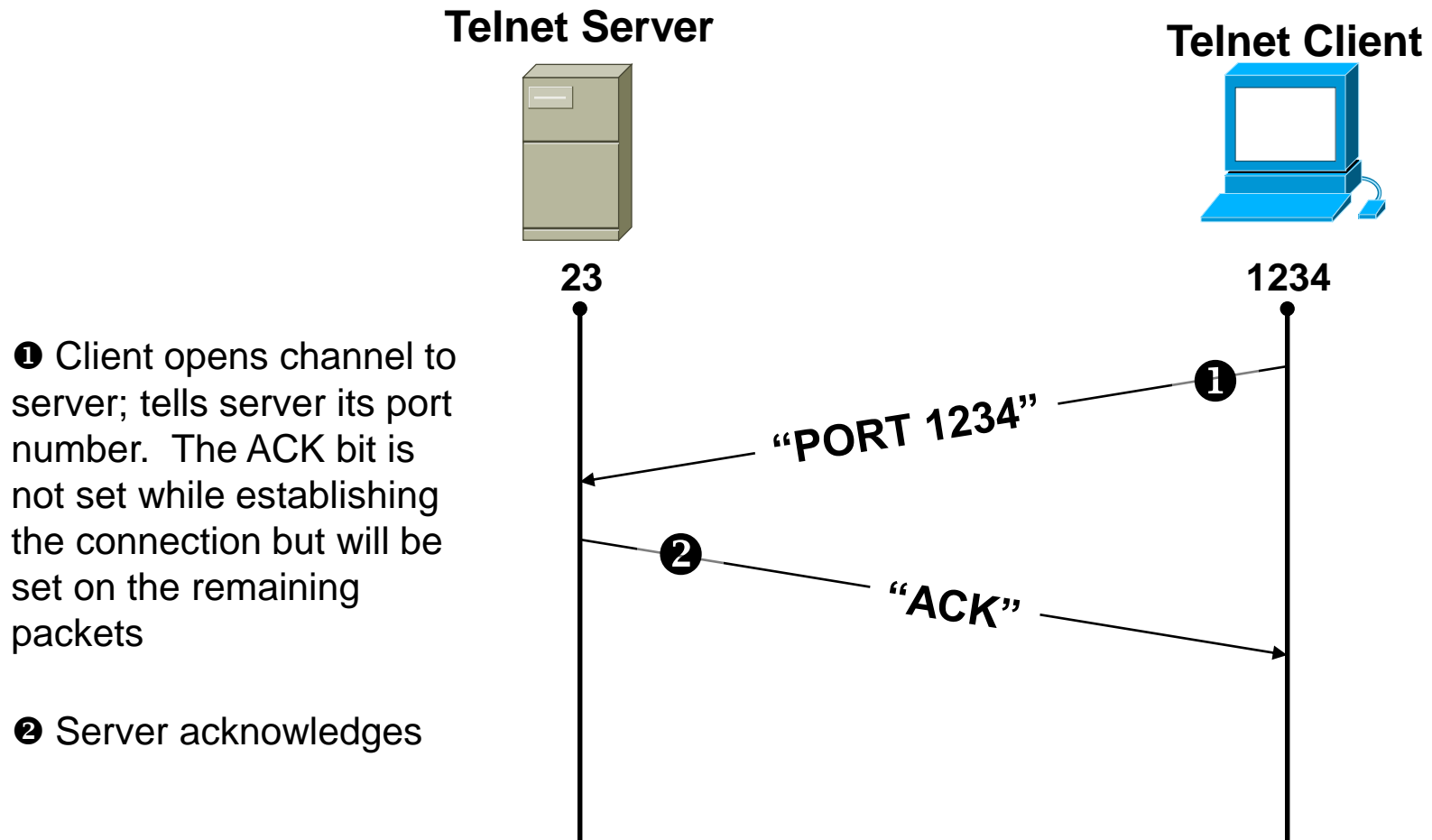
Known low port out, arbitrary high port in

If firewall blocks incoming port 1357 traffic then connection fails

Stateful or Dynamic Packet Filtering



Telnet



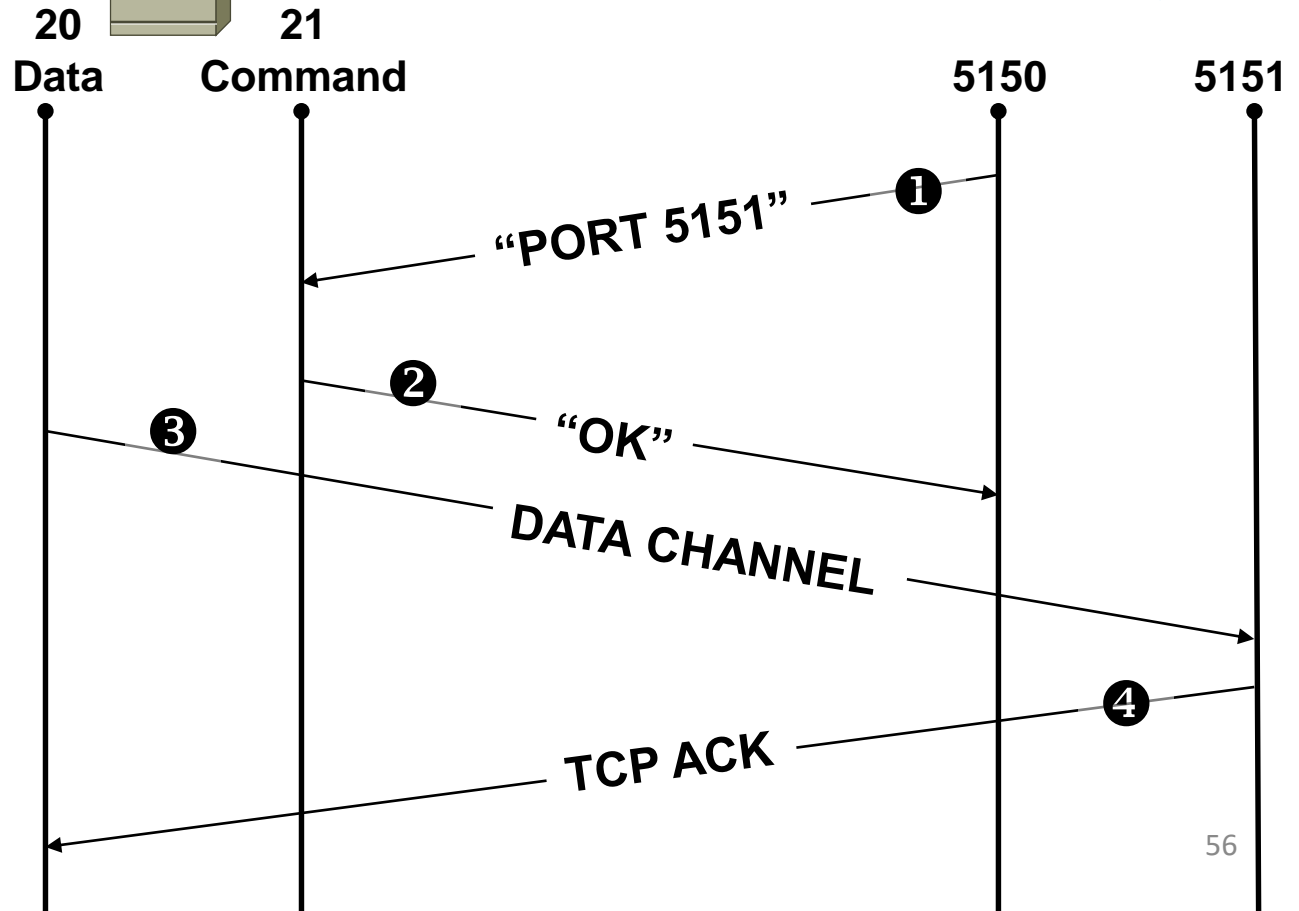
Stateful filtering can use this pattern to identify legitimate sessions

FTP

FTP Server



FTP Client



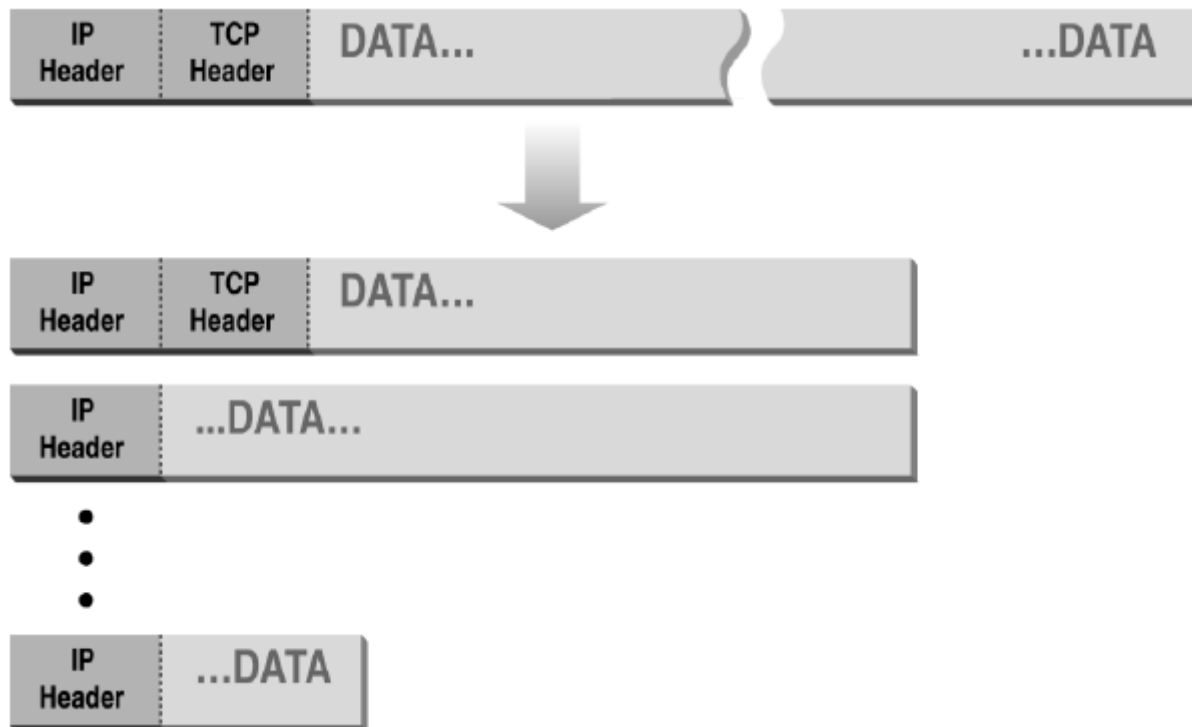
❶ Client opens command channel to server; tells server second port number

❷ Server acknowledges

❸ Server opens data channel to client's second port

❹ Client acknowledges

Normal IP Fragmentation



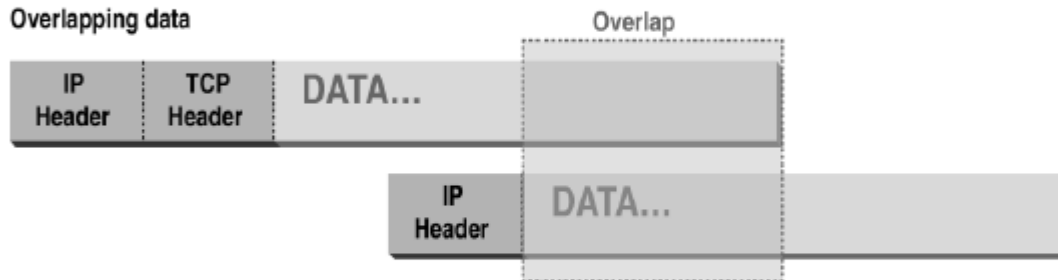
Flags and offset inside IP header indicate packet fragmentation

Abnormal Fragmentation

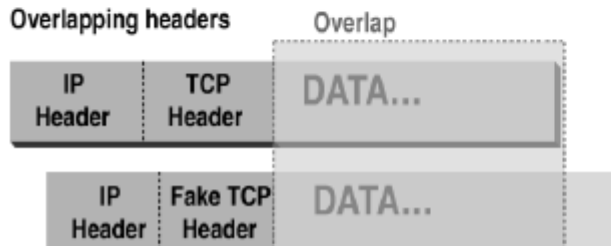
Normal



Overlapping data



Overlapping headers

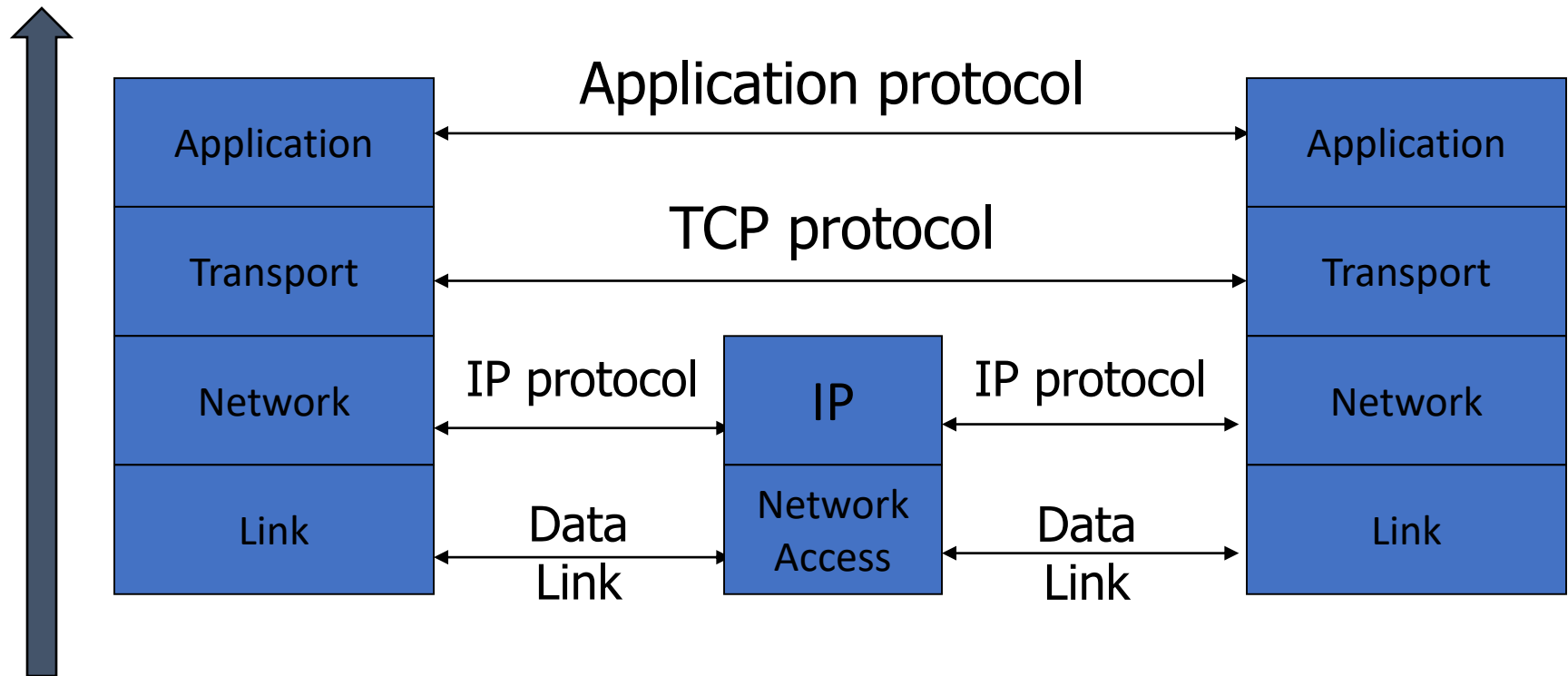


Low offset allows second packet to overwrite TCP header at receiving host

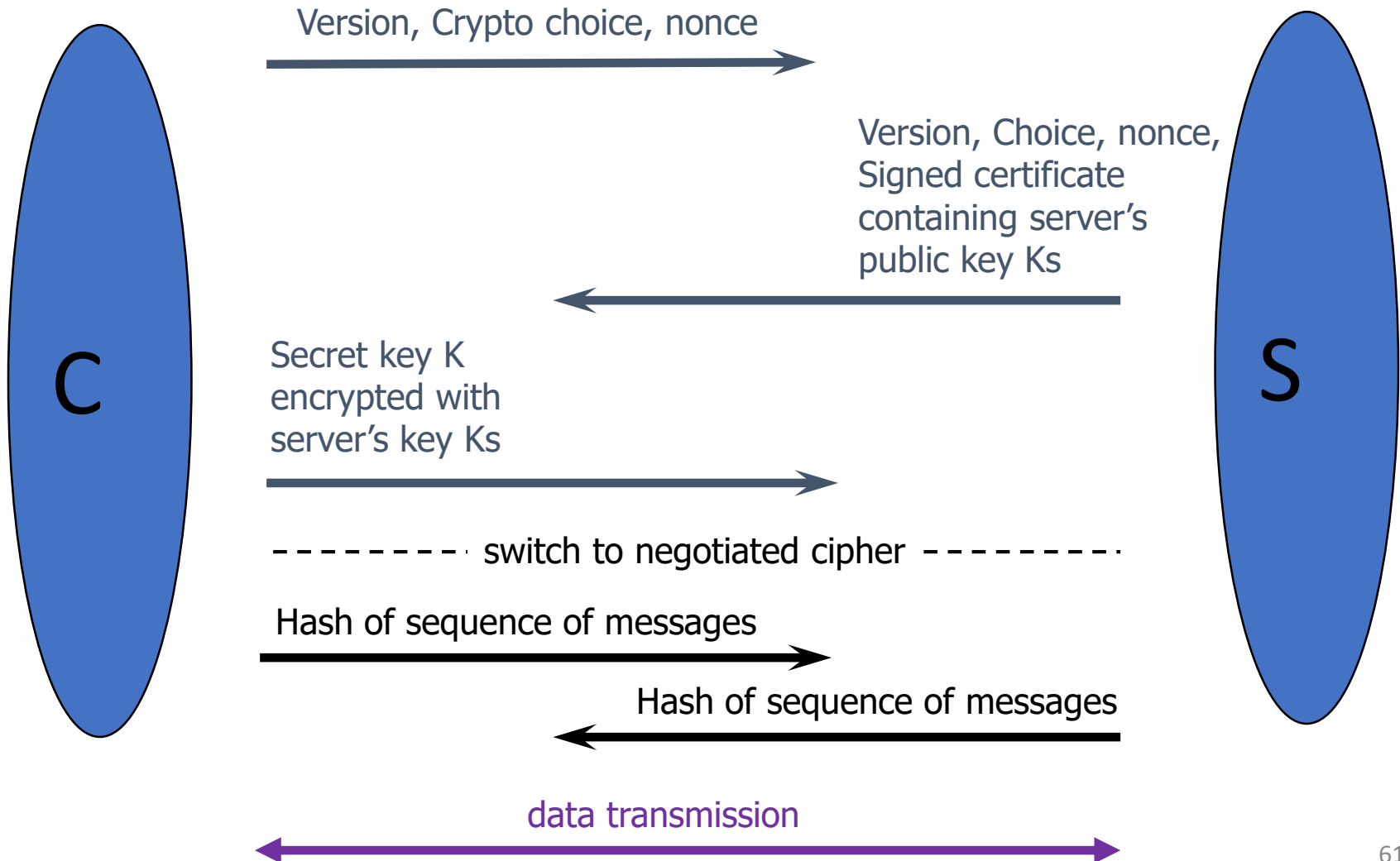
Packet Fragmentation Attack

- Firewall configuration
 - TCP port 23 is blocked but SMTP port 25 is allowed
- First packet
 - Fragmentation Offset = 0.
 - DF bit = 0 : "May Fragment"
 - MF bit = 1 : "More Fragments"
 - Destination Port = 25. TCP port 25 is allowed, so firewall allows packet
- Second packet
 - Fragmentation Offset = 1: second packet overwrites all but first 8 bits of the first packet
 - DF bit = 0 : "May Fragment"
 - MF bit = 0 : "Last Fragment."
 - Destination Port = 23. Normally be blocked, but sneaks by!
- What happens
 - Firewall ignores second packet "TCP header" because it is fragment of first
 - At host, packet reassembled and received at port 23

TCP Protocol Stack



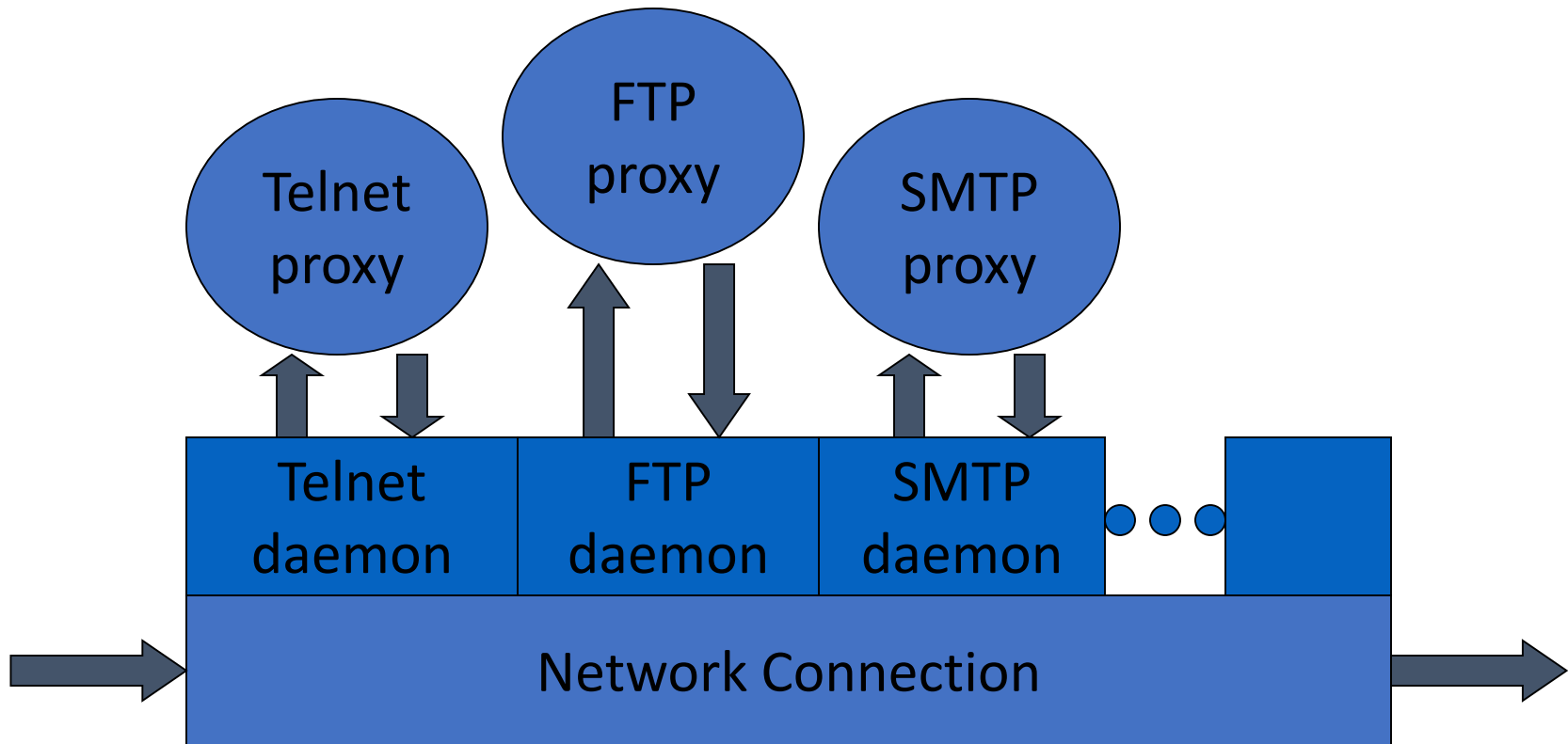
Remember SSL/TLS



Proxying Firewall

- Application-level proxies
 - Tailored to http, ftp, smtp, etc.
 - Some protocols easier to proxy than others
- Policy embedded in proxy programs
 - Proxies filter incoming, outgoing packets
 - Reconstruct application-layer messages
 - Can filter specific application-layer commands, etc.
 - Example: only allow specific ftp commands
 - Other examples: ?
- Several network locations – see next slides

Firewall with application proxies



Daemon spawns proxy when communication detected ...

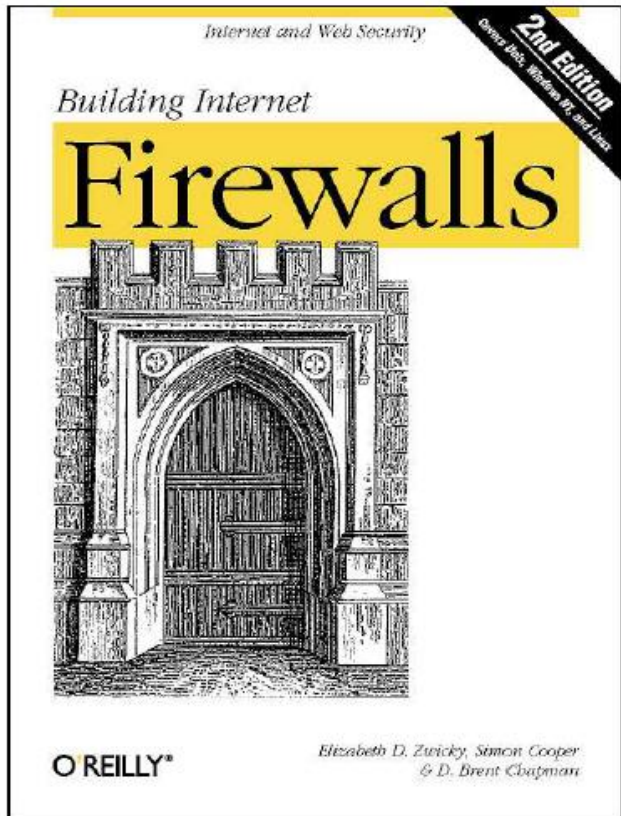
Application-level proxies

- Enforce policy for specific protocols
 - E.g., Virus scanning for SMTP
 - Need to understand MIME, encoding, Zip archives
 - Flexible approach, but may introduce network delays
- “Batch” protocols are natural to proxy
 - SMTP (E-Mail) NNTP (Net news)
 - DNS (Domain Name System) NTP (Network Time Protocol)
- Must protect host running protocol stack
 - Disable all non-required services; keep it simple
 - Install/modify services you want
 - Run security audit to establish baseline
 - Be prepared for the system to be compromised

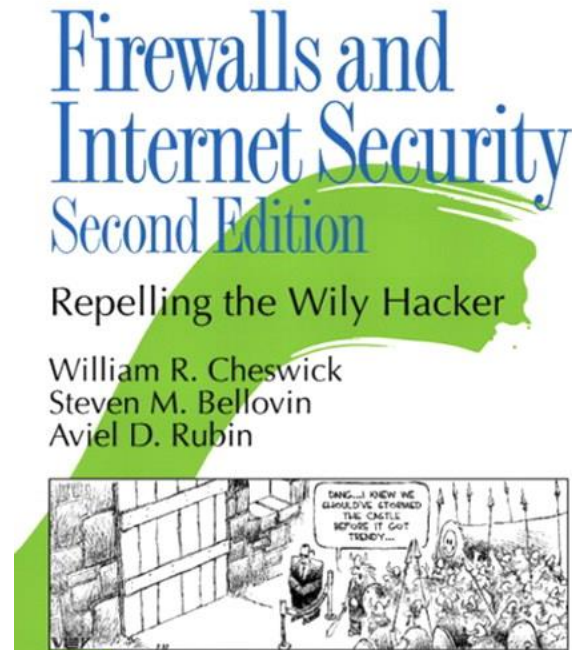
Web traffic scanning

- Intercept and proxy web traffic
 - Can be host-based
 - Usually at enterprise gateway
- Block known bad sites
- Block pages with known attacks
- Scan attachments
 - Usually traditional virus scanning methods

Firewall references

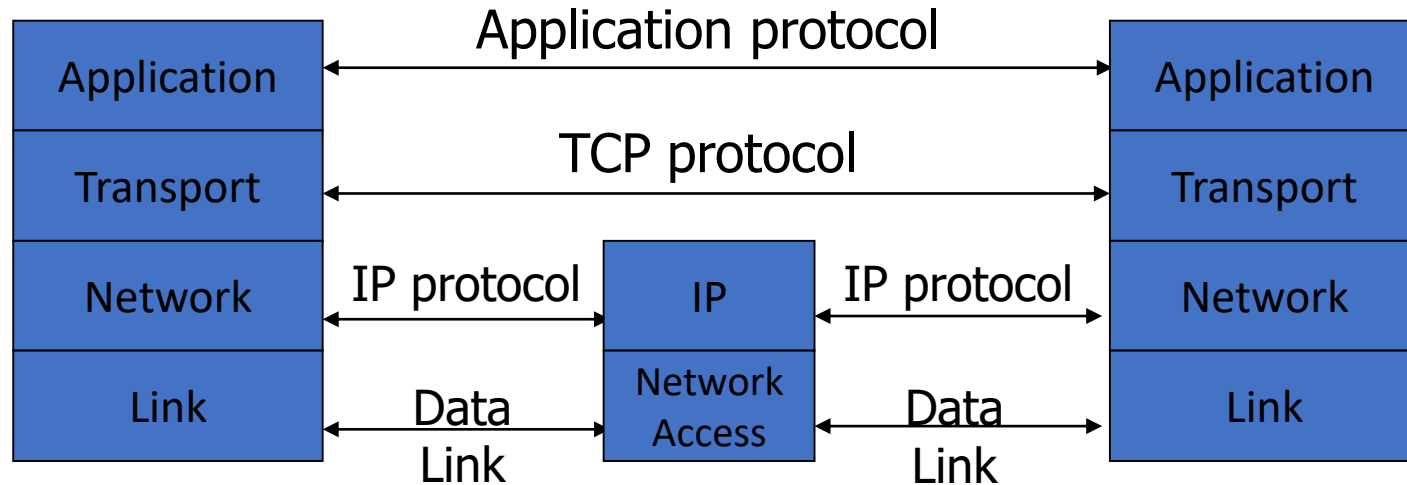


Elizabeth D. Zwicky
Simon Cooper
D. Brent Chapman



William R Cheswick
Steven M Bellovin
Aviel D Rubin

TCP Protocol Stack



- Intrusion detection
- Infrastructure protocols
 - BGP
 - DNS

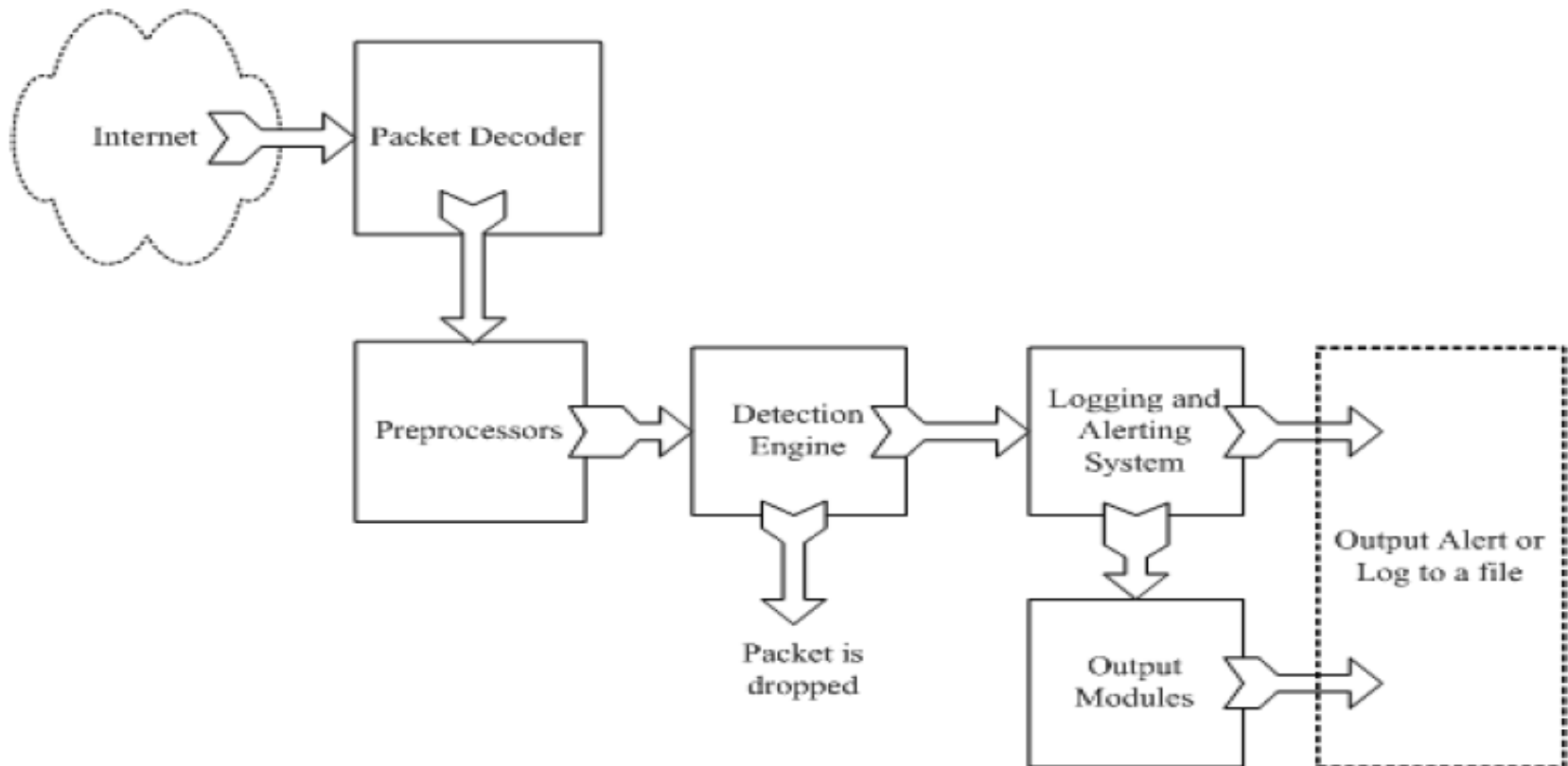
Intrusion detection

- Many intrusion detection systems
 - Close to 100 systems with current web pages
 - Network-based, host-based, or combination
- Two basic models
 - Misuse detection model
 - Maintain data on known attacks
 - Look for activity with corresponding signatures
 - Anomaly detection model
 - Try to figure out what is “normal”
 - Report anomalous behavior
- Fundamental problem: too many false alarms



<http://www.snort.org/>

Example: Snort

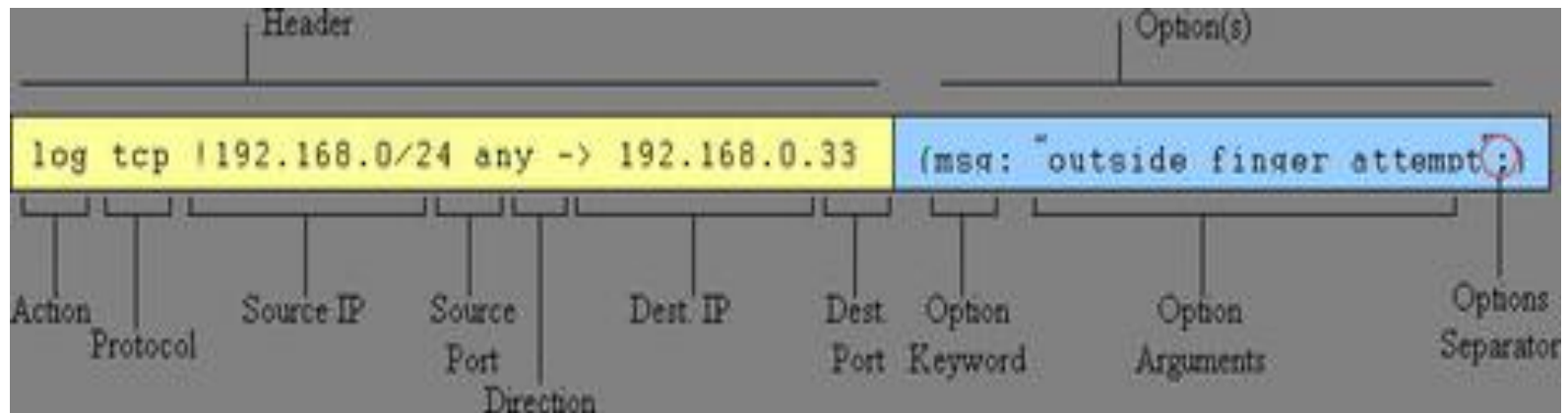
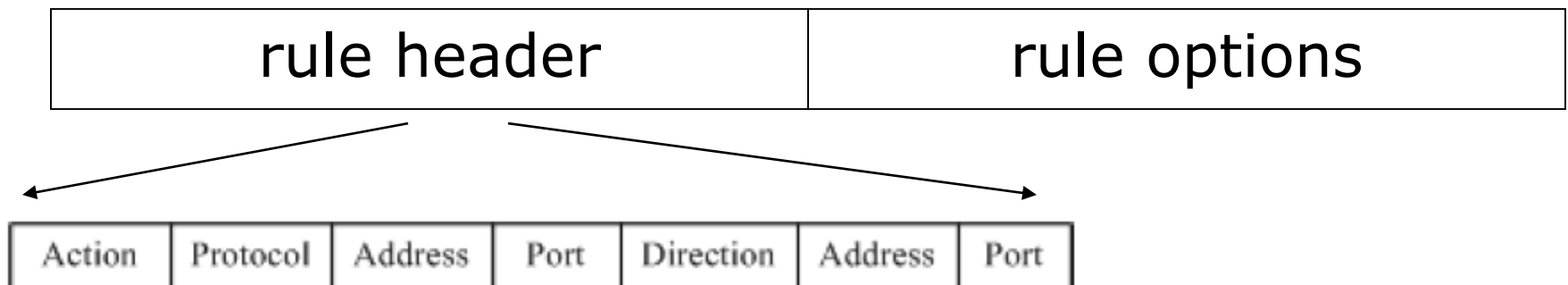


From: Rafeeq Ur Rehman, *Intrusion Detection Systems with Snort: Advanced IDS Techniques with Snort, Apache, MySQL, PHP, and ACID.*

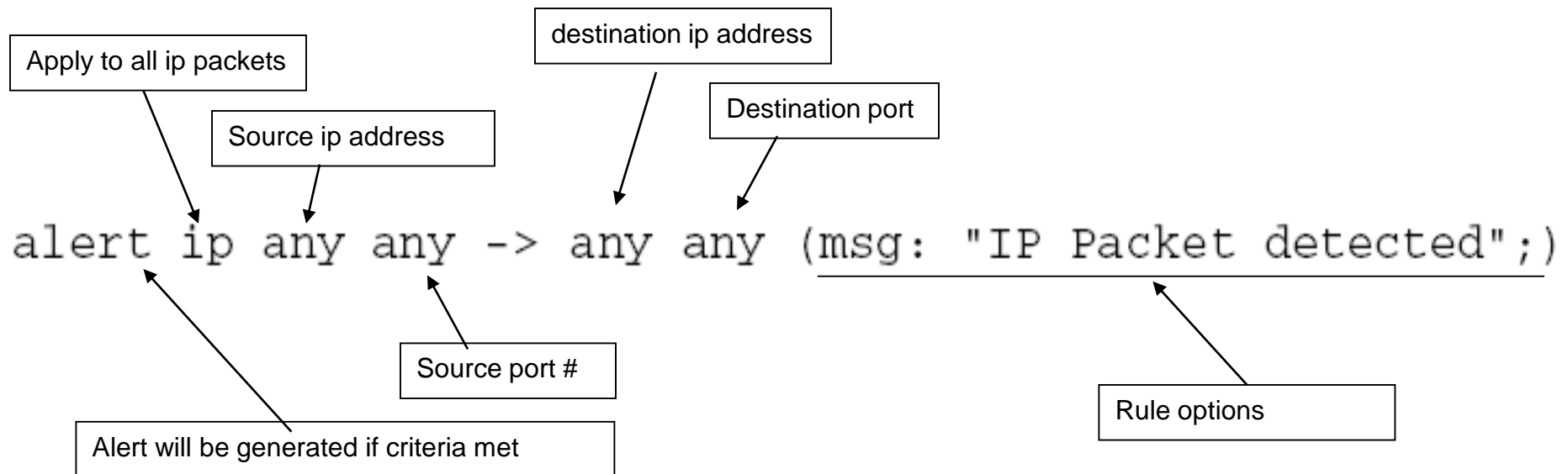
Snort components

- Packet Decoder
 - input from Ethernet, SLIP, PPP...
- Preprocessor:
 - detect anomalies in packet headers
 - packet defragmentation
 - decode HTTP URI
 - reassemble TCP streams
- Detection Engine: applies rules to packets
- Logging and Alerting System
- Output Modules: alerts, log, other output

Snort detection rules



Additional examples



```
alert tcp $TELNET_SERVERS 23 -> $EXTERNAL_NET any (msg:"TELNET
  Attempted SU from wrong group"; flow:
from_server,established; content:"to su root"; nocase;
  classtype:attempted-admin; sid:715; rev:6;)
```


Snort challenges

- Misuse detection – avoid known intrusions
 - Database size continues to grow
 - Snort version 2.3.2 had 2,600 rules
 - Snort spends 80% of time doing string match
- Anomaly detection – identify new attacks
 - Probability of detection is low

Difficulties in anomaly detection

- Lack of training data
 - Lots of “normal” network, system call data
 - Little data containing realistic attacks, anomalies
- Data drift
 - Statistical methods detect changes in behavior
 - Attacker can attack gradually and incrementally
- Main characteristics not well understood
 - By many measures, attack may be within bounds of “normal” range of activities
- False identifications are very costly
 - Sys Admin spend many hours examining evidence

Summary

- Network protocol security
 - Wireless security – 802.11i/WPA2
 - IPSEC
 - BGP instability and S-BGP
 - DNSSEC, DNS rebinding
- Standard network perimeter defenses
 - Firewall
 - Packet filter (stateless, stateful), Application layer proxies
 - Traffic shaping
 - Intrusion detection
 - Anomaly and misuse detection

Module 10: DDoS

Denial of Service

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

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

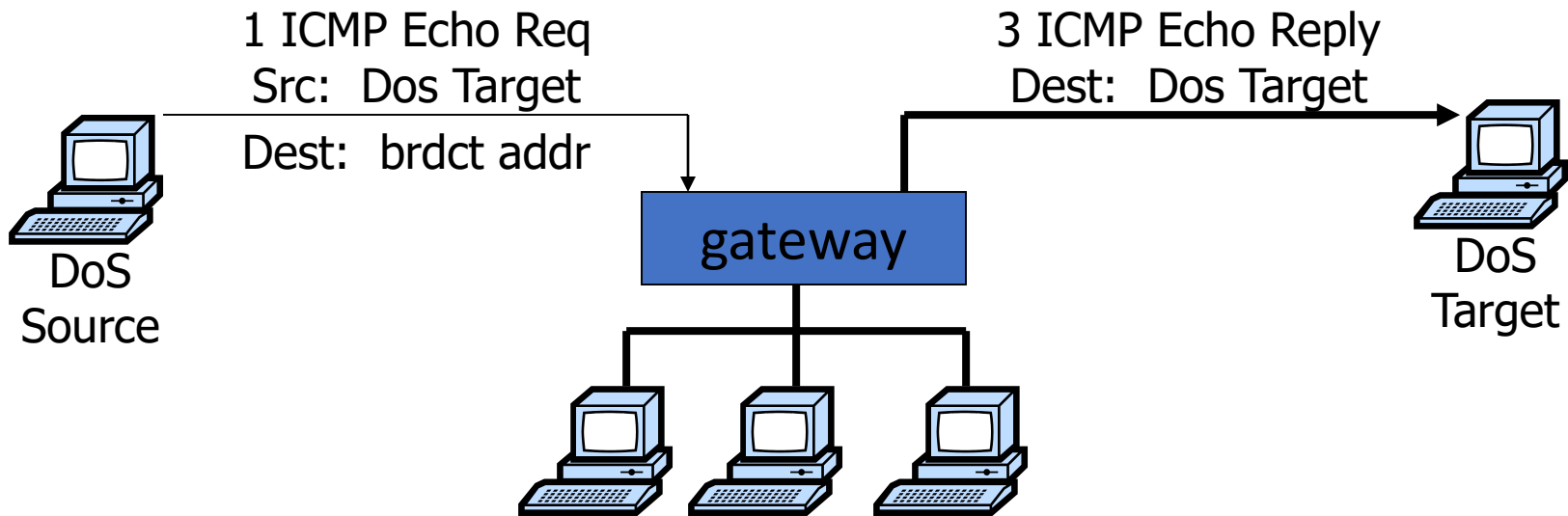
Warm up: 802.11b DoS bugs

- 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

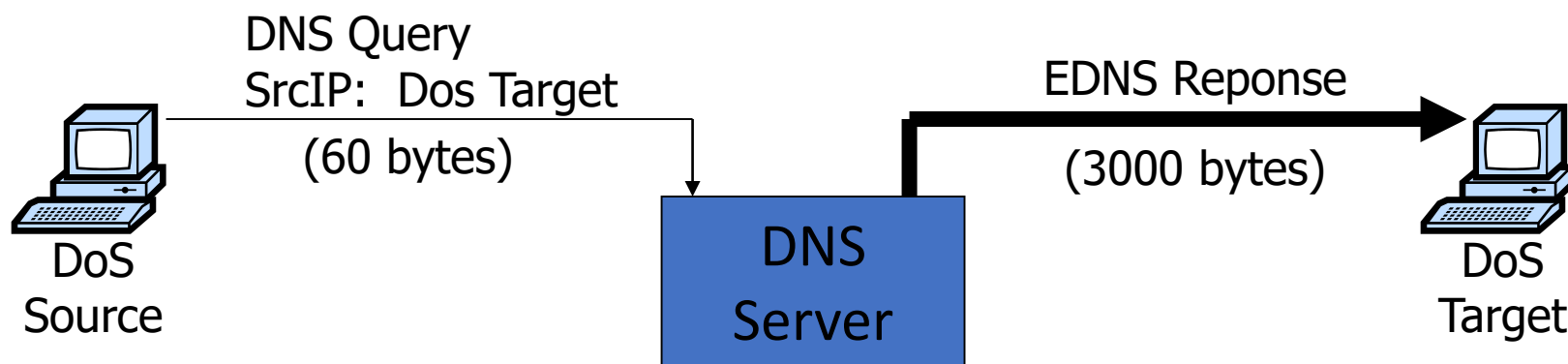
Smurf amplification DoS attack



- 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

Modern day example (Mar '13)

DNS Amplification attack: ($\times 50$ amplification)



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

2014: 28M open resolvers (openresolverproject.org)

81 \Rightarrow 3/2013: DDoS attack generating 309 Gbps for 28 mins.

Scale, Targeting and Frequency of Attacks

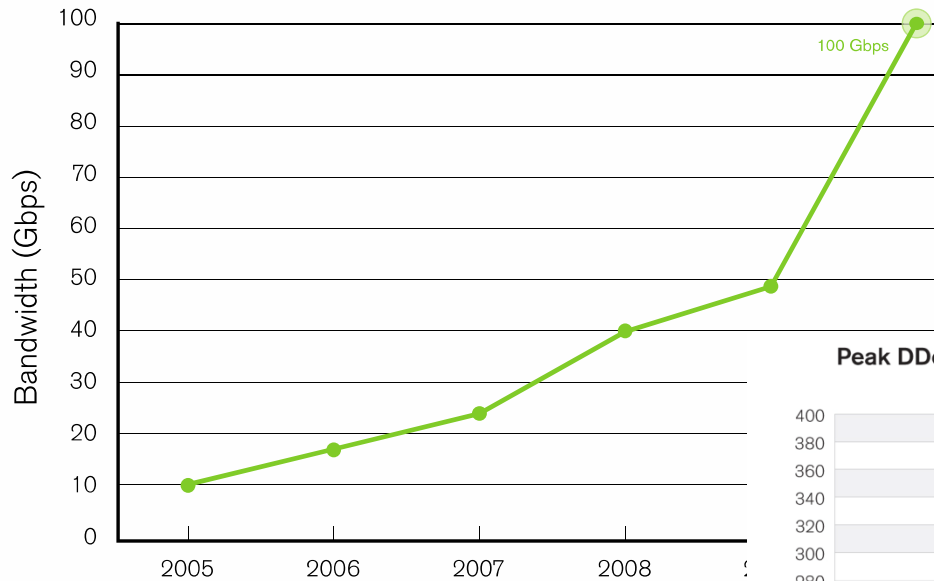
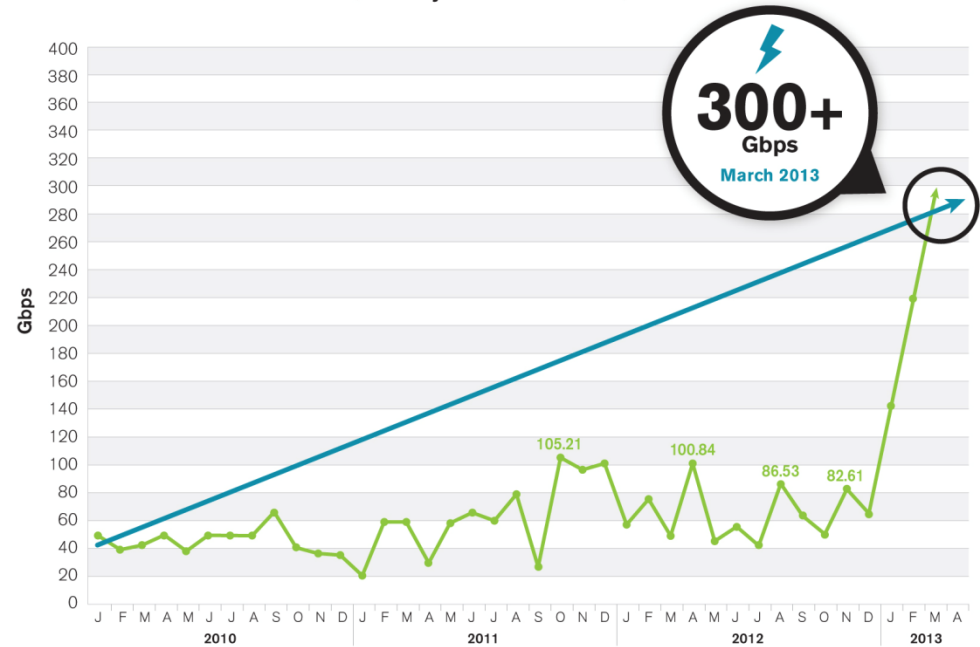


Figure 13

Source: Arbor Networks, Inc.

Peak DDoS Attack Size (January 2010 to Present)



Source: Arbor Networks, Inc.

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

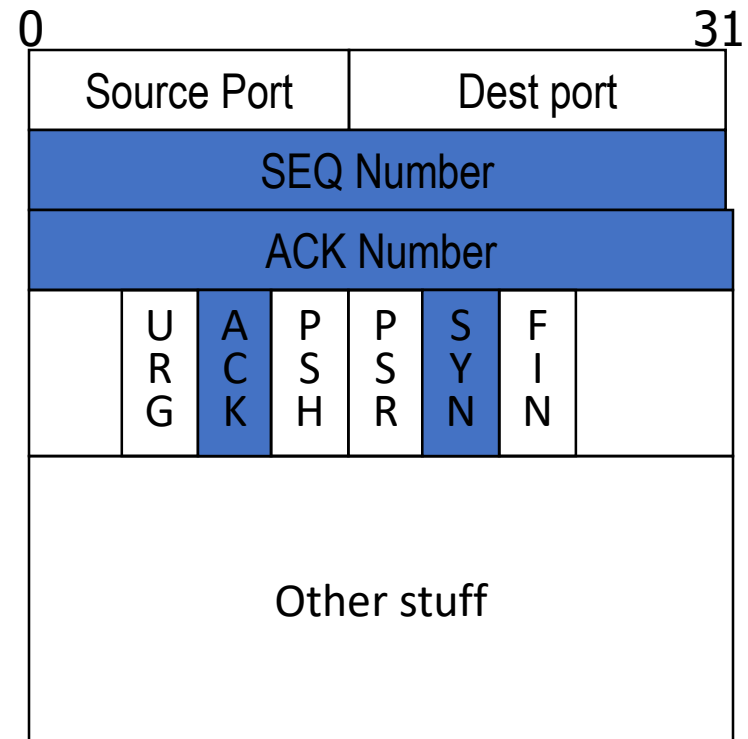
Review: IP Header format

- Connectionless
 - Unreliable
 - Best effort

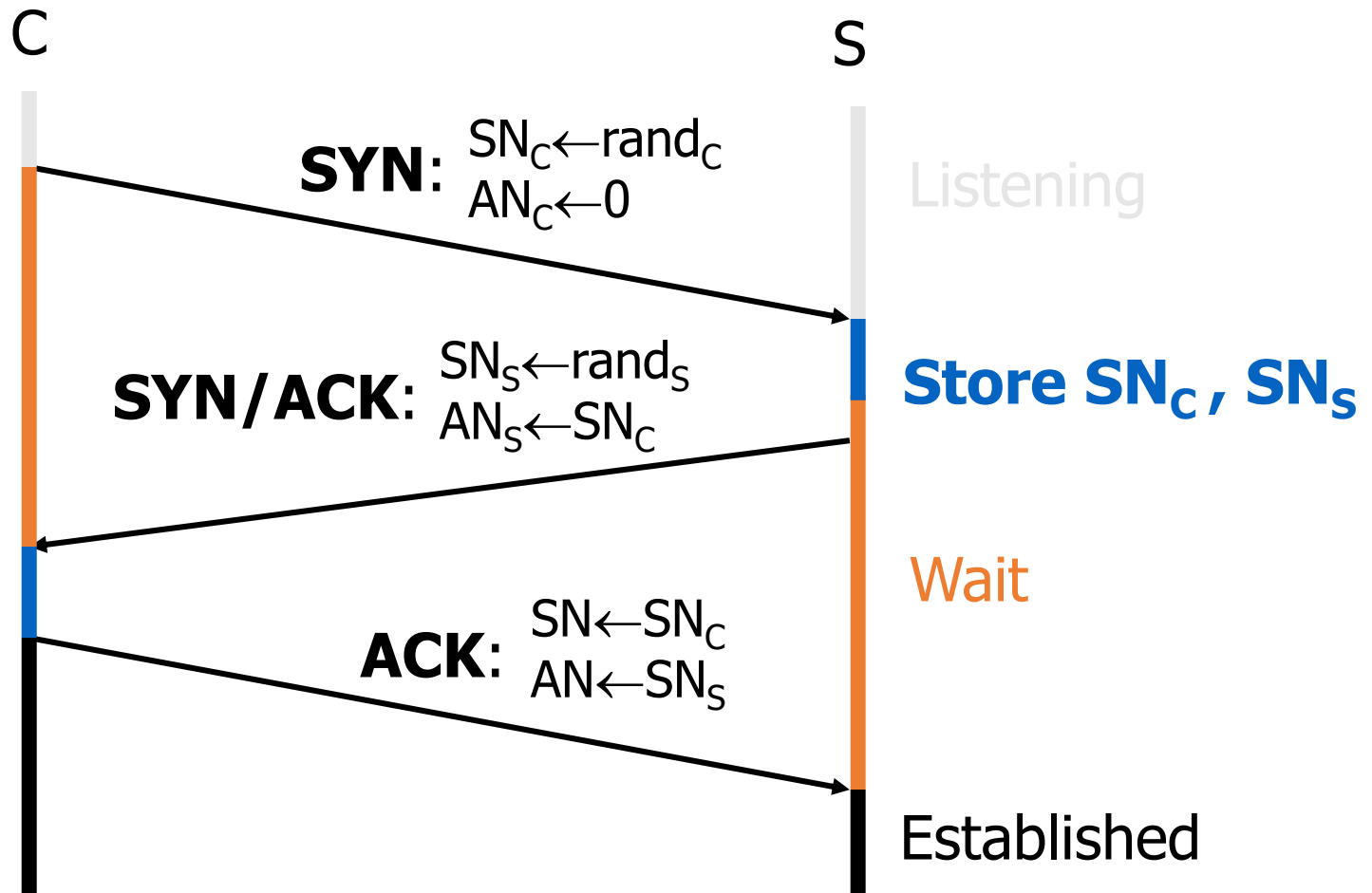
0	31
Version	Header Length
Type of Service	
Total Length	
Identification	
Flags	Fragment Offset
Time to Live	
Protocol	
Header Checksum	
Source Address of Originating Host	
Destination Address of Target Host	
Options	
Padding	
IP Data	

Review: TCP Header format

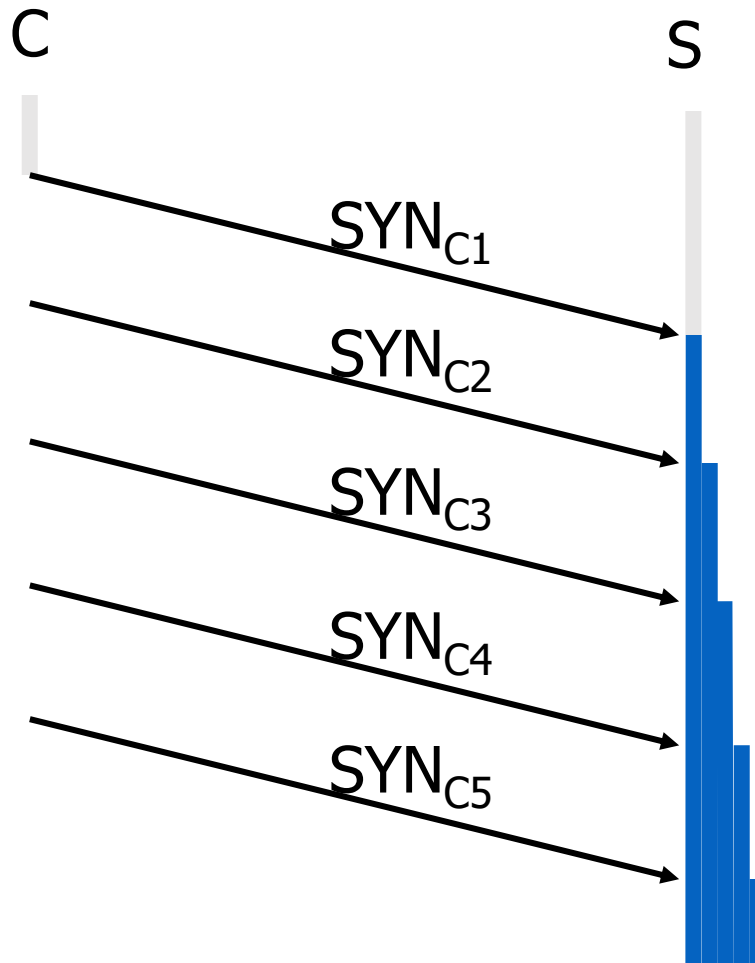
- TCP:
 - Session based
 - Congestion control
 - In order delivery



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

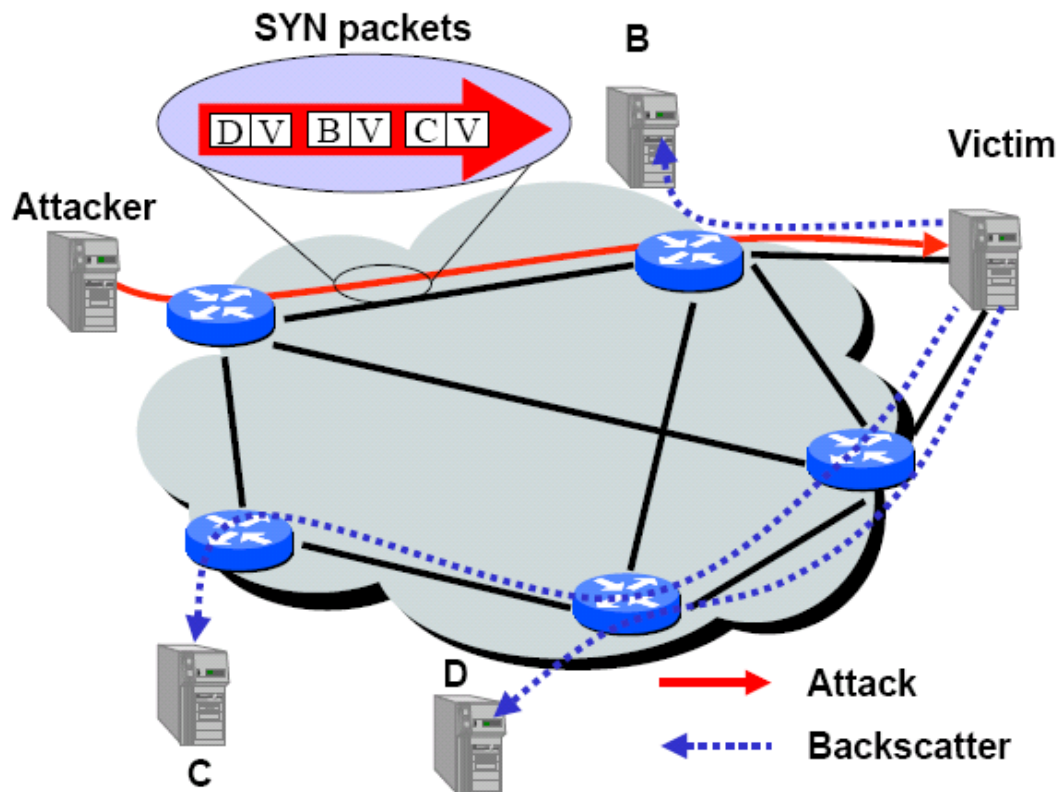
Syncookies

[Bernstein, Schenk]

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

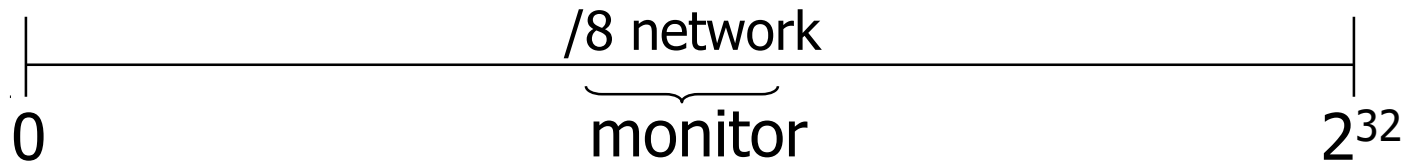
SYN floods: backscatter [MVS'01]

- 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

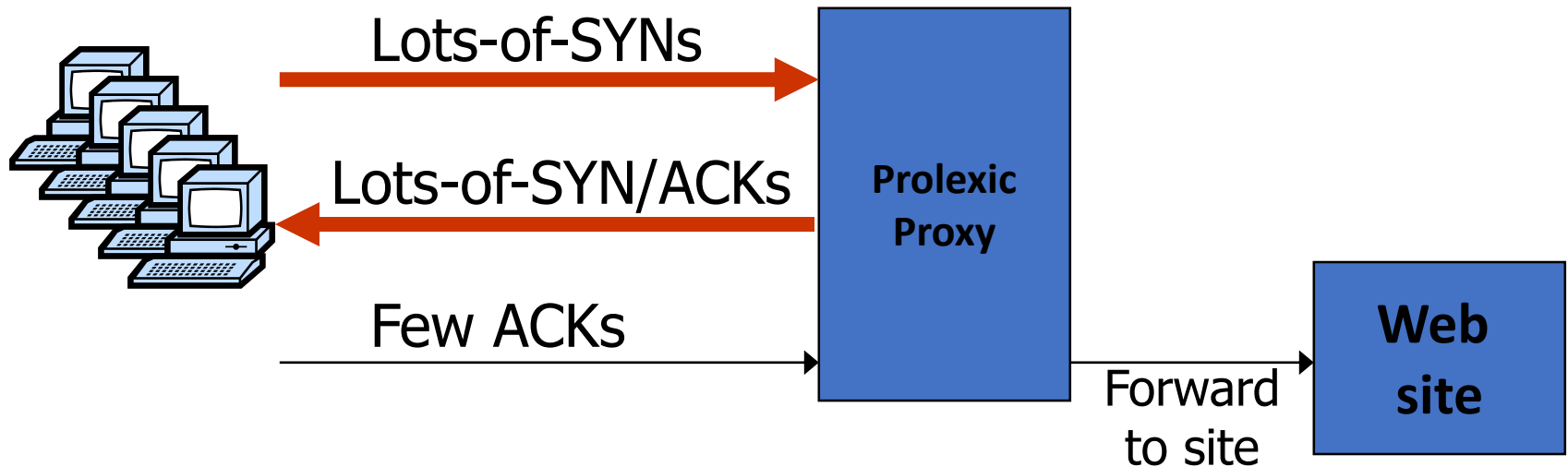
SYN Floods II: Massive flood

(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 \Rightarrow
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

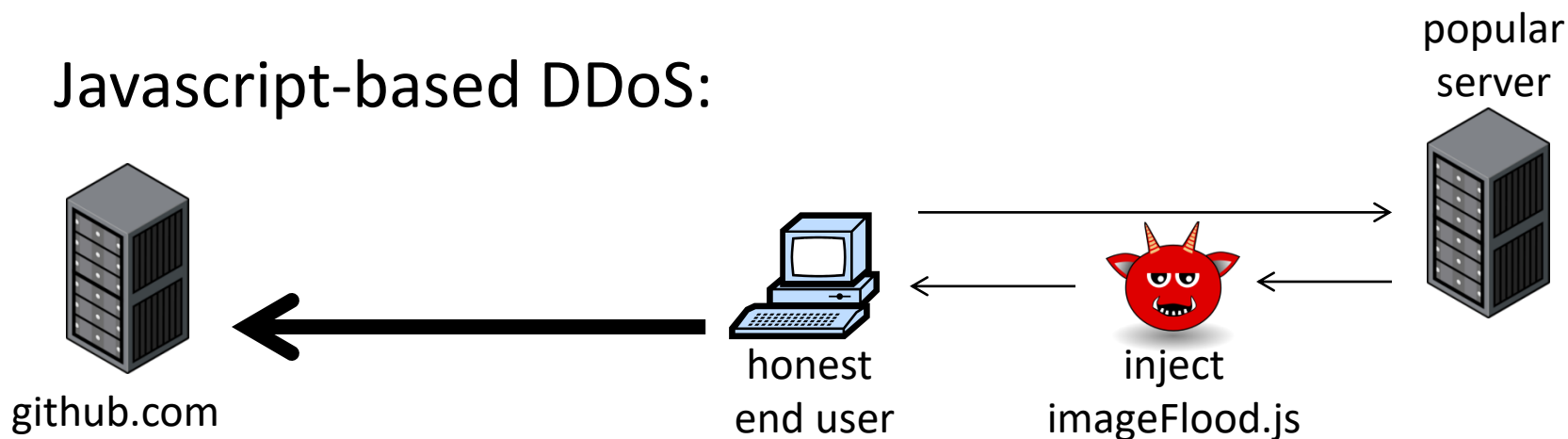
Proxy must keep floods of these away from web site

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?

DoS via route hijacking

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

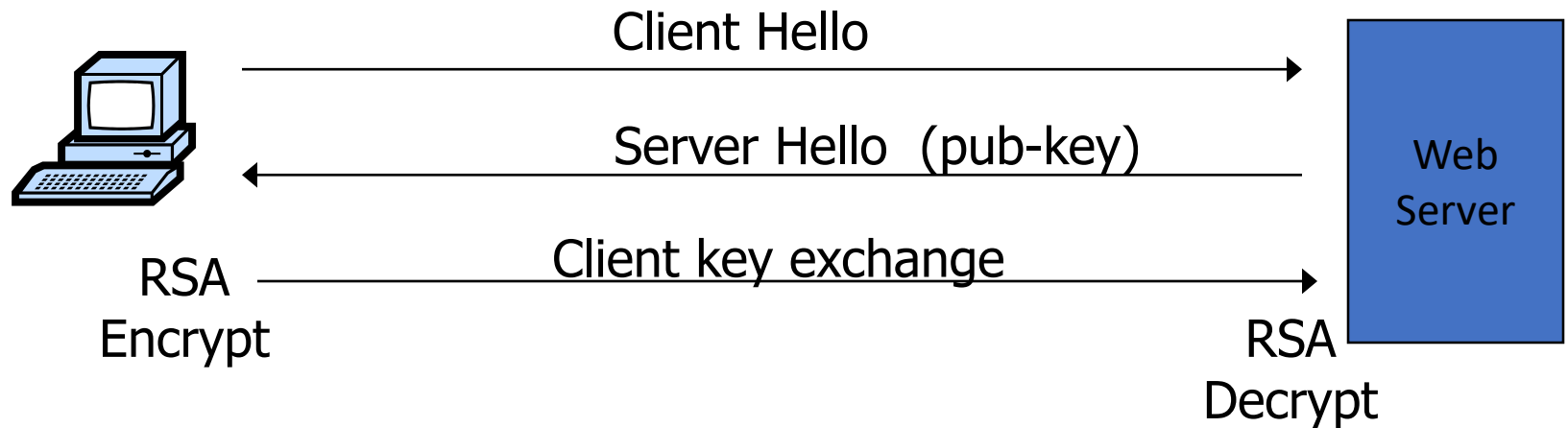
youtube.com is 208.65.153.238, ...

- 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 $\approx 10\times$ 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
$$\text{LSB}_n \left(\text{SHA-1}(C \parallel X) \right) = 0^n$$
 - 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: $C = \text{TCP server-seq-num}$
 - 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
 - ⇒ 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.

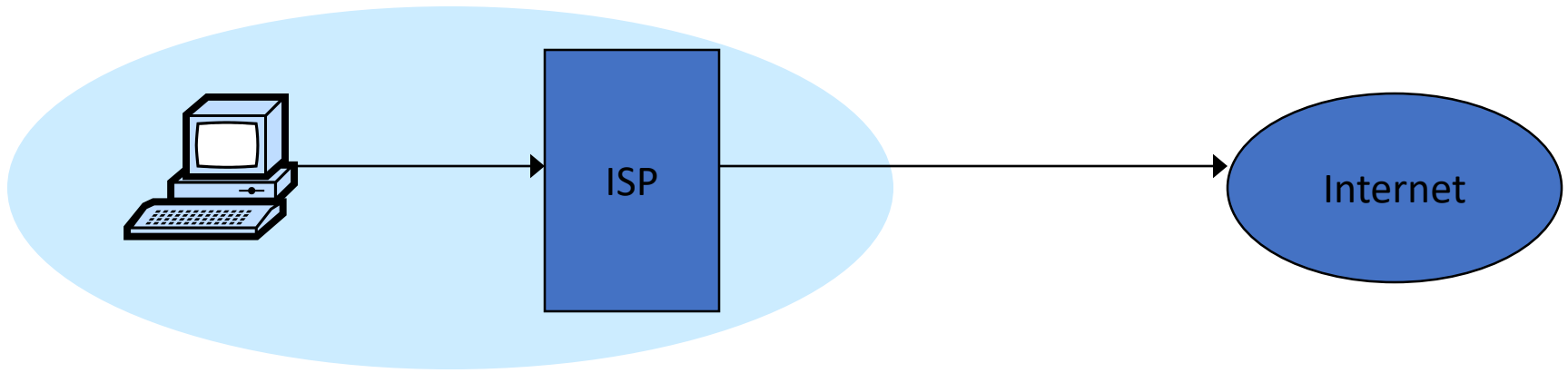
3. Source identification

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

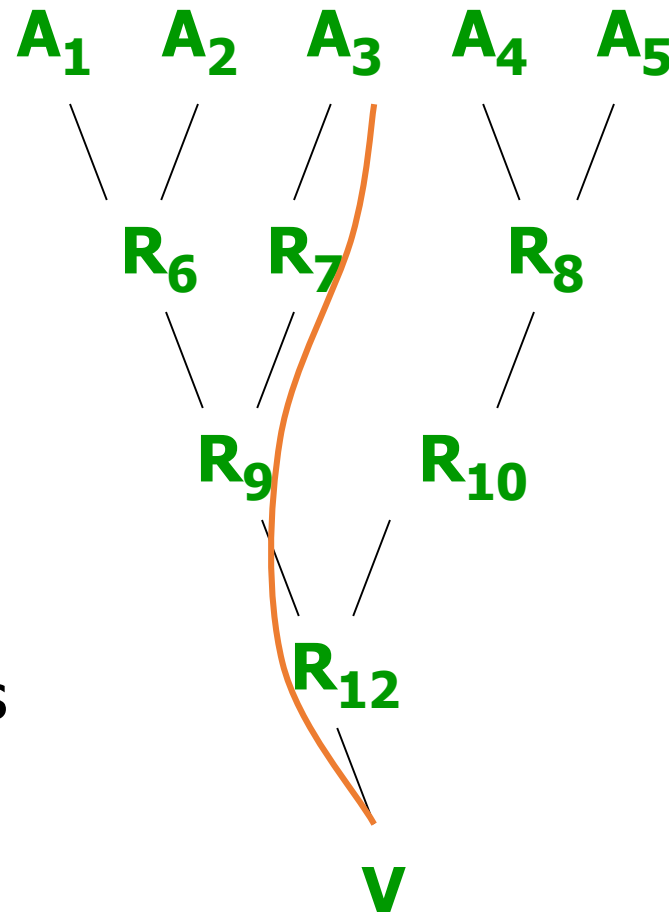
- Write path into network packet
 - Each router adds its own IP address to packet
 - Victim reads path from packet

◆ 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

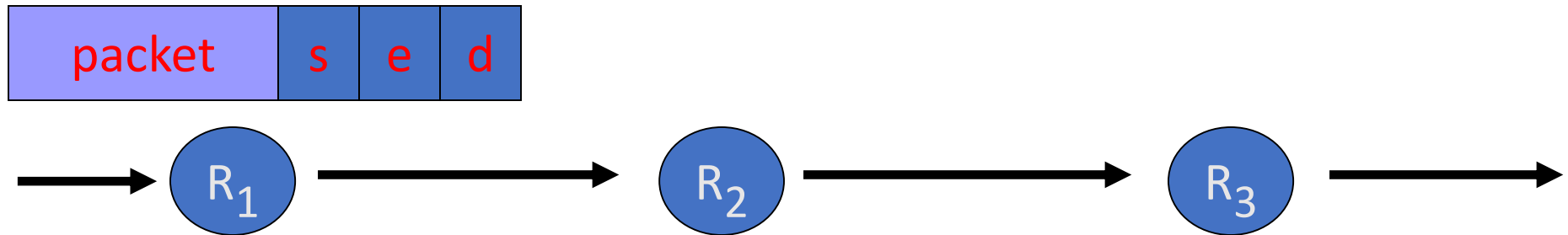


Edge Sampling

- 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

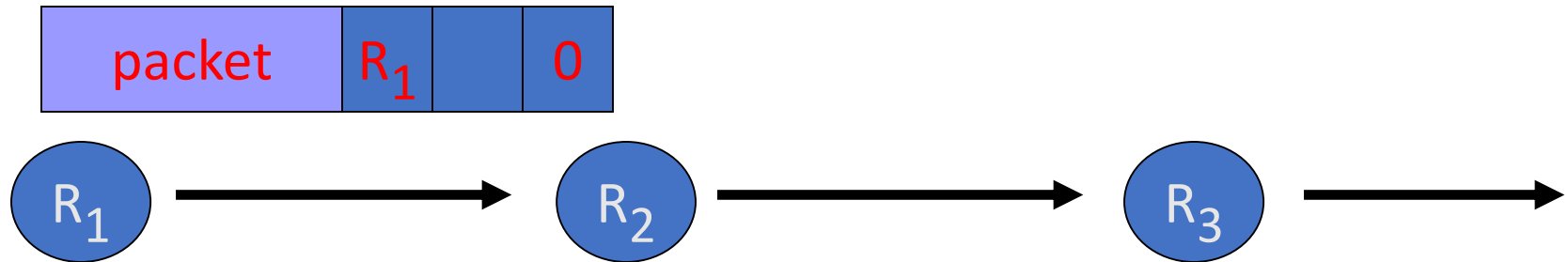
Edge Sampling: picture

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



Edge Sampling: picture

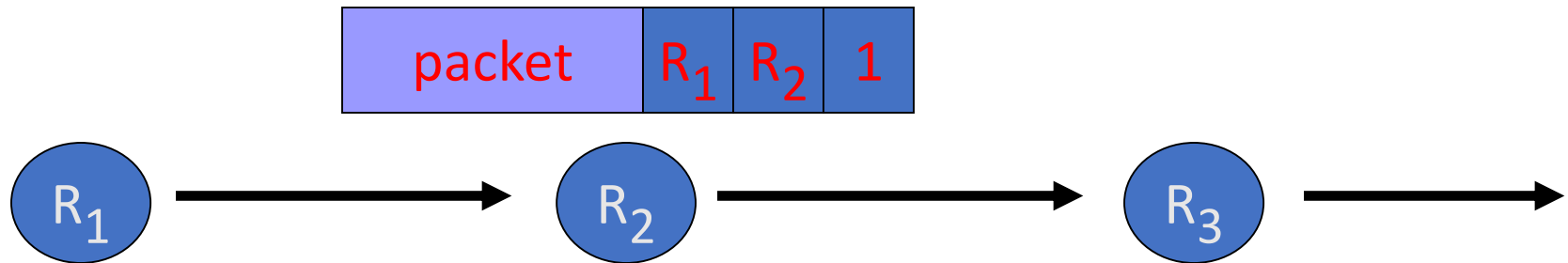
- Begin writing edge
 - R_1 chooses to write start of edge
 - Sets distance to 0



Edge Sampling

◆ Finish writing edge

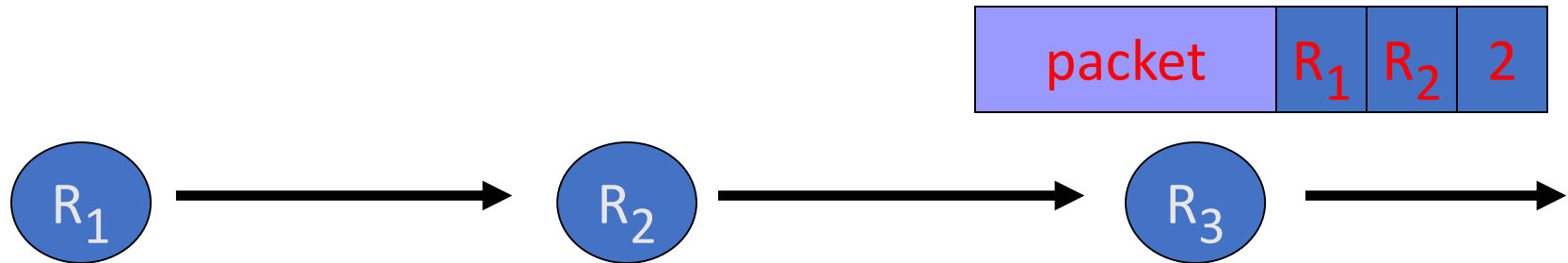
- R_2 chooses not to overwrite edge
- Distance is 0
 - ◆ Write end of edge, increment distance to 1



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

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

where p is marking probability, d is length of path

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:**

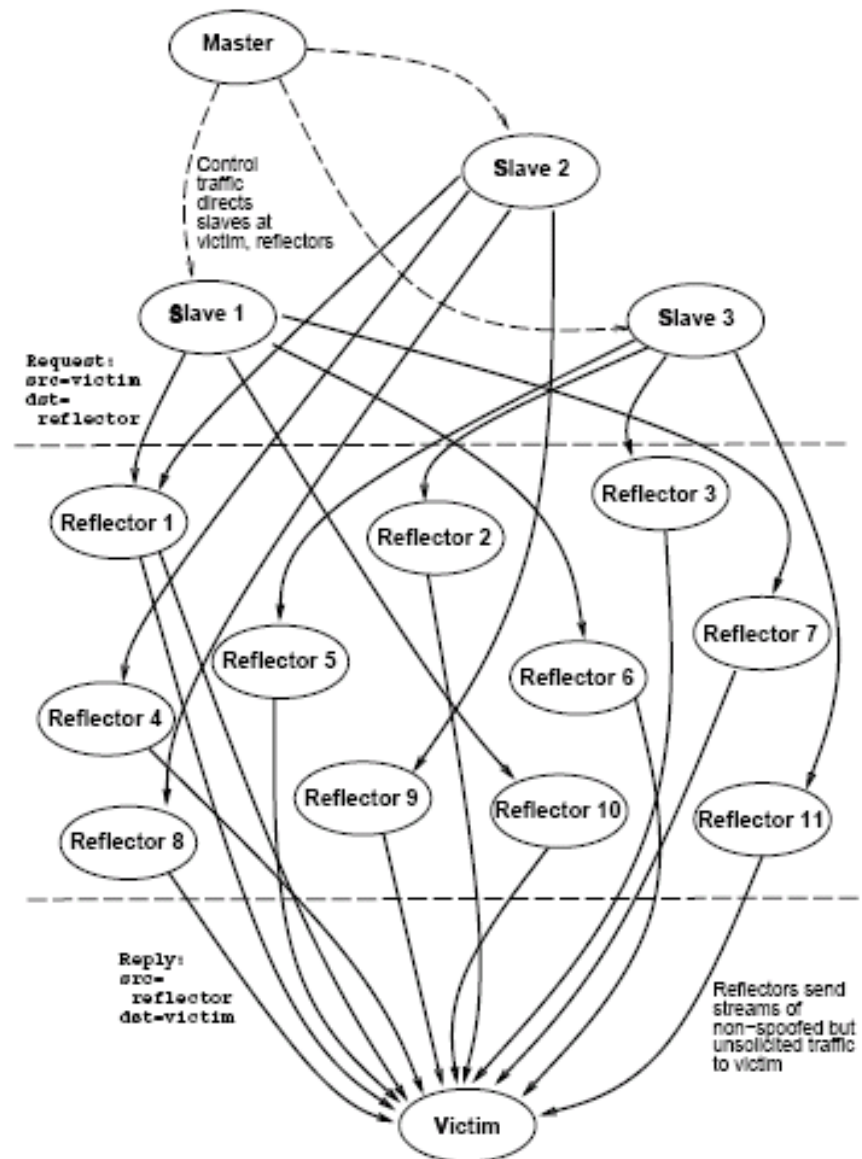
- A network component that responds to packets
- Response sent to victim (spoofed source IP)

- 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

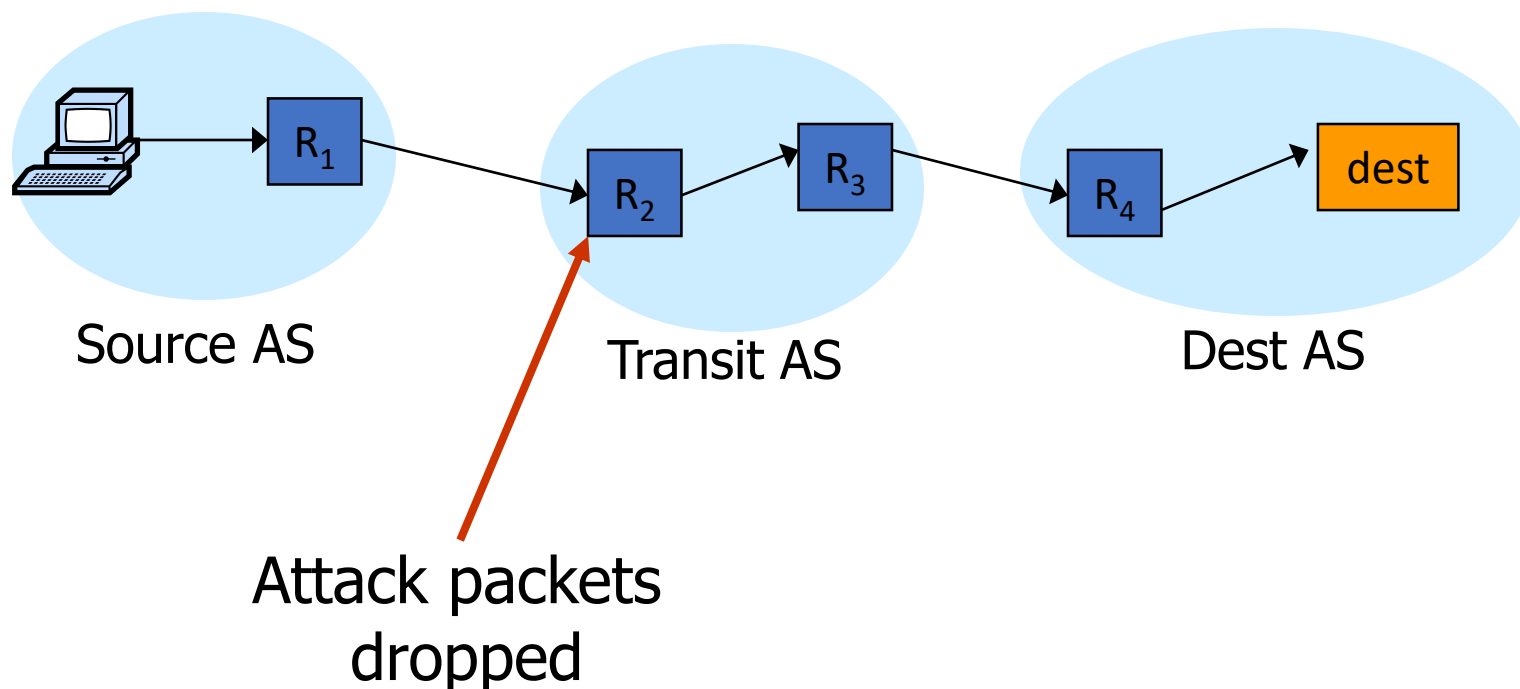
- Anderson, Roscoe, Wetherall.
 - Preventing internet denial-of-service with capabilities. SIGCOMM '04.
- Yaar, Perrig, and Song.
 - Siff: A stateless internet flow filter to mitigate DDoS flooding attacks. IEEE S&P '04.
- Yang, Wetherall, Anderson.
 - A DoS-limiting network architecture. SIGCOMM '05

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



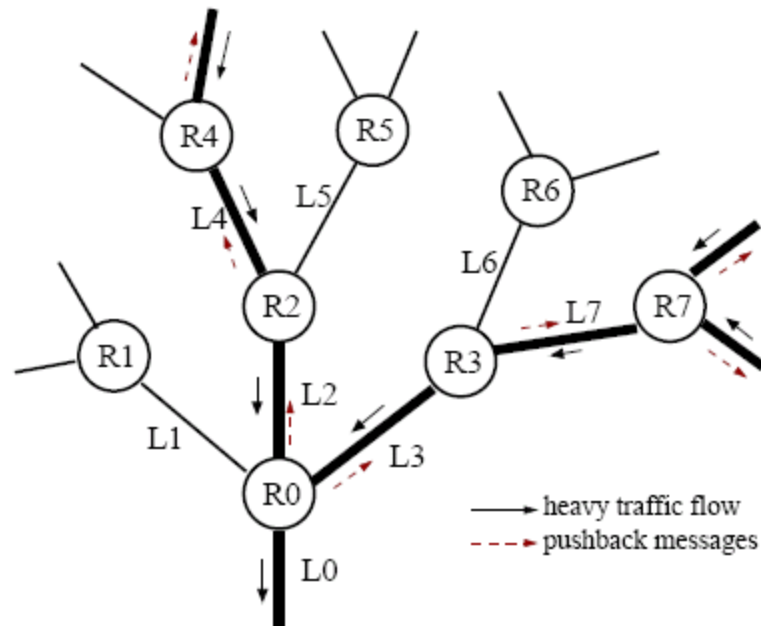
Pushback Traffic Filtering

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

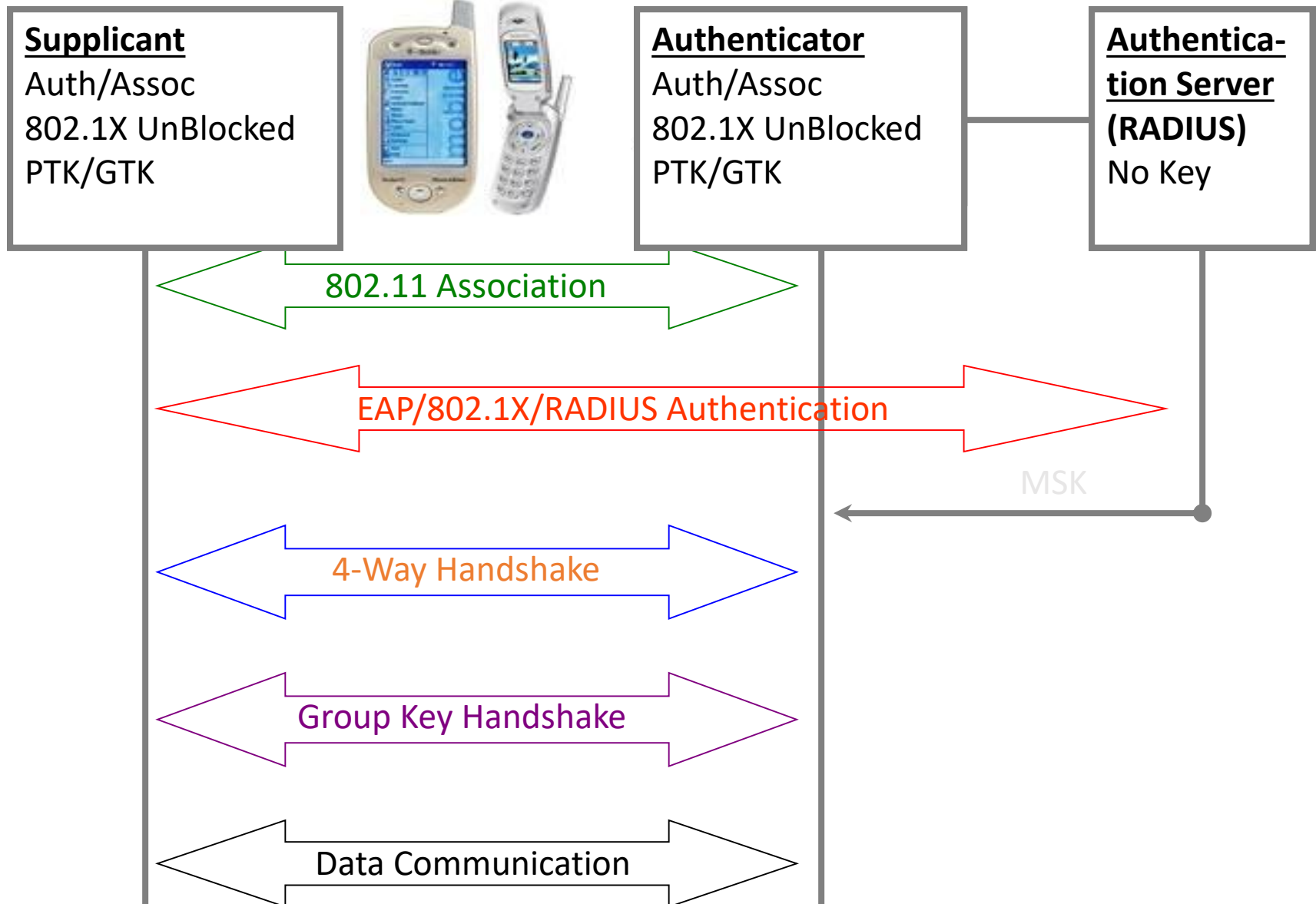
- Keromytis, Misra, Rubenstein.
SOS: Secure Overlay Services. SIGCOMM '02.
- D. Andersen. Mayday.
Distributed Filtering for Internet Services.
Usenix USITS '03.
- Lakshminarayanan, Adkins, Perrig, Stoica.
Taming IP Packet Flooding Attacks. HotNets '03.

Take home message:

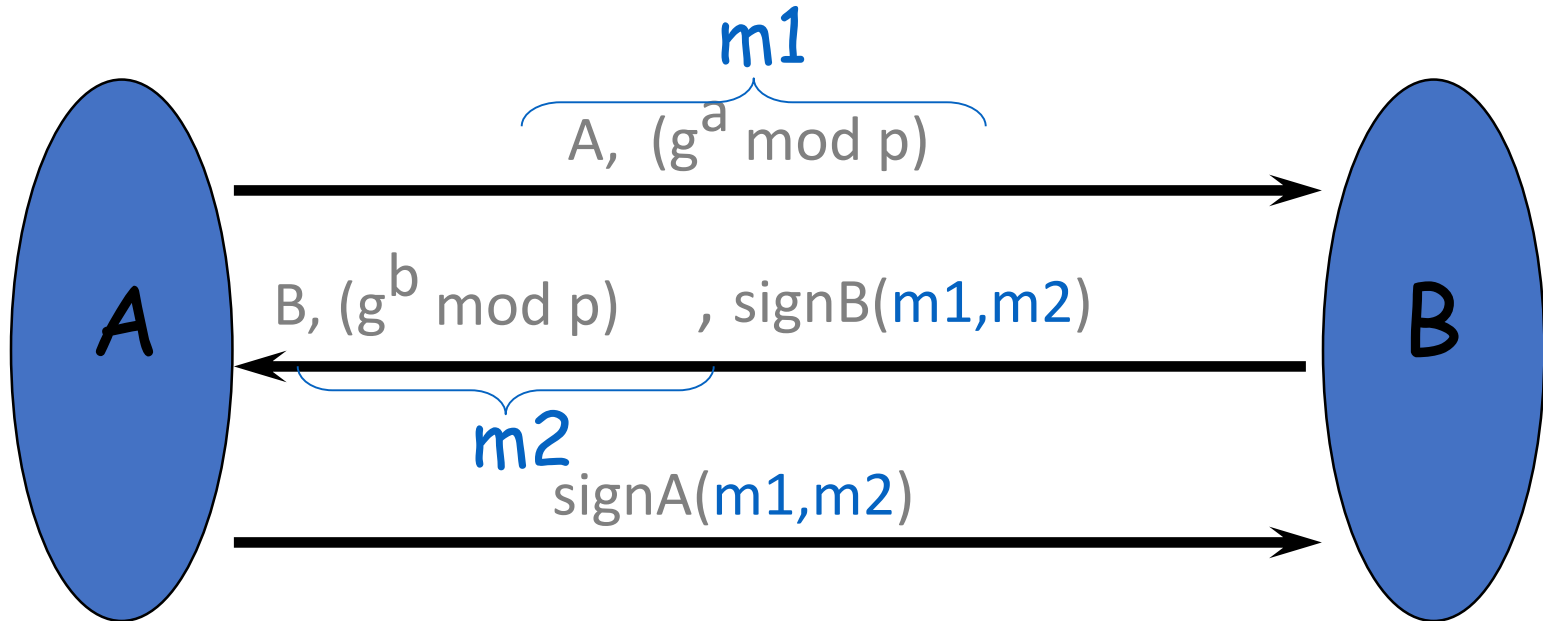
- Denial of Service attacks are real.
Must be considered at design time.
- Sad truth:
 - Internet is ill-equipped to handle DDoS attacks
 - Commercial solutions: CloudFlare, Prolexic
- Many good proposals for core redesign.

Link-layer connectivity

802.11i Protocol



IKE subprotocol from IPSEC



Result: A and B share secret $g^{ab} \bmod p$