



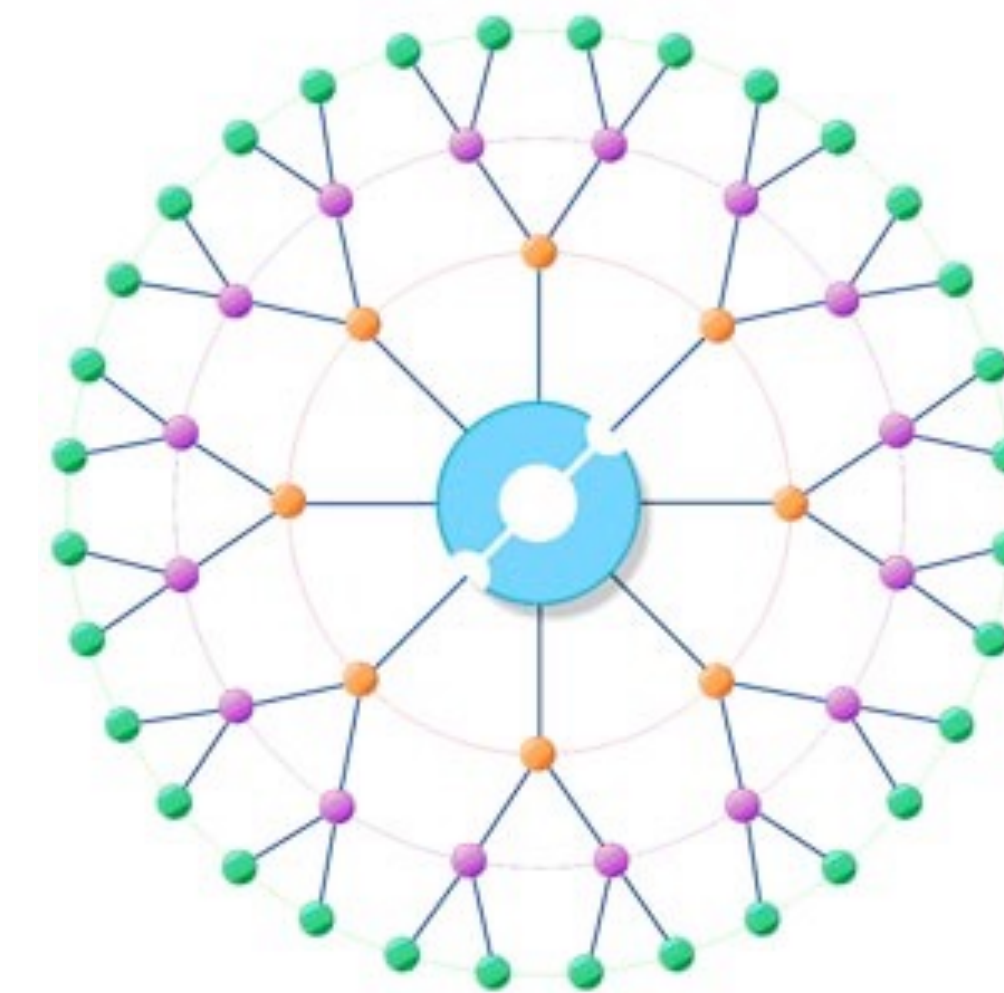
PennState

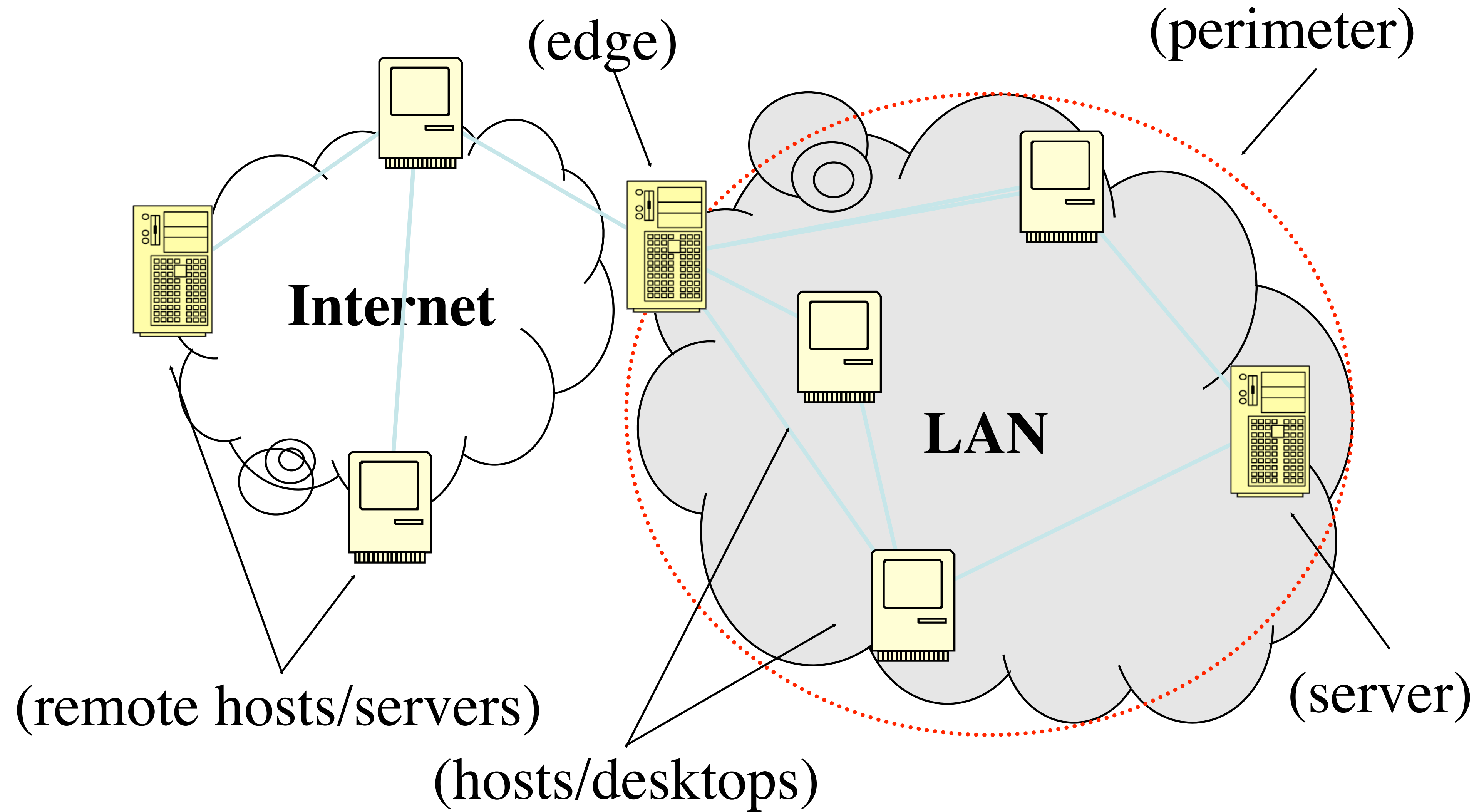
CSE 543: Computer Security

Module: Network Security

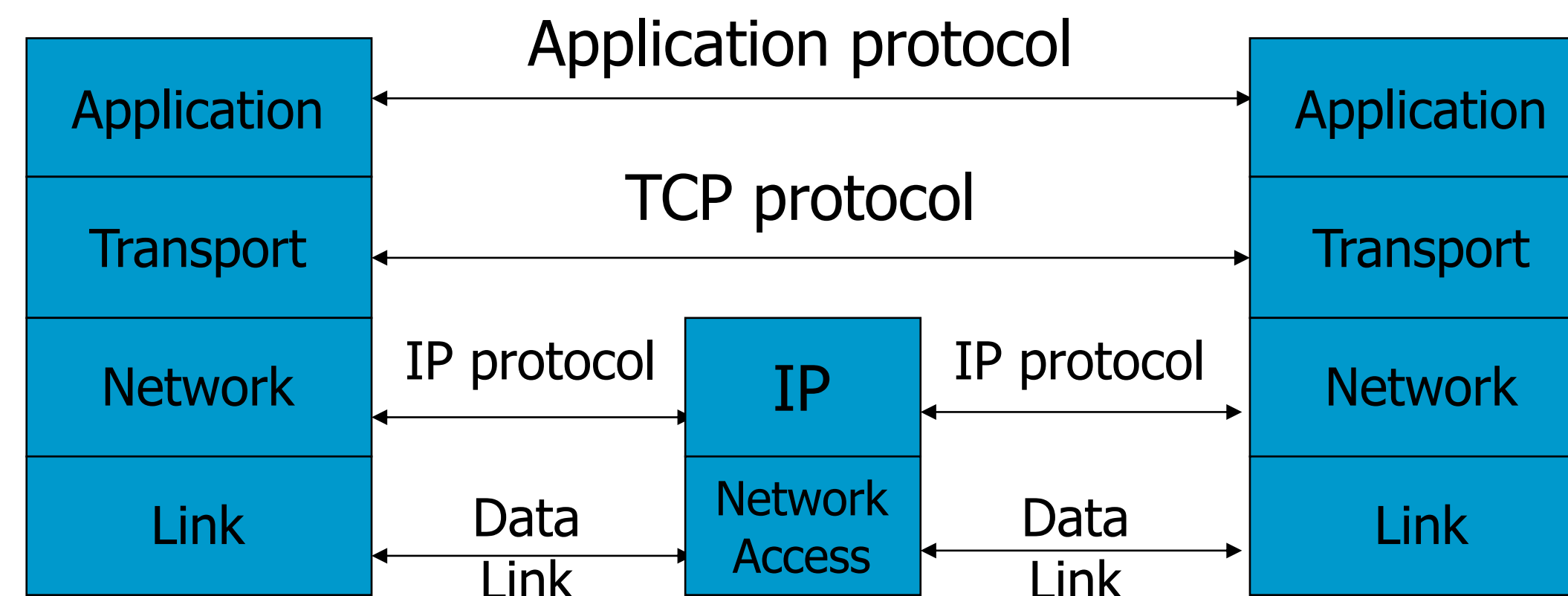
Syed Rafiul Hussain

- Fundamentally about transmitting information between two devices
- Direct communication is now possible between any two devices anywhere (just about)
 - ▶ Lots of abstraction involved
 - ▶ Lots of network components
 - ▶ Standard protocols
 - ▶ Wired and wireless
 - ▶ Works in *protection* environment
- What about ensuring *security*?





- Internet Protocol (IP)
 - ▶ Really refers to a whole collection of protocols making up the vast majority of the Internet
- Routing
 - ▶ How these packets move from place to place
- Network management
 - ▶ Administrators have to maintain the services and infrastructure supporting everyone's daily activities



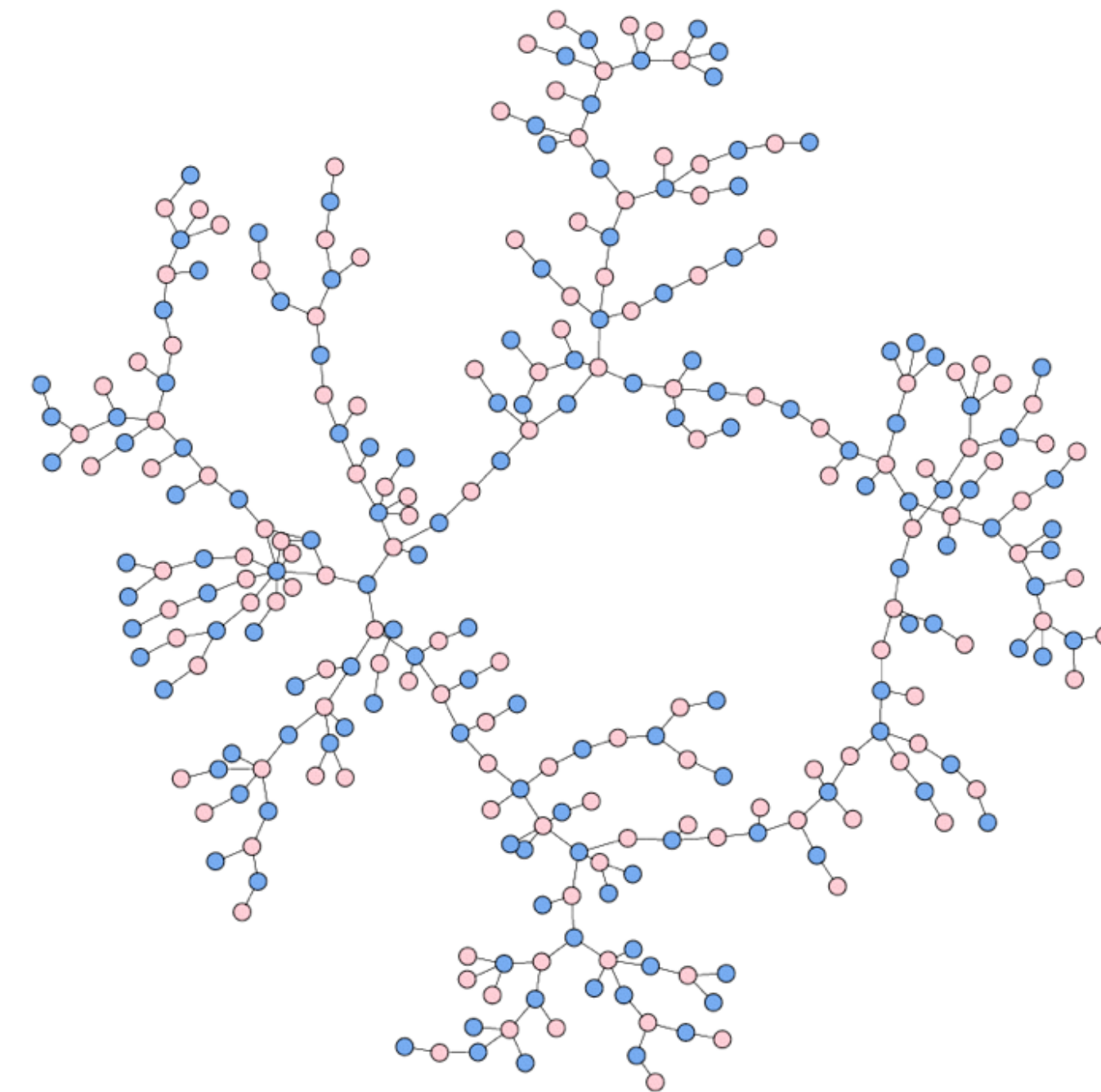
Network security: the high bits

- The network is ...
 - ▶ ... a collection of interconnected computers
 - ▶ ... with resources that must be protected
 - ▶ ... from unwanted inspection or modification
 - ▶ ... while maintaining adequate quality of service.
- Another way of seeing network security is ...
 - ▶ ... securing the networked **computers** such that the **integrity, confidentiality, and availability** of the resources is maintained.



Exploiting the network ...

- The Internet is extremely vulnerable to attack
 - ▶ it is a huge open system ...
 - ▶ which adheres to the *end-to-end* principle
 - smart end-points, dumb network



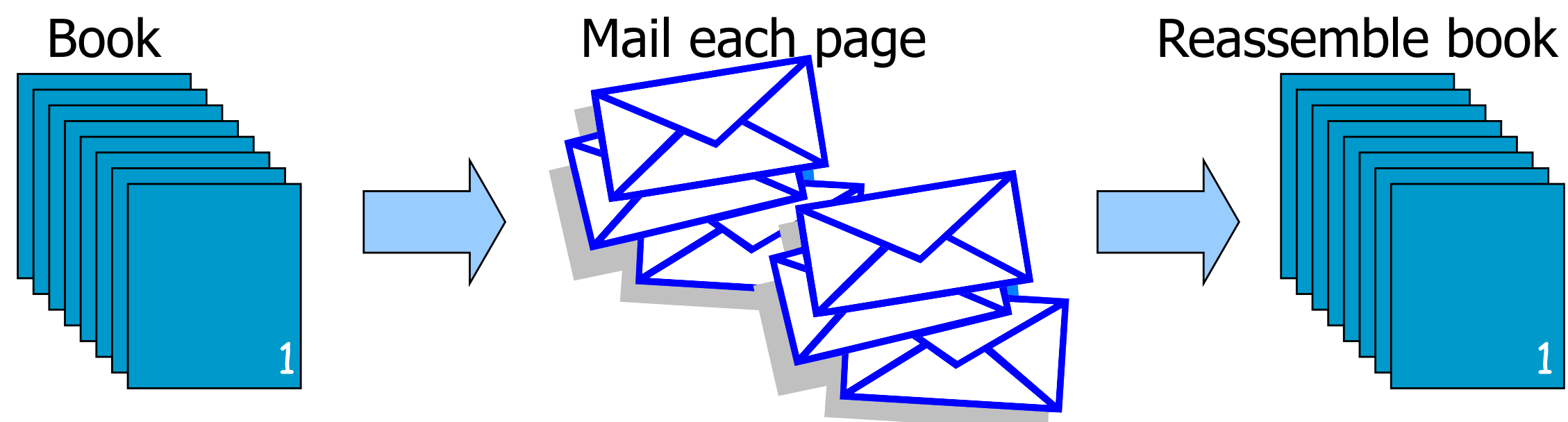
- Can you think of any *large-scale attacks* that would be enabled by this setup?

Types of Addresses in Internet

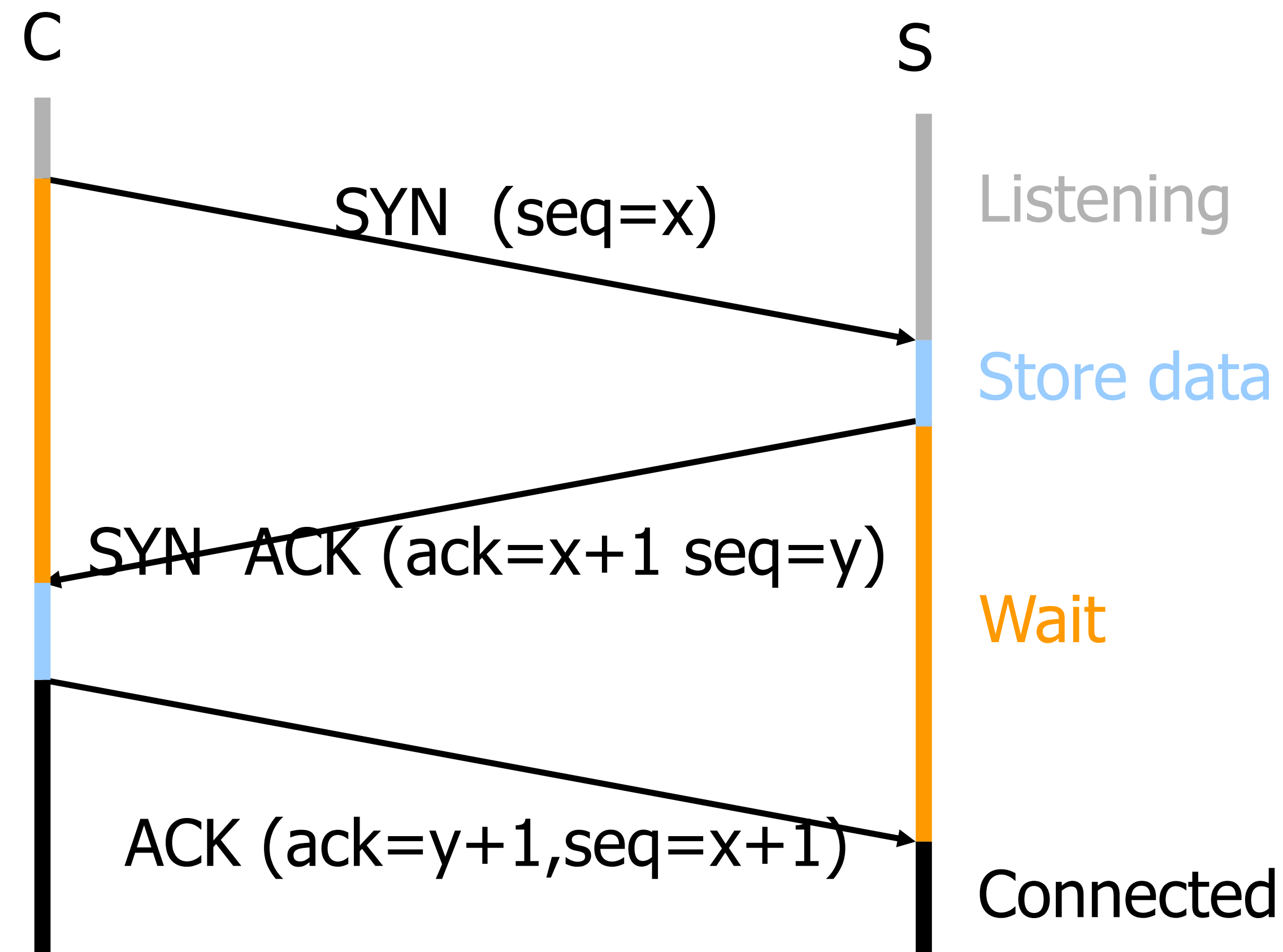
- Media Access Control (MAC) addresses in the network access layer
 - ▶ Associated w/ network interface card (NIC)
 - ▶ 48 bits or 64 bits
- IP addresses for the network layer
 - ▶ 32 bits for IPv4, and 128 bits for IPv6
 - ▶ E.g., 128.3.23.3
- IP addresses + ports for the transport layer
 - ▶ E.g., 128.3.23.3:80
- Domain names for the application/human layer
 - ▶ E.g., www.psu.edu

- Translation between IP addresses and MAC addresses
 - ▶ Address Resolution Protocol (ARP) for IPv4
 - ▶ Neighbor Discovery Protocol (NDP) for IPv6
- Routing with IP addresses
 - ▶ TCP, UDP, IP for routing packets, connections
 - ▶ Border Gateway Protocol for routing table updates
- Translation between IP addresses and domain names
 - ▶ Domain Name System (DNS)
 - ▶

- **Connection-oriented, preserves order**
 - ▶ **Sender**
 - Break data into packets
 - Attach sequence numbers
- **Receiver**
 - ▶ Acknowledge receipt; lost packets are resent
 - ▶ Reassemble packets in correct order



TCP Handshake

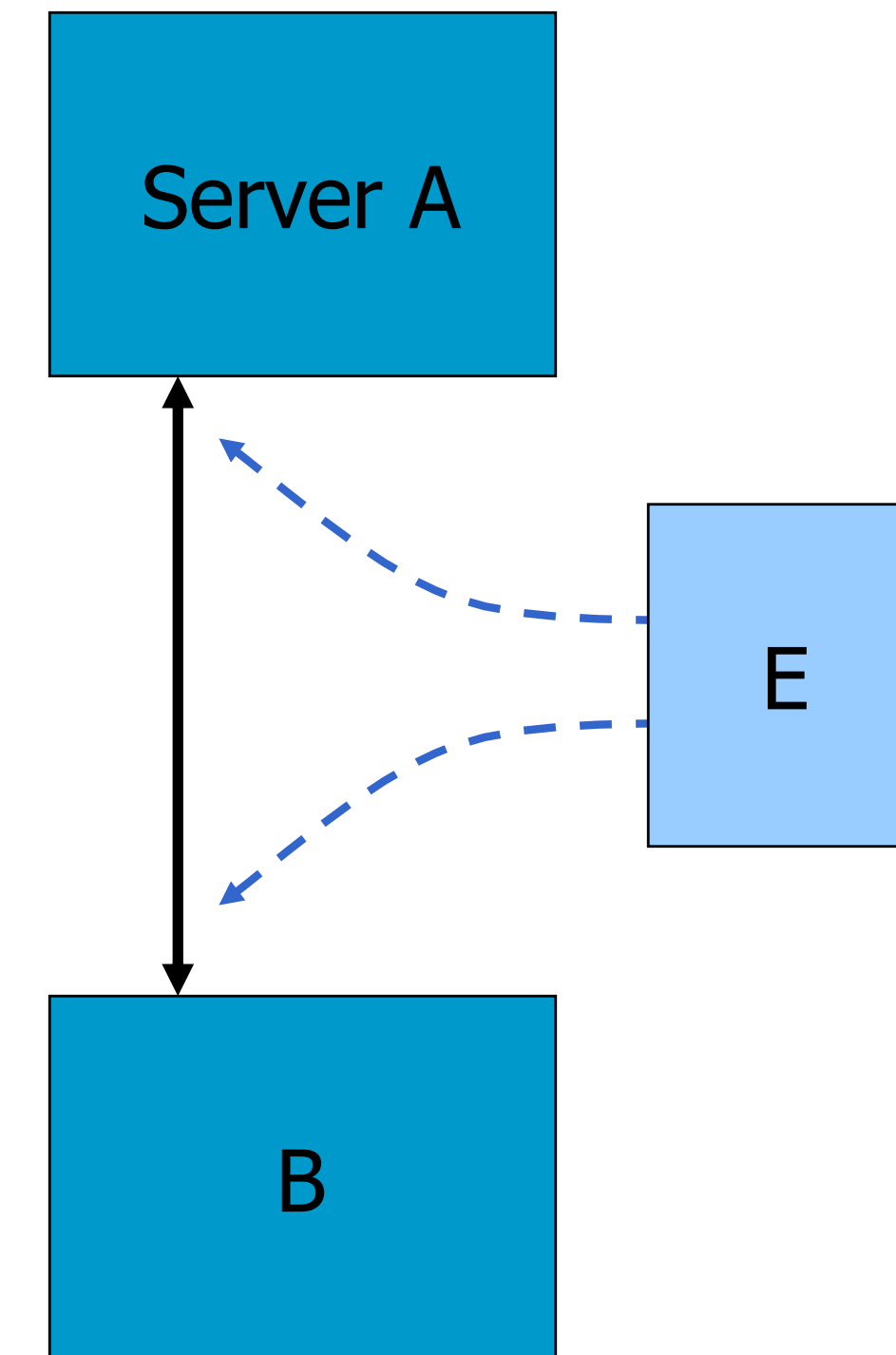


TCP Seq Prediction Attack

- Predict the sequence number used to identify the packets in a TCP connection, and then counterfeit packets.
- Adversary: do not have full control over the network, but can inject packets with fake source IP addresses
 - ▶ E.g., control a computer on the local network
- TCP sequence numbers are used for authenticating packets
- Initial seq# needs high degree of unpredictability
 - ▶ If attacker knows initial seq # and amount of traffic sent, can estimate likely current values
 - ▶ Some implementations are vulnerable

Blind TCP Session Hijacking

- A, B trusted connection
 - ▶ Send packets with predictable seq numbers
- E impersonates B to A
 - ▶ Opens connection to A to get initial seq number
 - ▶ DoS B's queue
 - ▶ Sends packets to A that resemble B's transmission
 - ▶ E cannot receive, but may execute commands on A



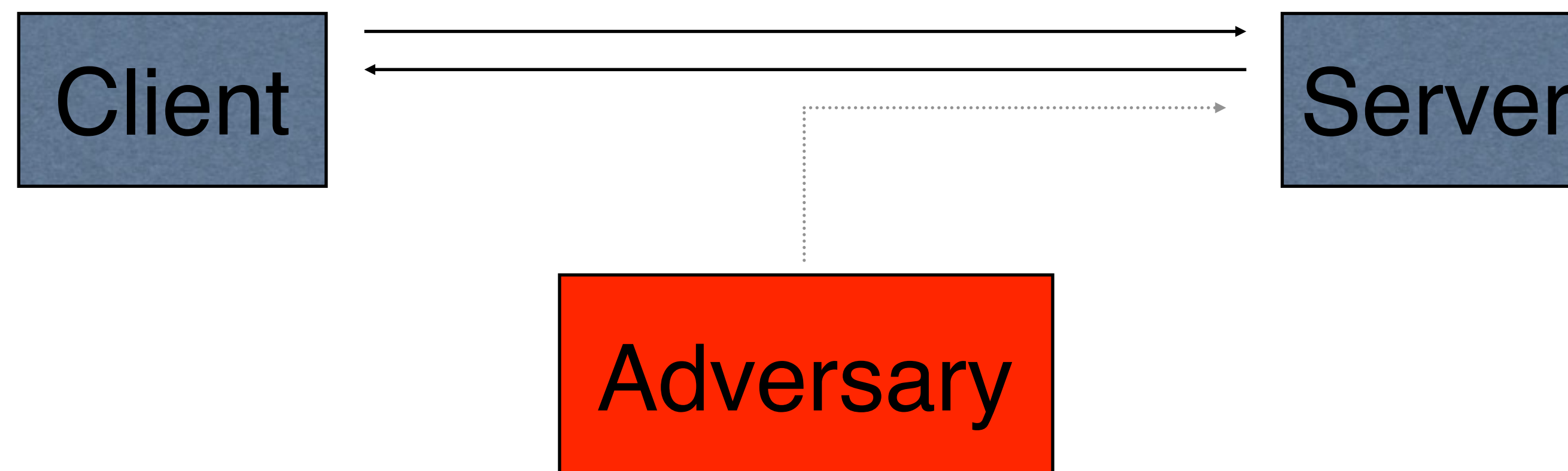
Attack can be blocked if E is outside firewall.

Risks from Session Hijacking

- Inject data into an **unencrypted** server-to-server traffic, such as an e-mail exchange, DNS zone transfers, etc.
- Inject data into an **unencrypted** client-to-server traffic, such as ftp file downloads, http responses.
- Spoof IP addresses, which are often used for preliminary checks on firewalls or at the service level.
- Carry out MITM attacks on weak cryptographic protocols.
 - ▶ often result in warnings to users that get ignored
- Denial of service attacks, such as resetting the connection.

Sequence number prediction

- TCP/IP uses a *three-way handshake* to establish a connection
 1. Client \rightarrow Server: Q_C
 2. Server \rightarrow Client: $Q_S, \text{ack}(Q_C)$ where sequence number Q_S is nonce
 3. C \rightarrow S: $\text{ack}(Q_S)$... then send data
- 2. However assume the bad guy does not hear msg 2, if he can guess Q_S , then he can get S to accept whatever data it wants (useful if doing IP authentication, e.g., “rsh”)

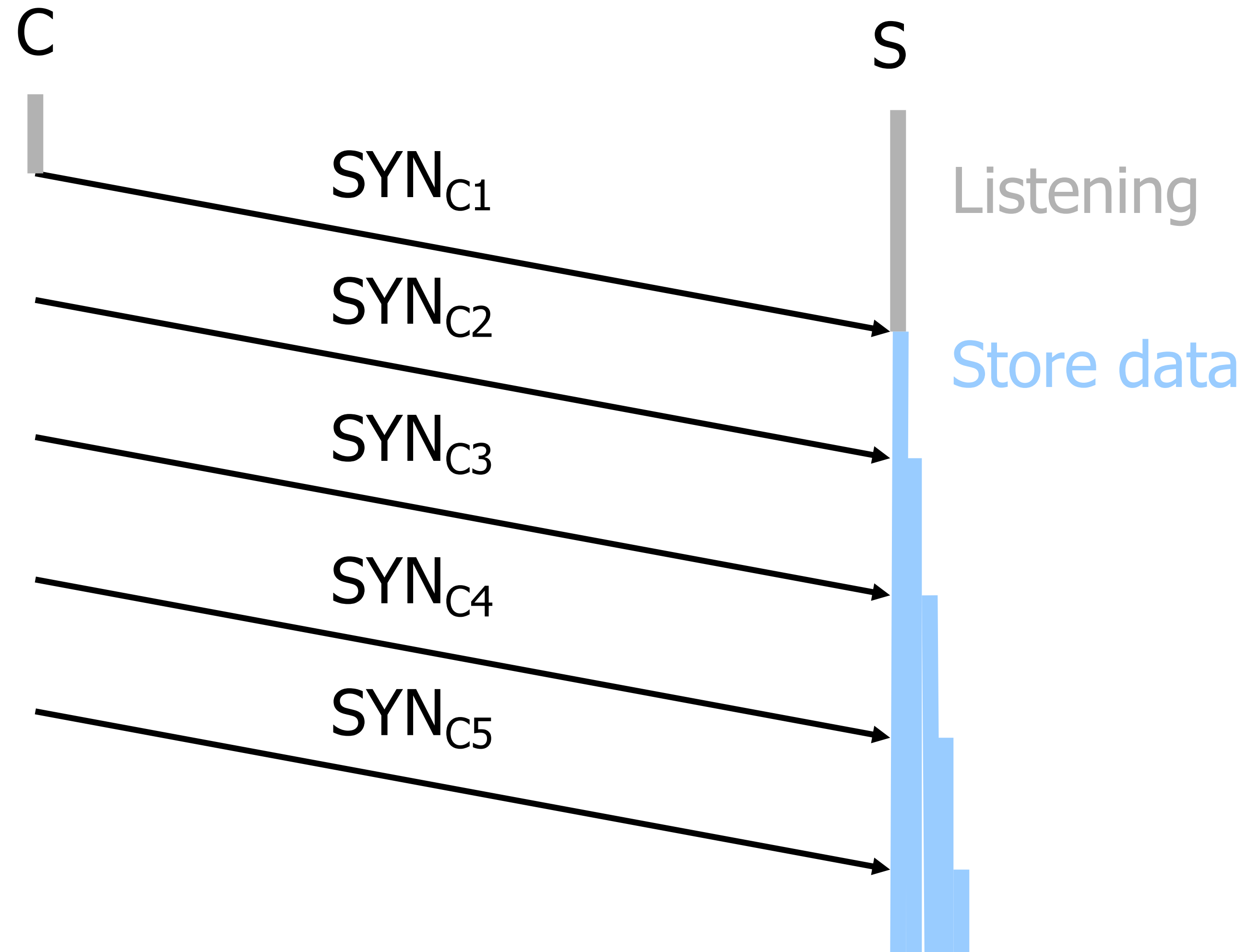


- Suppose attacker can guess seq. number for an existing connection:
 - ▶ Attacker can send Reset packet to close connection.
 - Results in DoS.
 - ▶ Naively, success prob. is $1/2^{32}$ (32-bit seq. #'s).
 - ▶ Most systems allow for a large window of acceptable seq. #'s
 - Much higher success probability.
- Attack is most effective against long lived connections, e.g. BGP.

Categories of DoS Attacks

	Stopping services	Exhausting resources
Locally	<ul style="list-style-type: none">• Process killing• Process crashing• System reconfiguration	<ul style="list-style-type: none">• Spawning processes to fill the process table• Filling up the whole file system• Saturate comm bandwidth
Remotely	<ul style="list-style-type: none">• Malformed packets to crash buggy services	<ul style="list-style-type: none">• Packet floods (Smurf, SYN flood, DDoS, etc)

SYN Flooding



SYN Flooding

- Attacker sends many connection requests
 - ▶ Spoofed source addresses
- Victim allocates resources for each request
 - ▶ Connection requests exist until timeout
 - ▶ Old implementations have a small and fixed bound on half-open connections
- Resources exhausted => requests rejected
- No more effective than other channel capacity-based attack today
-

Sequence Number Prediction (fixes)

- The only way you really fix this problem to stop making the sequence numbers predictable:
 - ▶ Randomize them -- you can use DES or some other mechanism to generate them randomly
 - ▶ There is an entire sub-field devoted to the creation and management of randomness in OSes
- Also, you could look for inconsistencies in timing information
 - ▶ Assumption: the adversary has different timing
 - ▶ OK, may be helpful, but far from definitive

- Collaborative TCP Sequence Number Inference Attack -- How to Crack Sequence Number Under A Second

Zhiyun Qian, Z. Morley Mao, Yinglian Xie

In Proceedings of ACM Conference on Computer and Communications Security (CCS) 2012, Raleigh, NC.

- Off-Path TCP Sequence Number Inference Attack -- How Firewall Middleboxes Reduce Security

Zhiyun Qian, Z. Morley Mao

In Proceedings of IEEE Security and Privacy (Oakland) 2012, San Francisco, CA.

- Still have TCP sequence number attacks

Internet Control Message Protocol (ICMP)

- ICMP is used as a control plane for IP messages
 - ▶ Ping (connectivity probe)
 - ▶ Destination Unreachable (error notification)
 - ▶ Time-to-live exceeded (error notification)
- These are largely indispensable tools for network management and control
 - ▶ Error notification codes can be used to reset connections without any authentication
- Solution: verify/sanity check sources and content
 - ▶ ICMP “returned packets”
- Real solution: filter most of ICMP, ignore it

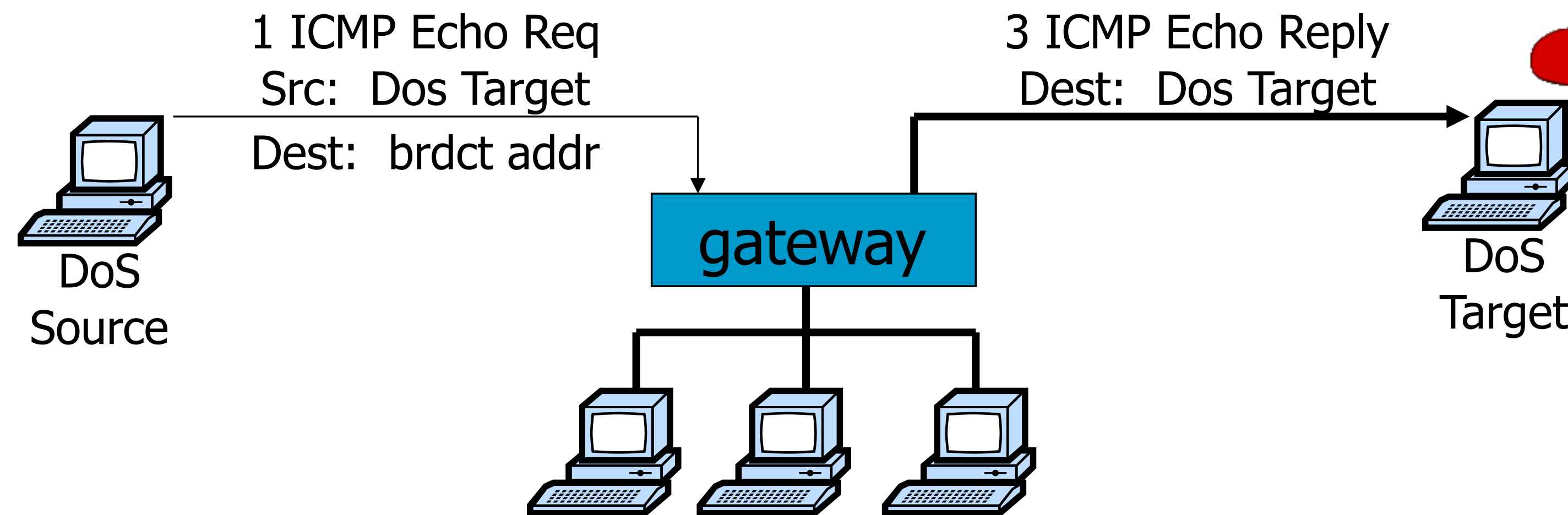
The “ping of death” ...

- In 1996, someone discovered that many operating systems, routers, etc. could be crashed/rebooted by sending a single malformed packet
 - ▶ It turns out that you can send a IP packet larger than 65,535 (2^{16}), it would crash the system
 - ▶ The real reason lies in the way fragmentation works
 - It allows somebody to send a packet bigger than IP allows which blows up most fixed buffer size implementations
 - ... and dumps core, blue screen of death, etc.
 - ▶ Note: this is not really ICMP specific, but easy (try it)

```
% ping -l 65555 your.host.ip.address
```
- This was a popular pastime of early hackers

Smurf 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
- Ping reply stream can overload victim
-



Prevention: reject external packets to broadcast address

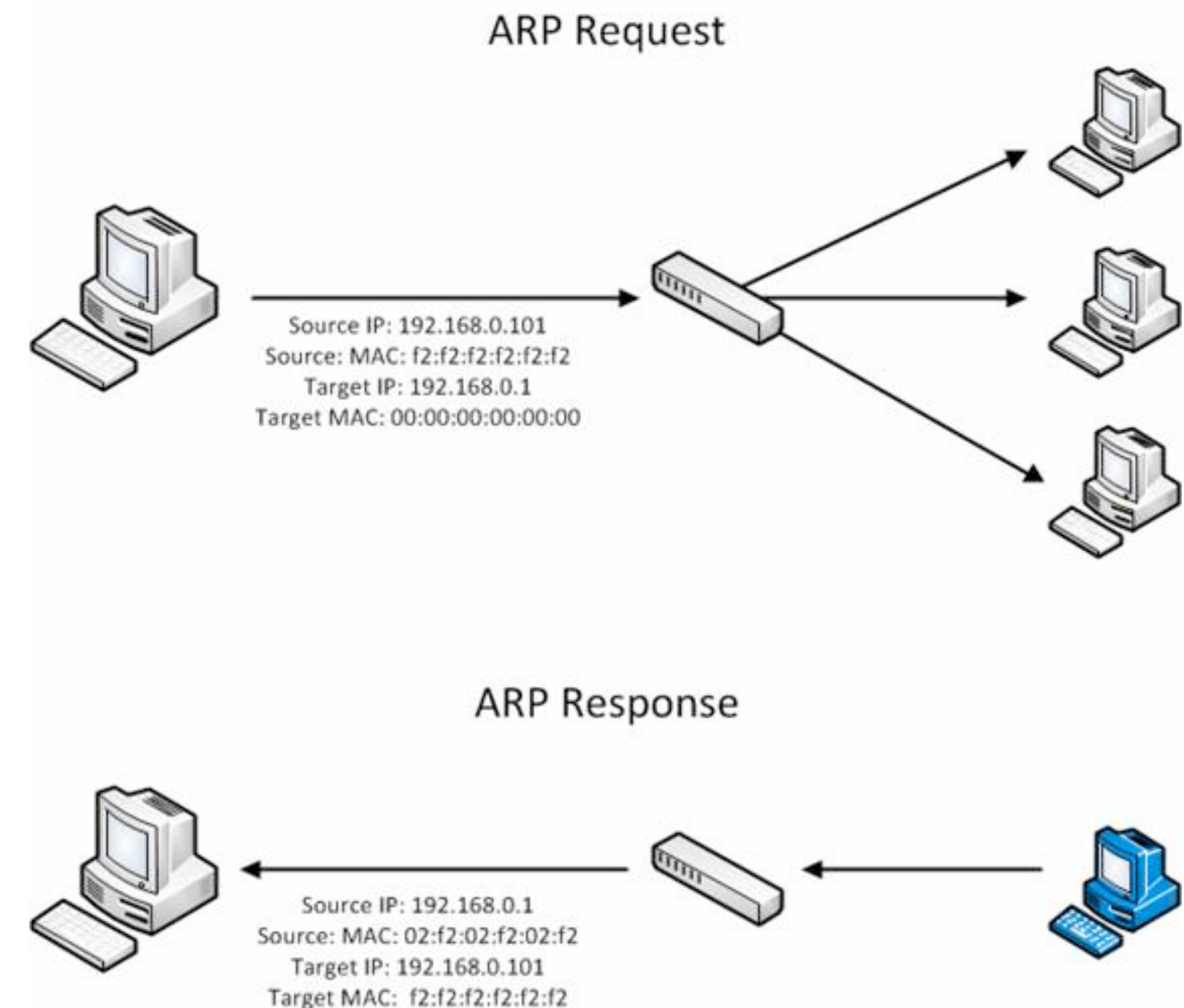
Address Resolution Protocol (ARP)

- Protocol used to map IP address onto the physical layer addresses (MAC)

1) ARP request: who has x.x.x.x?

2) ARP response: me!

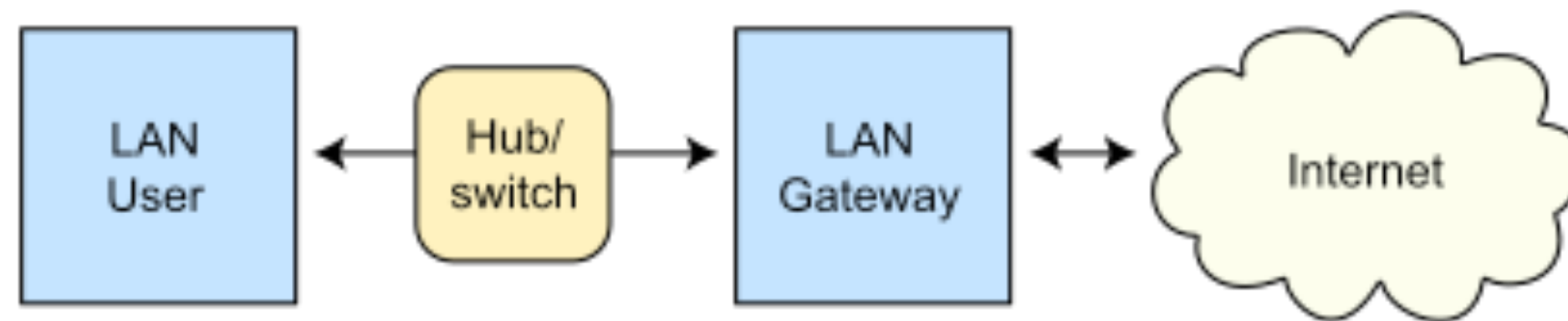
- Policy: **last response in wins**
- Used to forward packets on the appropriate interfaces by network devices
 - ▶ Also used for IP over other LAN technologies, e.g. IEEE 802.11
 - ▶ **Each host maintains a table of IP to MAC addresses**
Q: Why would you want to spoof an IP address?



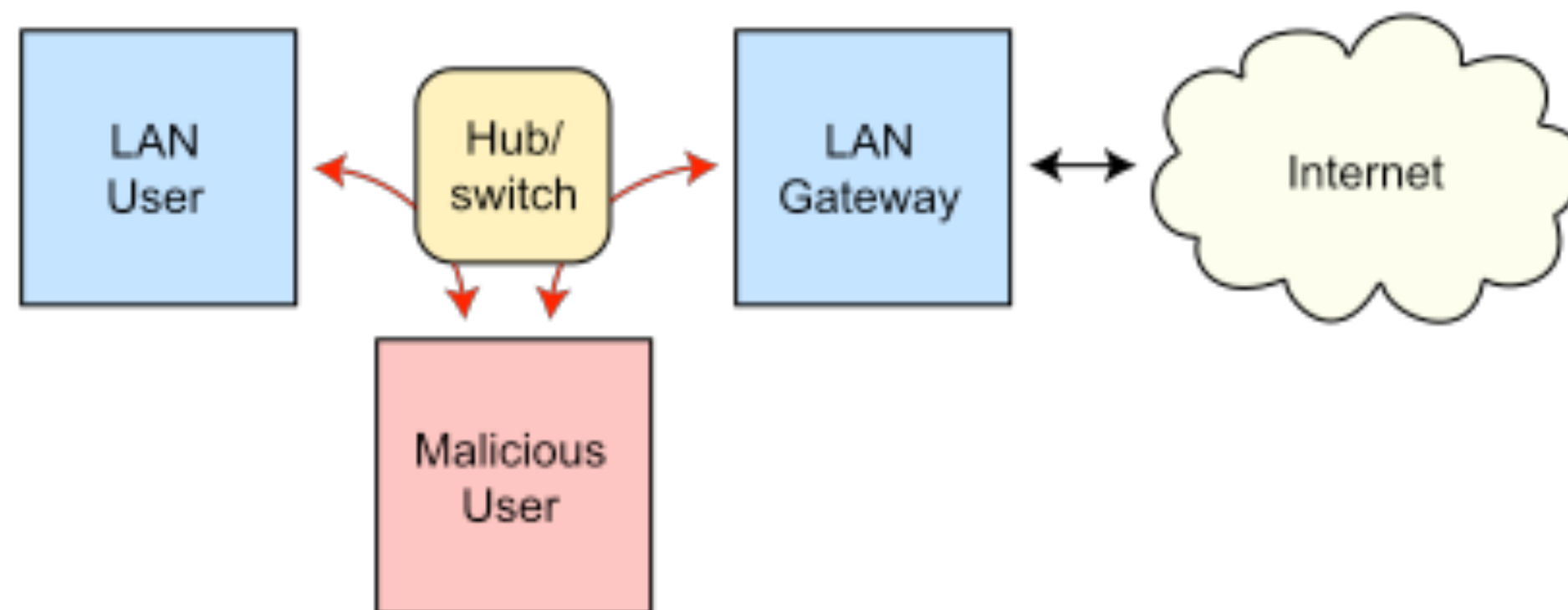
ARP Spoofing (Poisoning)

- Send fake or 'spoofed' ARP messages to an Ethernet LAN.
- To have other machines associate IP addresses with the attacker's MAC

Routing under normal operation



Routing subject to ARP cache poisoning



- Defenses

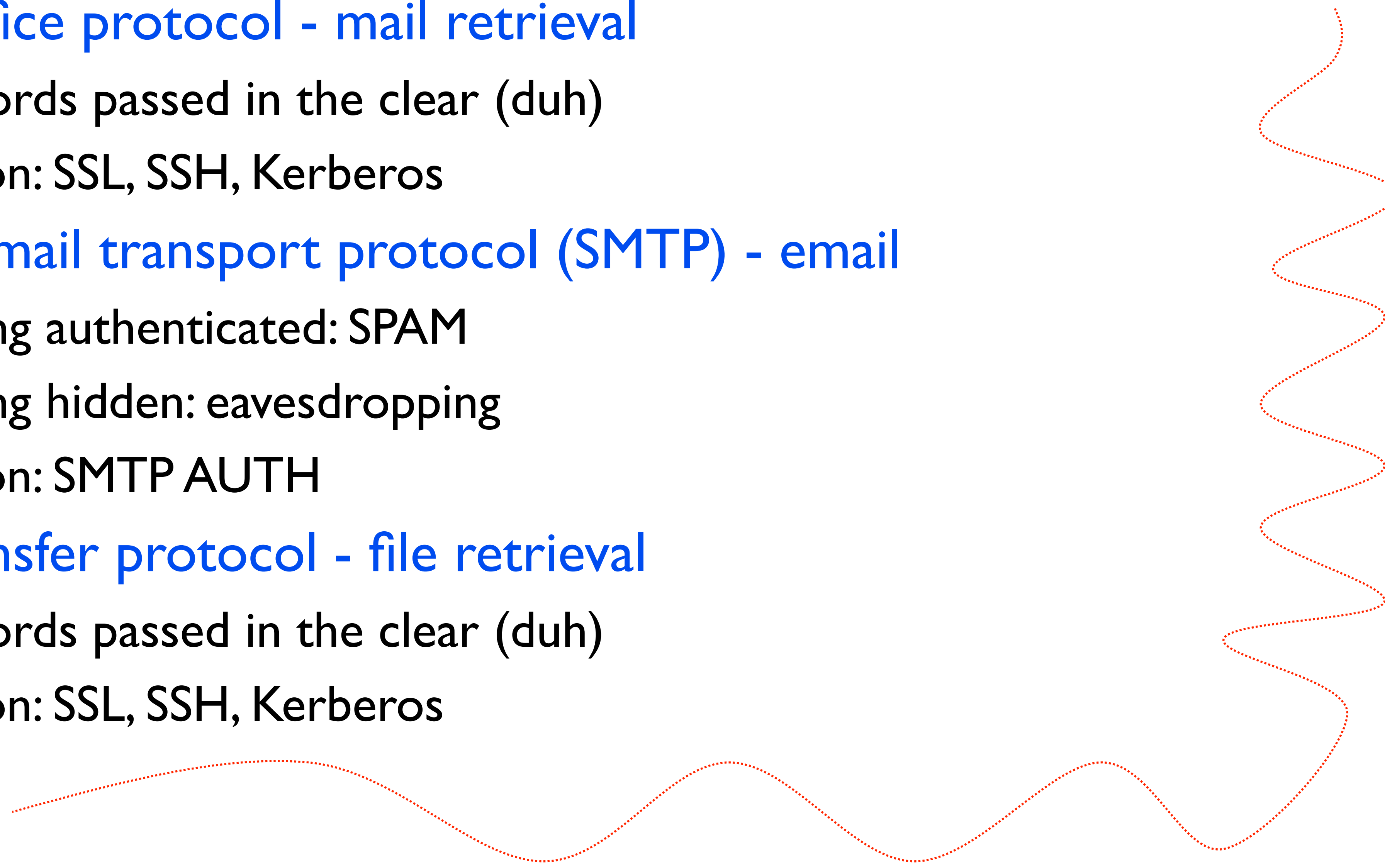
- ▶ static ARP table
- ▶ DHCP Certification (use access control to ensure that hosts only use the IP addresses assigned to them, and that only authorized DHCP servers are accessible).
- ▶ Detection: Arpwatch (sending email when updates occur),

ARP poisoning

- Attack: replace good entries with your own
- Leads to
 - ▶ Session hijacking
 - ▶ Man-in-the-middle attacks
 - ▶ Denial of service, etc.
- Lots of other ways to abuse ARP.
- Nobody has really come up with a good solution
 - ▶ Except smart switches, routers that keep track of MACs
- However, some not worried
 - ▶ If adversary is in your perimeter, you are in big trouble
 - ▶ You should validate the source of each packet independently (e.g., via IPsec)

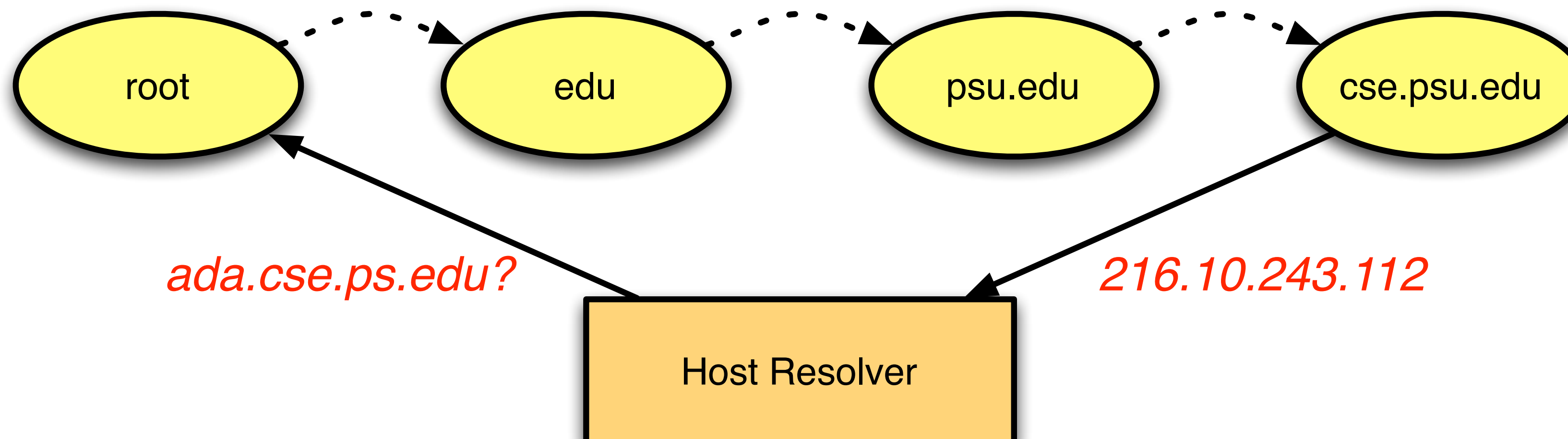


- **Post office protocol - mail retrieval**
 - ▶ Passwords passed in the clear (duh)
 - ▶ Solution: SSL, SSH, Kerberos
- **Simple mail transport protocol (SMTP) - email**
 - ▶ Nothing authenticated: SPAM
 - ▶ Nothing hidden: eavesdropping
 - ▶ Solution: SMTP AUTH
- **File Transfer protocol - file retrieval**
 - ▶ Passwords passed in the clear (duh)
 - ▶ Solution: SSL, SSH, Kerberos

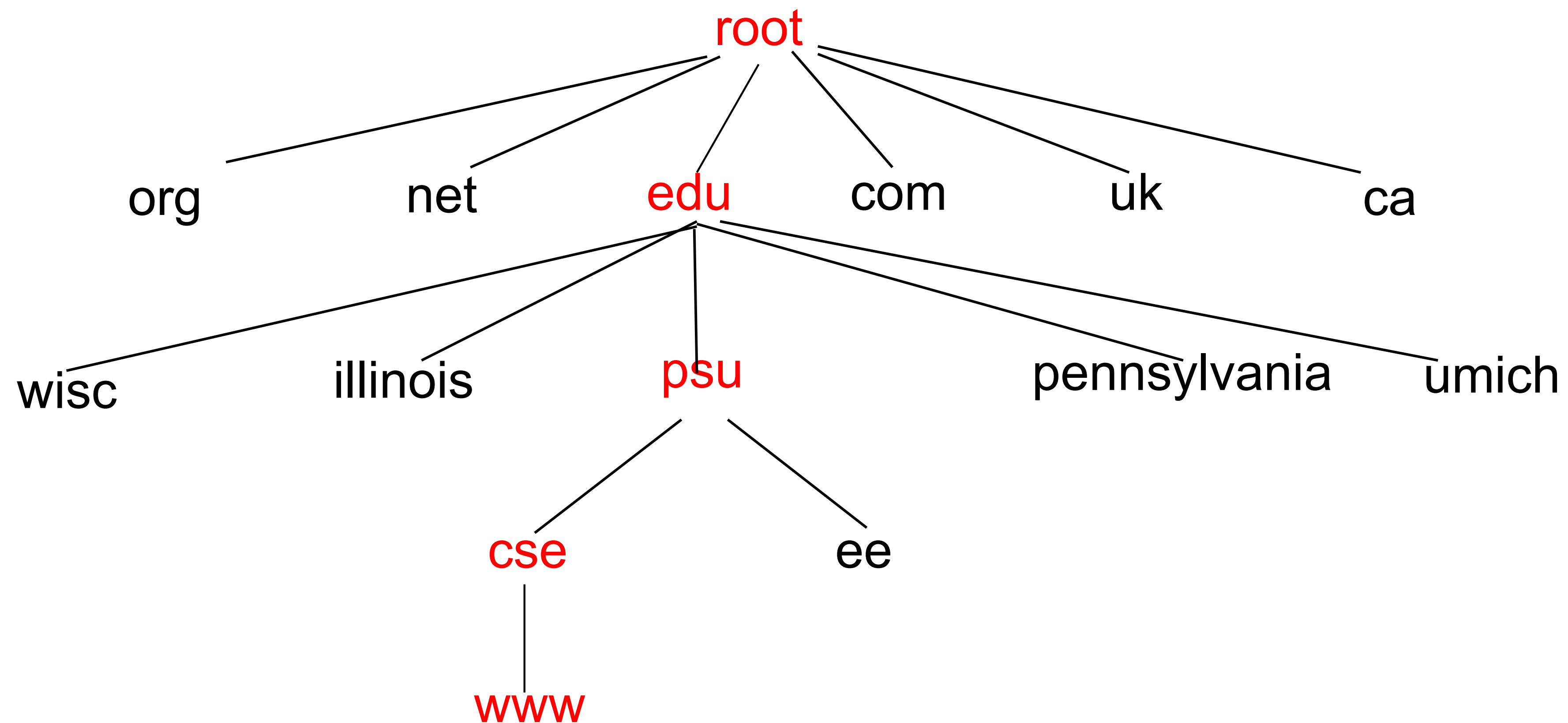


DNS - The domain name system

- DNS maps between IP address (12.1.1.3) and domain and host names (ada.cse.psu.edu)
 - ▶ How it works: the “root” servers redirect you to the top level domains (TLD) DNS servers, which redirect you to the appropriate sub-domain, and recursively
 - ▶ Note: there are 13 “root” servers that contain the TLDs for .org, .edu, and country specific registries (.fr, .ch)



Hierarchical Name Space



- **Top-level domain (TLD) servers:**
 - ▶ responsible for com, org, net, edu, etc, and all top-level country domains, e.g. uk, fr, ca, jp.
 - ▶ Network Solutions maintains servers for “.com”
- **Authoritative DNS servers:**
 - ▶ organization’s DNS servers, providing authoritative hostname to IP mappings for organization’s servers.
 - ▶ can be maintained by organization or service provider
- **Local Name Server**
 - ▶ does not strictly belong to hierarchy
 - ▶ each ISP (residential ISP, company, university) has one.

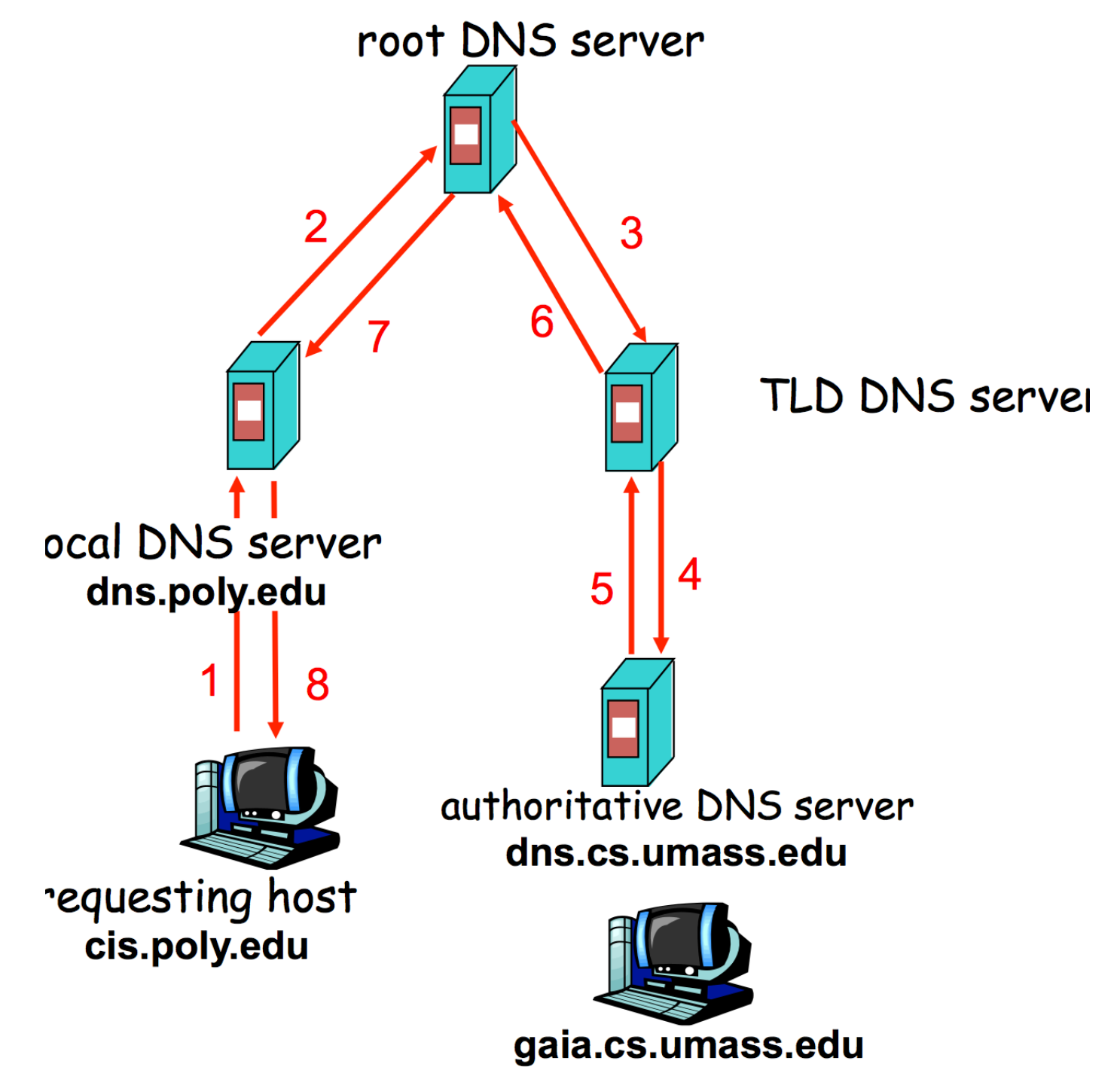
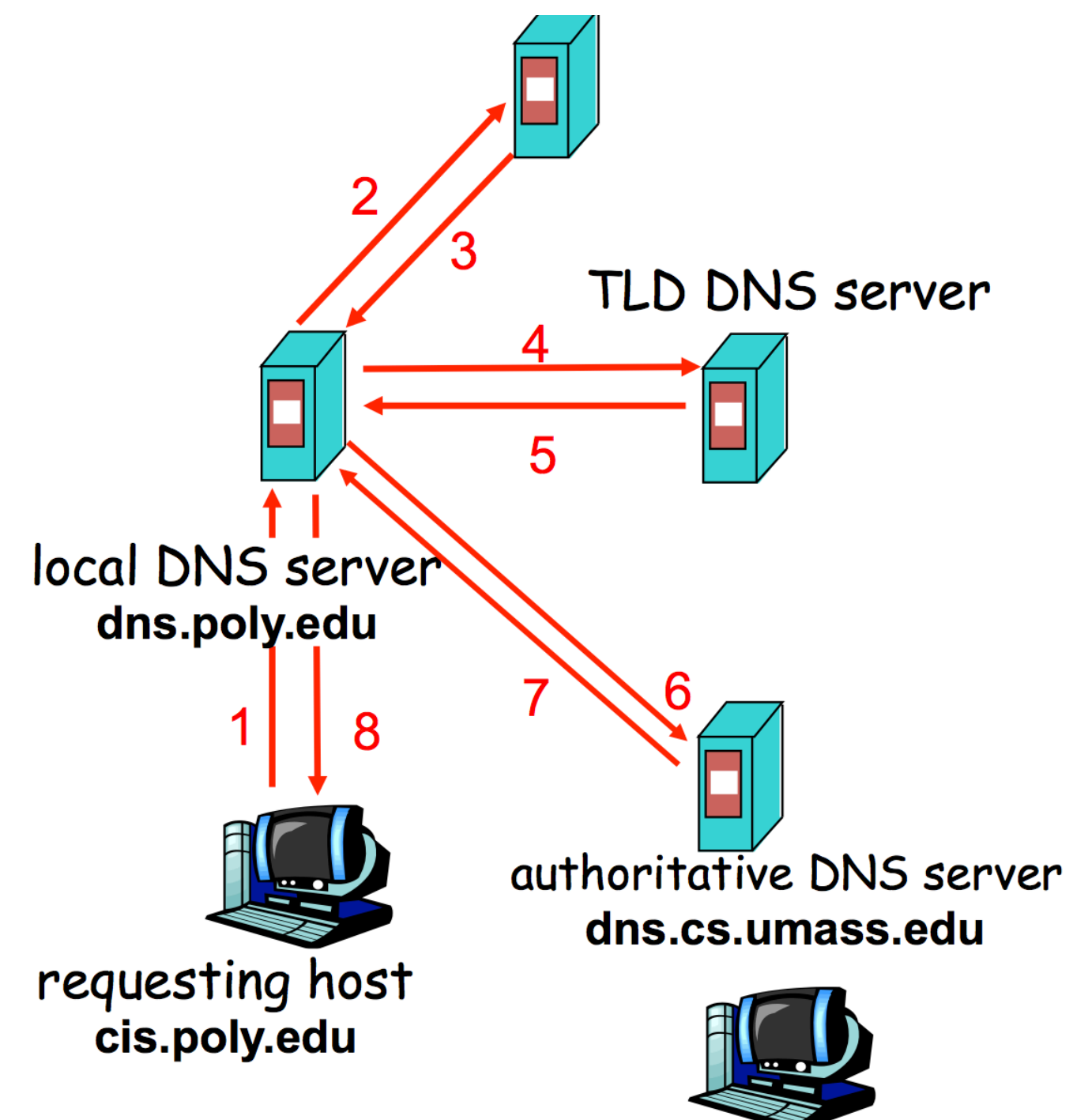


DNS Resolving

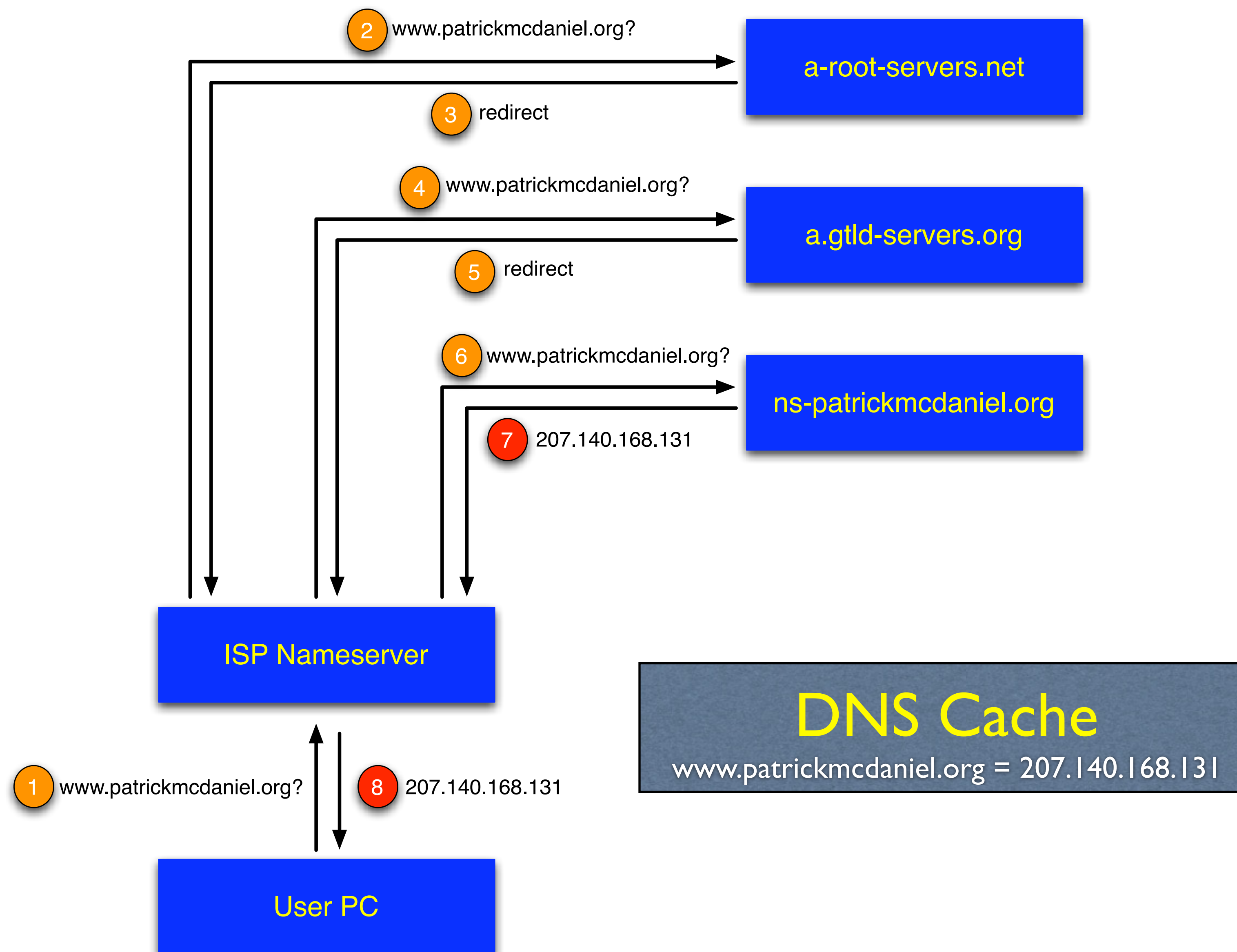
- When host makes DNS query, query is sent to its local DNS server.
 - ▶ acts as proxy, forwards query into hierarchy.

- Two resolving schemes:

- ▶ Iterative, and
- ▶ Recursive.

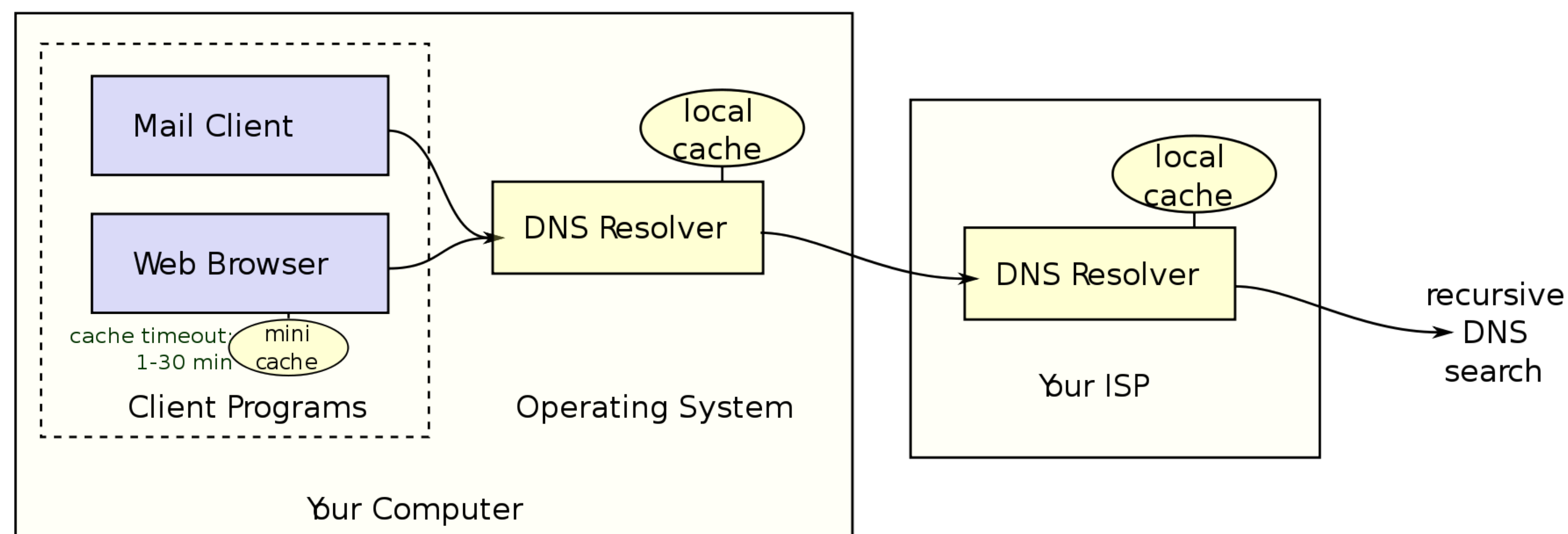


A DNS query



Caching

- DNS responses are cached
 - ▶ Quick response for repeated translations
- Negative results are also cached
 - ▶ Save time for nonexistent sites, e.g. misspelling
- Cached data periodically times out
 - ▶ Each record has a TTL field



“Glue” information

- Suppose you ask a name server for a record and it redirects you to another name server (NS record)
 - ▶ e.g., if you ask a root for a **NS** (name server) record for NET, it returns **NS records** for the authoritative servers for .net
- It will also give you the **A** (resource) record for the authoritative servers you were directed to
 - ▶ avoid looking them up
 - ▶ This is known as the “glue” records

🔍 .NET referrals

```
/* Authority section */
NET.                IN  NS  A.GTLD-SERVERS.NET.
                   IN  NS  B.GTLD-SERVERS.NET.
                   IN  NS  C.GTLD-SERVERS.NET.
                   ...
                   IN  NS  M.GTLD-SERVERS.NET.

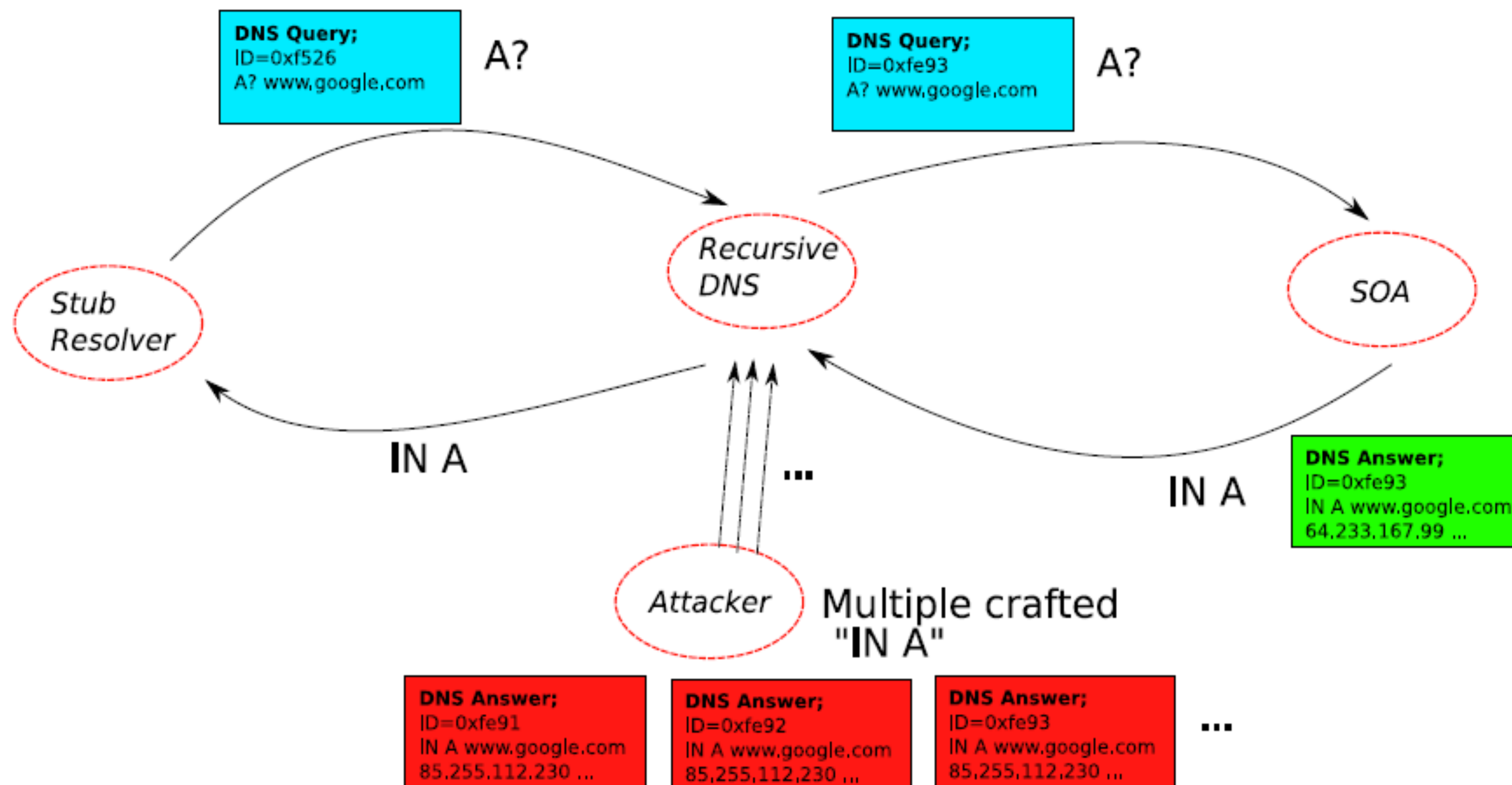
/* Additional section - "glue" records */
A.GTLD-SERVERS.net. IN  A  192.5.6.30
B.GTLD-SERVERS.net. IN  A  192.33.14.30
C.GTLD-SERVERS.net. IN  A  192.26.92.30
...
M.GTLD-SERVERS.net. IN  A  192.55.83.30
```


- **Nothing is authenticated, so really the game is over**
 - ▶ You cannot really trust what you hear ...
 - ▶ But, many applications are doing just that.
 - ▶ Spoofing of DNS is really dangerous
- **Moreover, DNS is a catalog of resources**
 - ▶ Zone-transfers allow bulk acquisition of DNS data
 - ▶ ... and hence provide a map for attacking the network
- **Lots of opportunity to abuse the system**
 - ▶ **Relies heavily on caching for efficiency -- cache pollution**
 - ▶ Once something is wrong, it can remain that way in caches for a long time (e.g., it takes a long time flush)
 - ▶ Data may be corrupted before it gets to authoritative server

A Cache Poisoning Attack

- All requests have a unique query ID
- The nameserver/resolver uses this information to match up requests and responses
- If an adversary can guess the query ID, then it can forge the responses and pollute the DNS cache
 - ▶ 16-bit query IDs (not hard)
 - ▶ Some servers increment IDs (or use other bad algo.)
 - ▶ First one in wins!!!
- Note: If you can observe the traffic going to a name server, you can pretty much arbitrarily own the Internet for the clients it serves.

DNS Cache Poisoning: Racing to Respond First



User Side Attack - Pharming

- Exploit DNS poisoning attack
 - ▶ Change IP addresses to redirect URLs to fraudulent sites
 - ▶ Potentially more dangerous than phishing attacks
 - Why?
- DNS poisoning attacks have occurred:
 - ▶ January 2005, the domain name for a large New York ISP, Panix, was hijacked to a site in Australia.
 - ▶ In November 2004, Google and Amazon users were sent to Med Network Inc., an online pharmacy
 - ▶

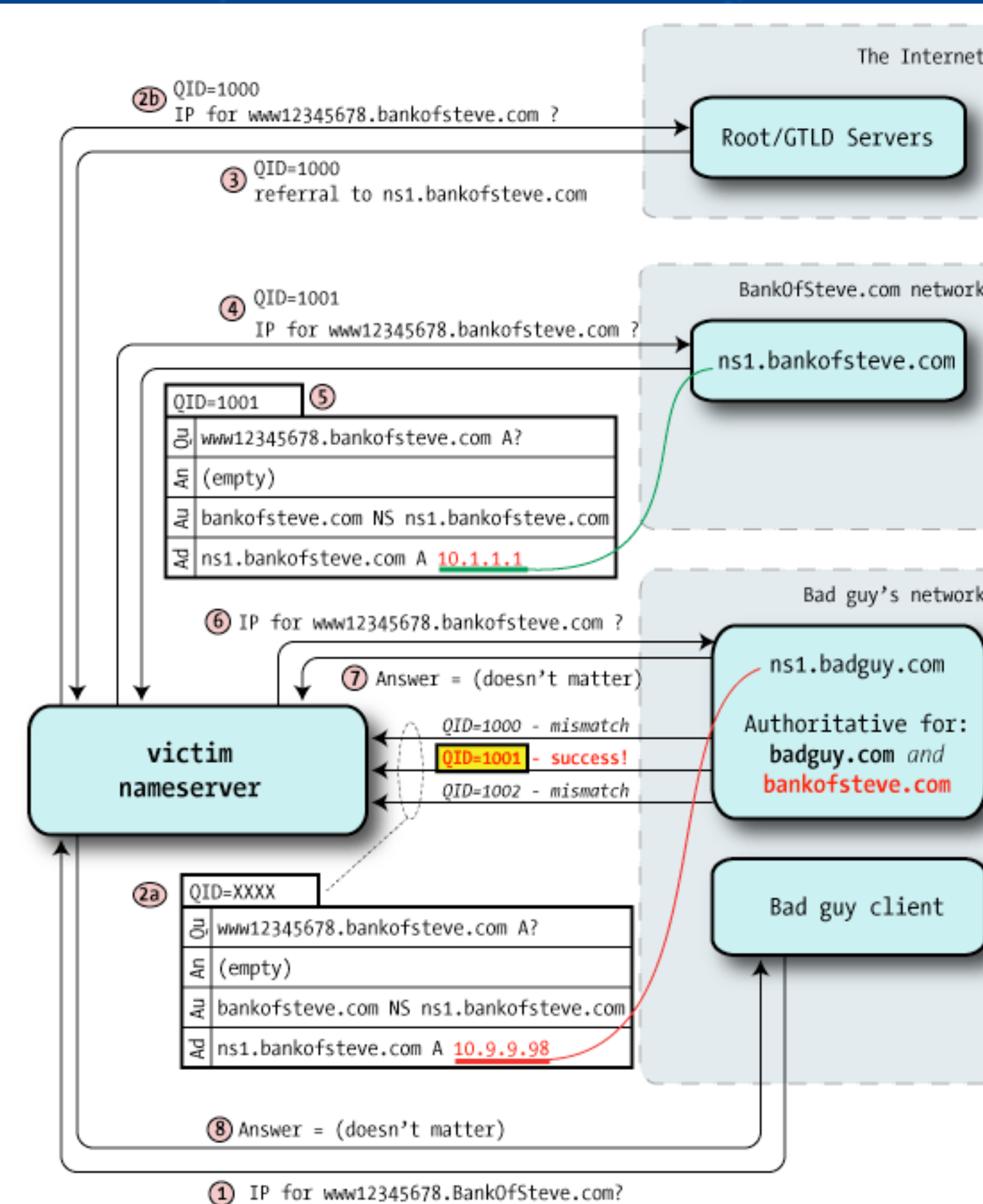
DNS Cache Poisoning

- Attacker wants his IP address returned for a DNS query
- When the resolver asks ns1.google.com for www.google.com, the attacker could reply first, with his own IP
- What is supposed to prevent this?
 - ▶ Transaction ID
 - 16-bit random number
 - The real server knows the number, because it was contained in the query
 - The attacker has to guess
-

- Responding before the real nameserver
 - ▶ An attacker can guess when a DNS cache entry times out and a query has been sent, and provide a fake response.
 - ▶ The fake response will be accepted only when its 16-bit transaction ID matches the query
 - ▶ CERT reported in 1997 that BIND uses sequential transaction ID and is easily predicted
 - ▶ fixed by using random transaction IDs

Kaminsky DNS Vulnerability

1. Query a random host in a victim zone, e.g., 1234.cse.psu.edu
2. Spoof responses* as before, but *delegate authority* to some server which you own.
1. The glue records you give make you authoritative
3. You now own the domain.
4. unixwiz.net/techtips/lguide-kaminsky-dns-vuln.html



*the original attack exploited poor ID selection

- Make the ID harder to guess (randomized ports)
 - ▶ Amplified ID space from 2^{16} to 2^{27}
- Prevent foreign requests from being processed
 - ▶ E.g., filter requests from outside domain
- Observe and filter conflicting requests
 - ▶ E.g., if you see a lot of bogus looking requests, be careful
- All of this treats the symptoms, not the disease.
 - ▶ Lack of authenticated values
 - ▶ Thus, if you can observe request traffic, prevent legitimate responses, or are just plain patient, you can mount these attacks.

- A standard-based (IETF) solution to security in DNS

- ▶ Prevents data spoofing and corruption
- ▶ Public key based solution to verifying DNS data

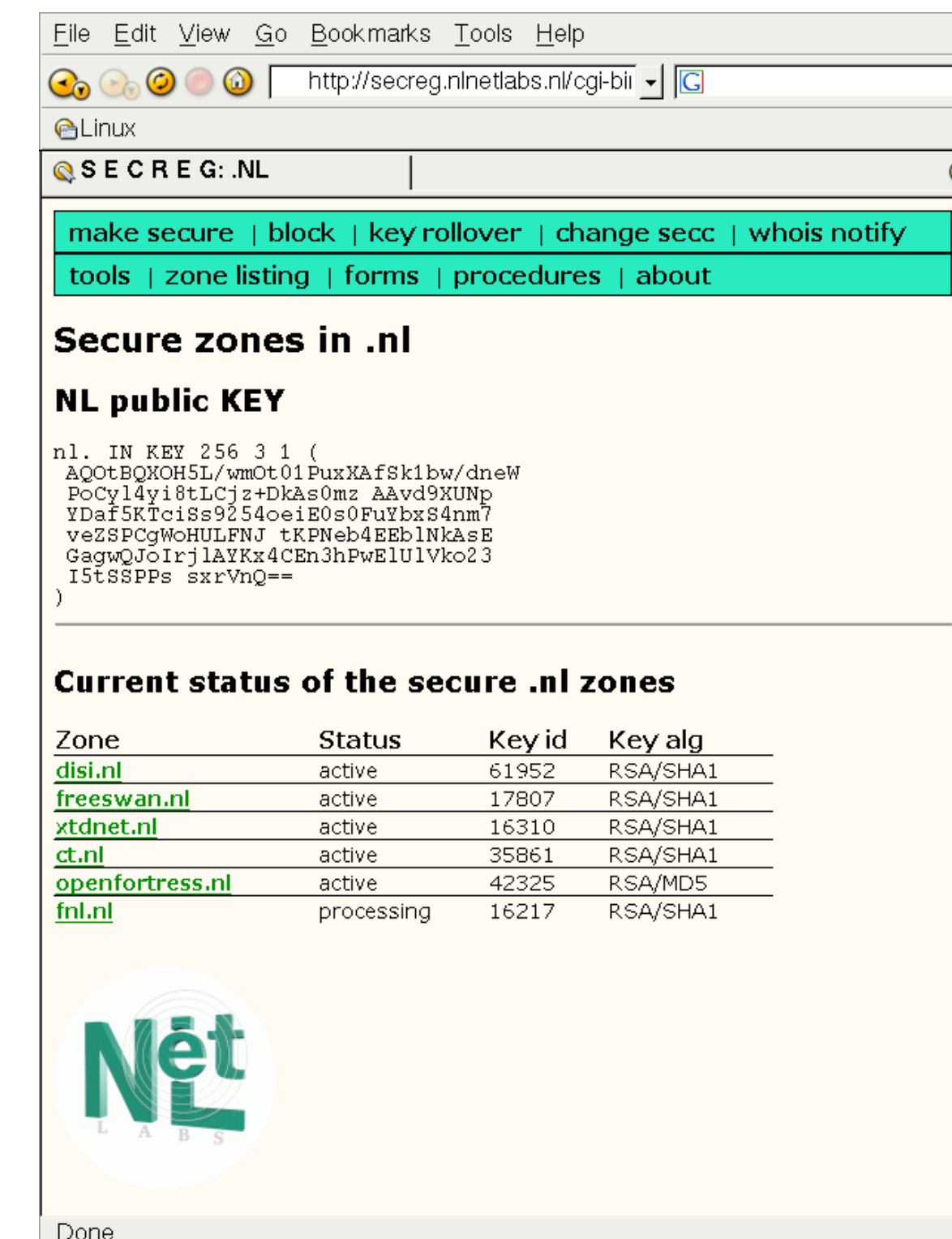
- ▶ Authenticates

- Communication between servers

- DNS data

- ▶ content
- ▶ existence
- ▶ non-existence

- Public keys (a bootstrap for PKI?)



File Edit View Go Bookmarks Tools Help

http://secreg.nlnetlabs.nl/cgi-bin

Linux

SECUREG.NL

[make secure](#) | [block](#) | [key rollover](#) | [change secc](#) | [whois notify](#)
[tools](#) | [zone listing](#) | [forms](#) | [procedures](#) | [about](#)


Secure zones in .nl

NL public KEY

```
n1. IN KEY 256 3 1 (  
AQOtBQXOH5L/wmOt01PuxKAfsk1bw/dneW  
PoCy14yi8tLCjz+DkAs0mz AAvd9XUNp  
YDaf5KTCiSs9254oeiE0s0FuYbxS4nm7  
veZSPCgWoHULFNJ tKPNeb4EEb1NkAsE  
GagwQJoIrrj1AYKx4CEn3hPwELU1Vko23  
I5tSSPPs sxrVnQ==  
)
```

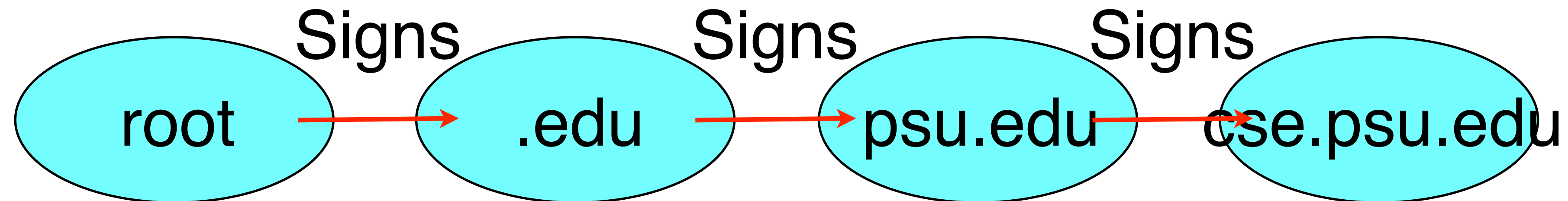
Current status of the secure .nl zones

Zone	Status	Key id	Key alg
disi.nl	active	61952	RSA/SHA1
freeswan.nl	active	17807	RSA/SHA1
xtdnet.nl	active	16310	RSA/SHA1
ct.nl	active	35861	RSA/SHA1
openfortress.nl	active	42325	RSA/MD5
fnl.nl	processing	16217	RSA/SHA1



Done

- Securing the DNS records
 - ▶ Each domain signs their “zone” with a private key
 - ▶ Public keys published via DNS
 - ▶ *Indirectly* signed by parent zones
 - ▶ Ideally, you only need a self-signed root, and follow keys down the hierarchy



DNSSEC Mechanisms

- **TSIG** : transaction signatures protect DNS operations
 - ▶ Zone loads, some server to server requests (master -> slave), etc.
 - ▶ Time-stamped signed responses for dynamic requests
 - ▶ A misnomer -- it currently uses shared secrets for TSIG (HMAC) or do real signatures using public key cryptography
- **SIG0**: a public key equivalent of TSIG
 - ▶ Works similarly, but with public keys
 - ▶ Not as popular as TSIG
- **Note**: these mechanisms assume clock sync. (NTP)

