



CS4700/5700: Network fundamentals

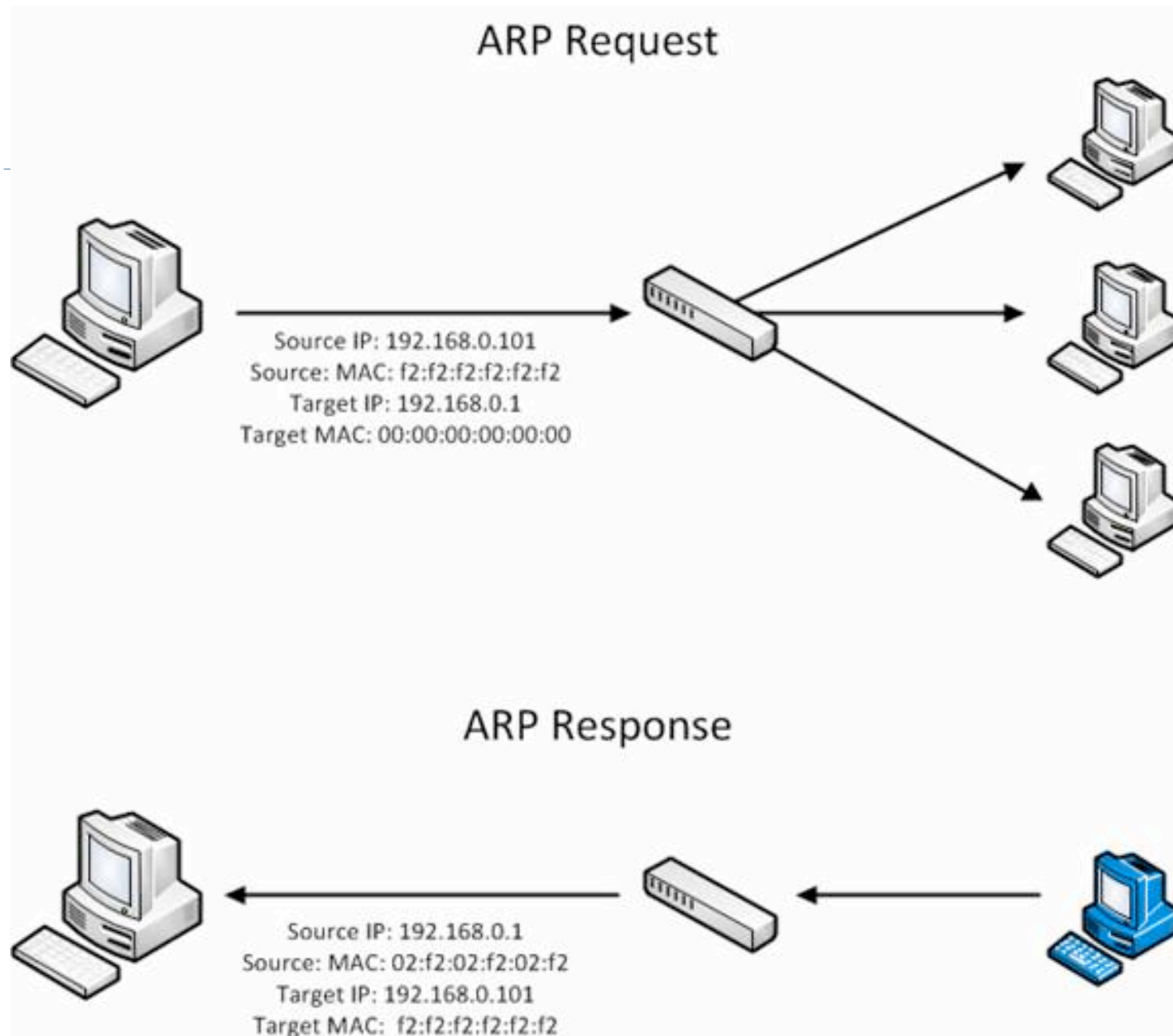
ARP. NAT, DNS.



1: ARP

Address Resolution Protocol (ARP)

- ▶ Primarily used to translate IP addresses to Ethernet MAC addresses
 - ▶ The device driver for Ethernet NIC needs ARP to send a packet
- ▶ Also used for IP over other LAN technologies, e.g. IEEE 802.11
- ▶ Each host maintains a table of IP to MAC addresses
- ▶ Message types:
 - ▶ ARP request
 - ▶ ARP reply
 - ▶ ARP announcement



<http://www.windowsecurity.com>

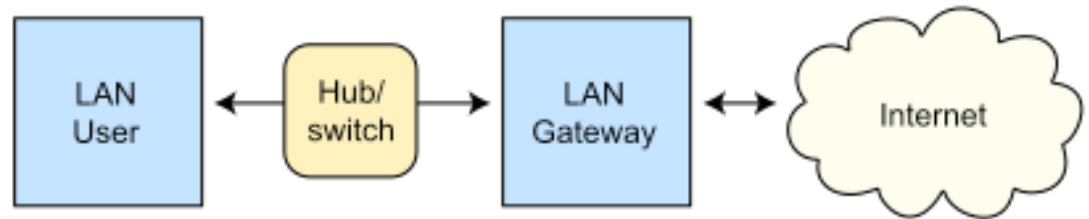
ARP packet

Hardware Type		Protocol Type
Hardware length	Protocol length	Operation Request 1, Reply 2
Sender hardware address (For example, 6 bytes for Ethernet)		
Sender protocol address (For example, 4 bytes for IP)		
Target hardware address (For example, 6 bytes for Ethernet) (It is not filled in a request)		
Target protocol address (For example, 4 bytes for IP)		

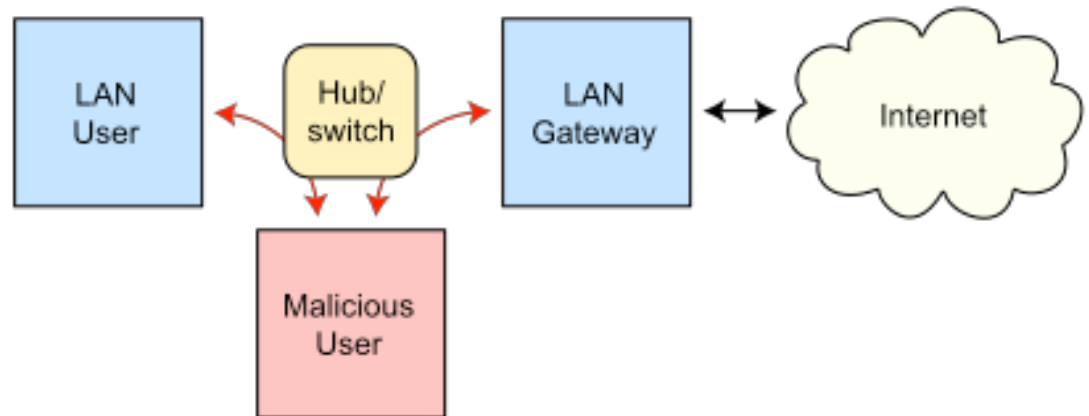
ARP spoofing (ARP Poisoning)

- ▶ Send fake or 'spoofed' ARP messages to an Ethernet LAN
 - ▶ To have other machines associate honest IP addresses with the attacker's MAC
- ▶ Legitimate use
 - ▶ Redirect a user to a registration page before allow usage of the network.
 - ▶ Implementing redundancy and fault tolerance

Routing under normal operation



Routing subject to ARP cache poisoning



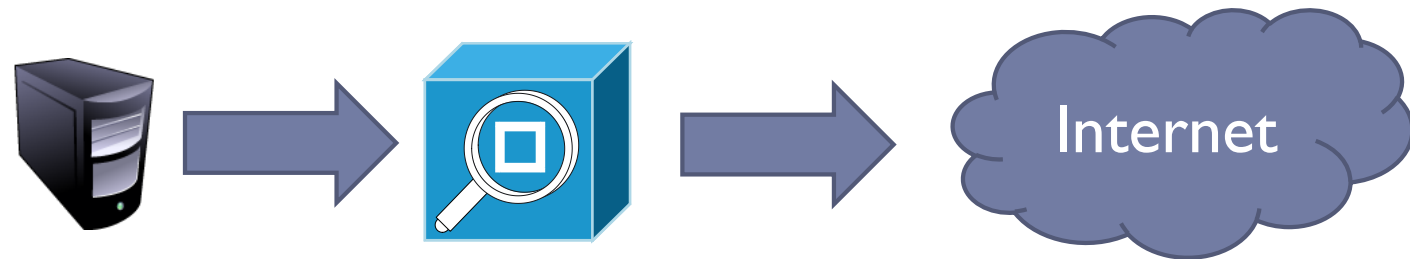
ARP spoofing defenses

- ▶ Static ARP table
 - ▶ Effective for small and fixed size networks
- ▶ 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).
- ▶ Monitoring and detection of changes: Arpwatch (can also send email when updates occur)
- ▶ Some switches can block gratuitous ARP replies

2: Middleboxes: NAT

Middleboxes

- ▶ Devices in the network that interact with network traffic from the IP layer and up
- ▶ Common functions
 - ▶ **NAT**
 - ▶ Firewall and other security
 - ▶ Proxy
 - ▶ Shaping
 - ▶ Filtering
 - ▶ Caching
 - ▶ ...



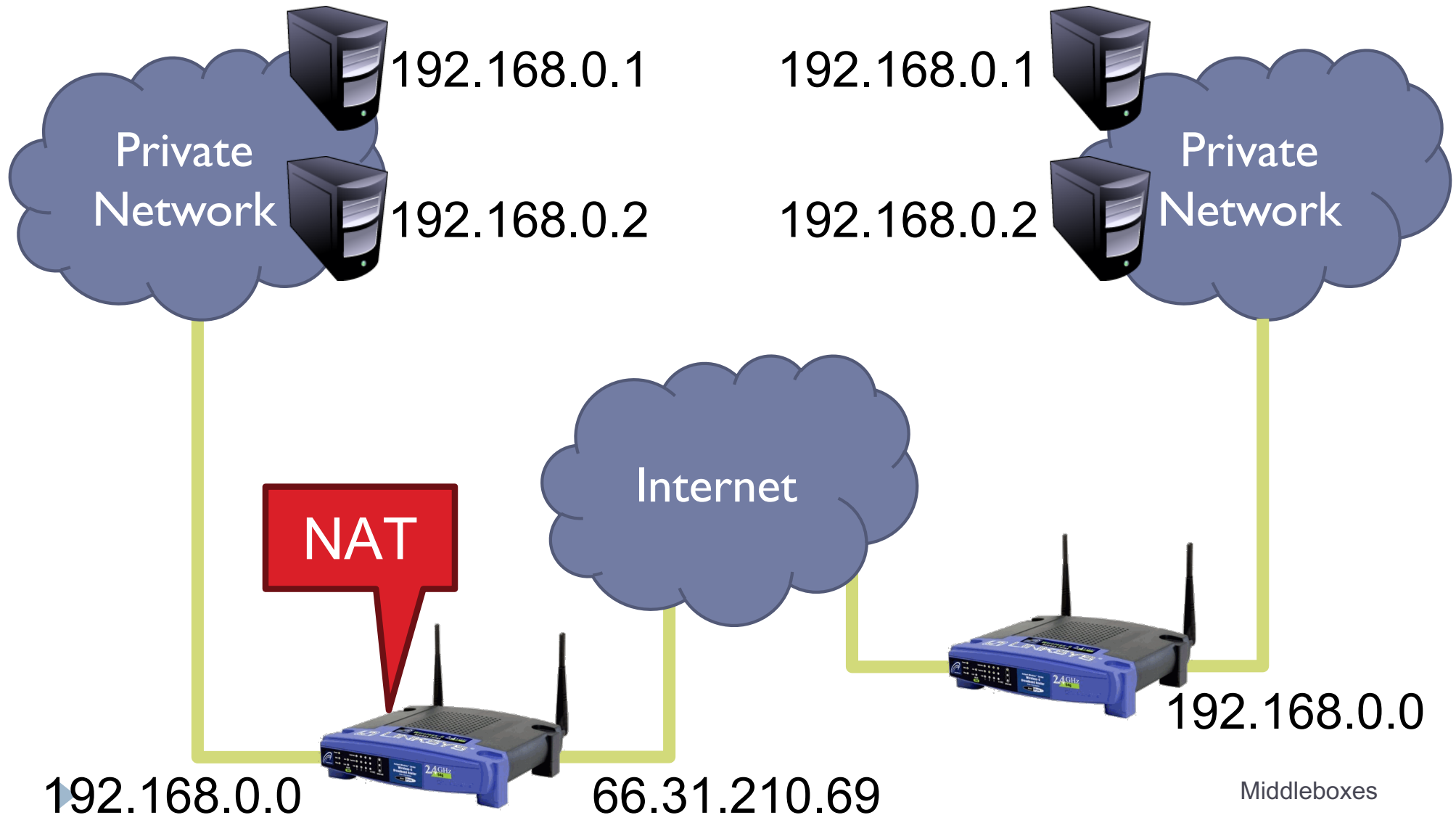
The IPv4 Shortage

- ▶ Problem: consumer ISPs typically only give one IP address per-household
 - ▶ Additional IPs cost extra
 - ▶ More IPs may not be available
- ▶ Today's households have more networked devices than ever
 - ▶ Laptops and desktops
 - ▶ TV, bluray players, game consoles
 - ▶ Tablets, smartphones, eReaders
- ▶ How to get all these devices online?

Private IP Networks

- ▶ **Idea: create a range of private IPs that are separate from the rest of the network**
 - ▶ Use the private IPs for internal routing
 - ▶ Use a special router to bridge the LAN and the WAN
- ▶ **Properties of private IPs**
 - ▶ Not globally unique
 - ▶ Usually taken from non-routable IP ranges (why?)
- ▶ **Typical private IP ranges**
 - ▶ 10.0.0.0 – 10.255.255.255
 - ▶ 172.16.0.0 – 172.31.255.255
 - ▶ 192.168.0.0 – 192.168.255.255

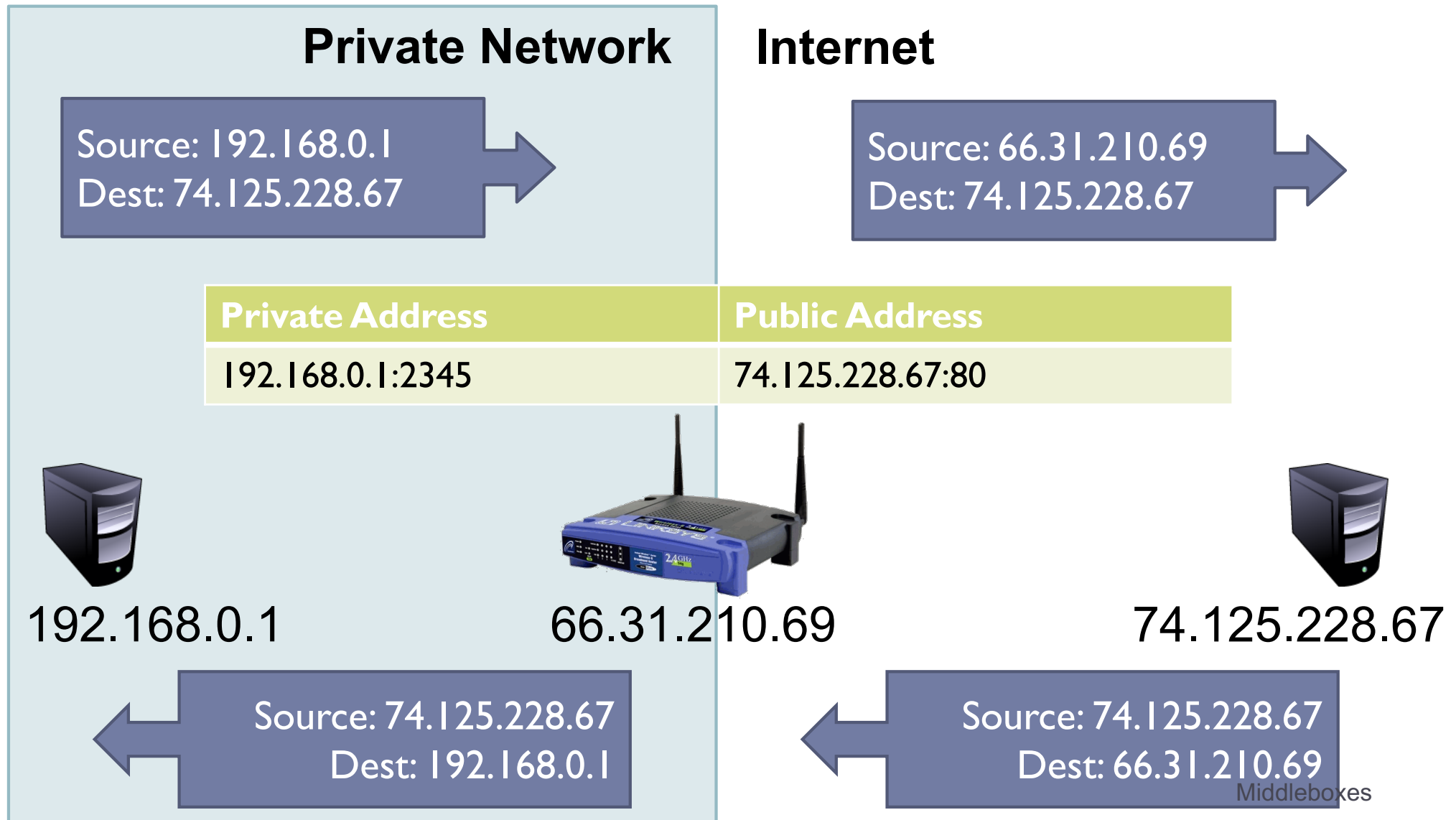
Private Networks



Network Address Translation (NAT)

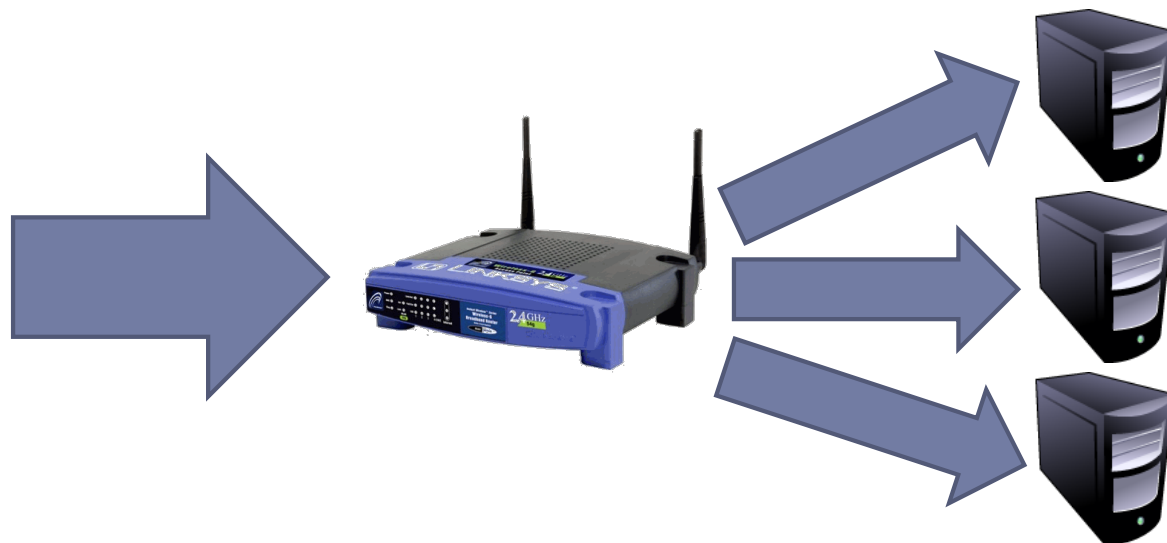
- ▶ **NAT allows hosts on a private network to communicate with the Internet**
 - ▶ Warning: connectivity is not seamless
- ▶ **Special router at the boundary of a private network**
 - ▶ Replaces internal IPs with external IP
 - ▶ This is “Network Address Translation”
 - ▶ May also replace TCP/UDP port numbers
- ▶ **Maintains a table of active flows**
 - ▶ Outgoing packets initialize a table entry
 - ▶ Incoming packets are rewritten based on the table

Basic NAT Operation



Advantages of NATs

- ▶ Allow multiple hosts to share a single public IP
- ▶ Allow migration between ISPs
 - ▶ Even if the public IP address changes, you don't need to reconfigure the machines on the LAN
- ▶ Load balancing
 - ▶ Forward traffic from a single public IP to multiple private hosts



Middleboxes

Natural Firewall

Private Network

Internet

Private Address

Public Address



192.168.0.1



66.31.210.69



74.125.228.67

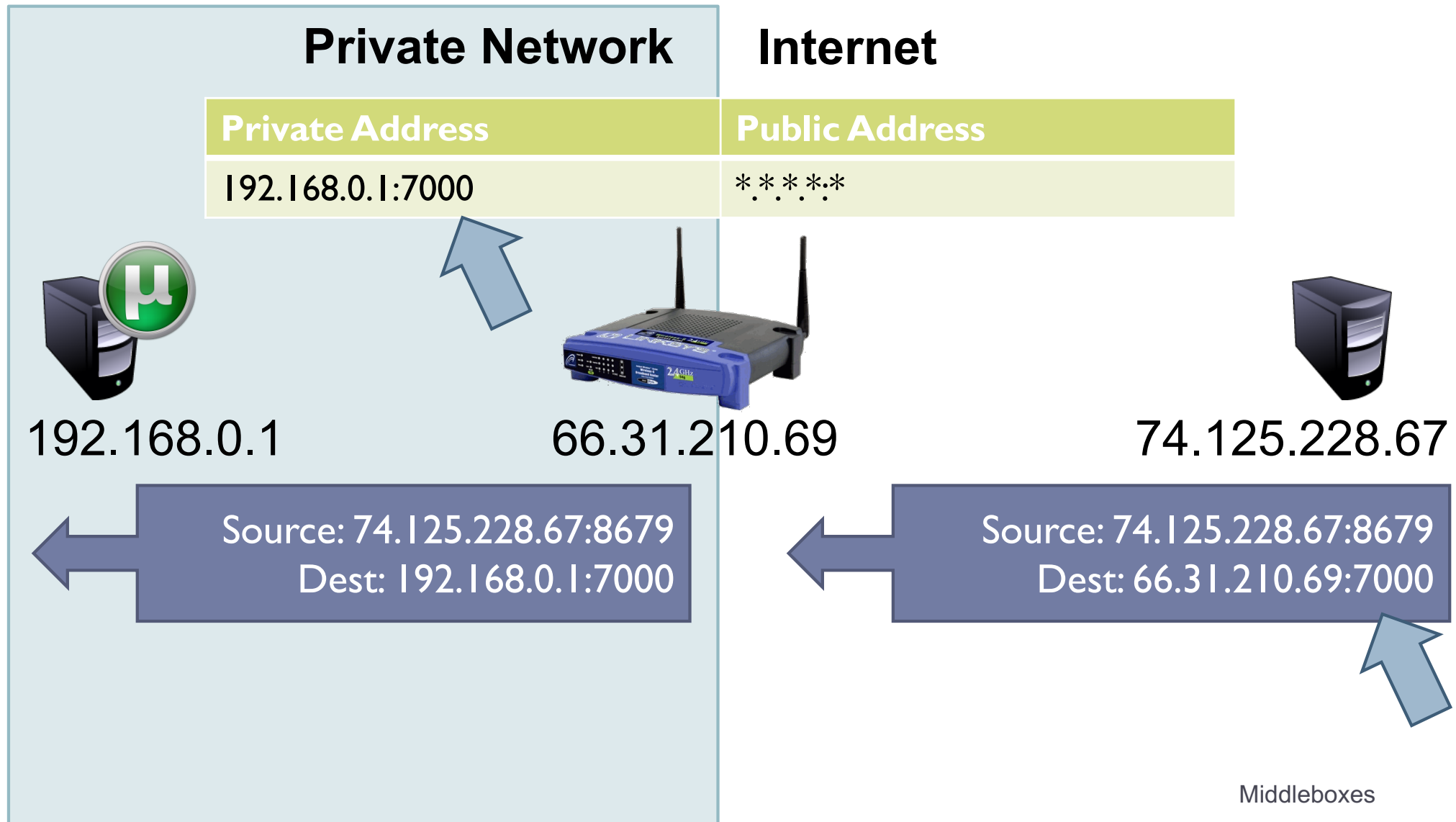
Source: 74.125.228.67
Dest: 66.31.210.69

Middleboxes

Concerns About NAT

- ▶ **Performance/scalability issues**
 - ▶ Per flow state!
 - ▶ Modifying IP and Port numbers means NAT must recompute IP and TCP checksums
- ▶ **Breaks the layered network abstraction**
- ▶ **Breaks end-to-end Internet connectivity**
 - ▶ 192.168.*.* addresses are private
 - ▶ Cannot be routed to on the Internet
 - ▶ Problem is worse when **both** hosts are behind NATs
- ▶ **What about IPs embedded in data payloads?**

Port Forwarding



Hole Punching

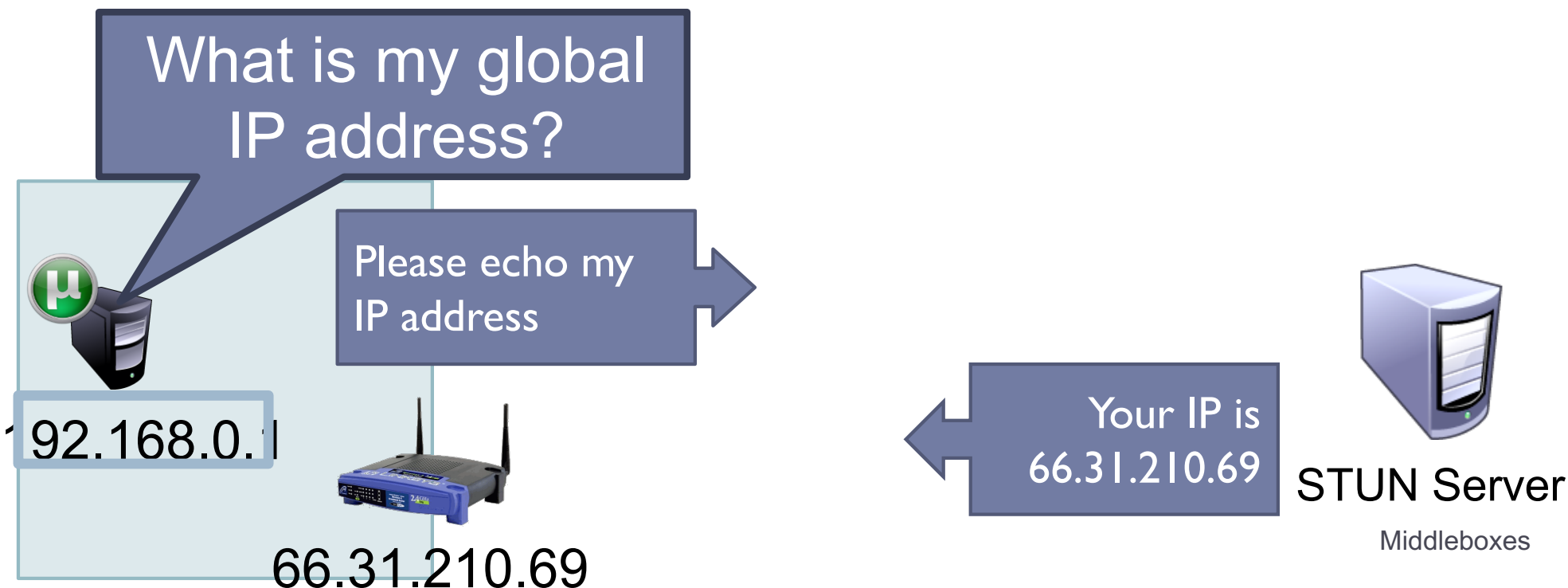
- ▶ Problem: How to enable connectivity through NATs?



- Two application-level protocols for hole punching
 - ▣ STUN
 - ▶ ▣ TURN

STUN

- ▶ **S**ession **T**raversal **U**tilities for **N**AT
 - ▶ Use a third-party to echo your global IP address
 - ▶ Also used to probe for symmetric NATs/firewalls
 - ▶ i.e. are external ports open or closed?



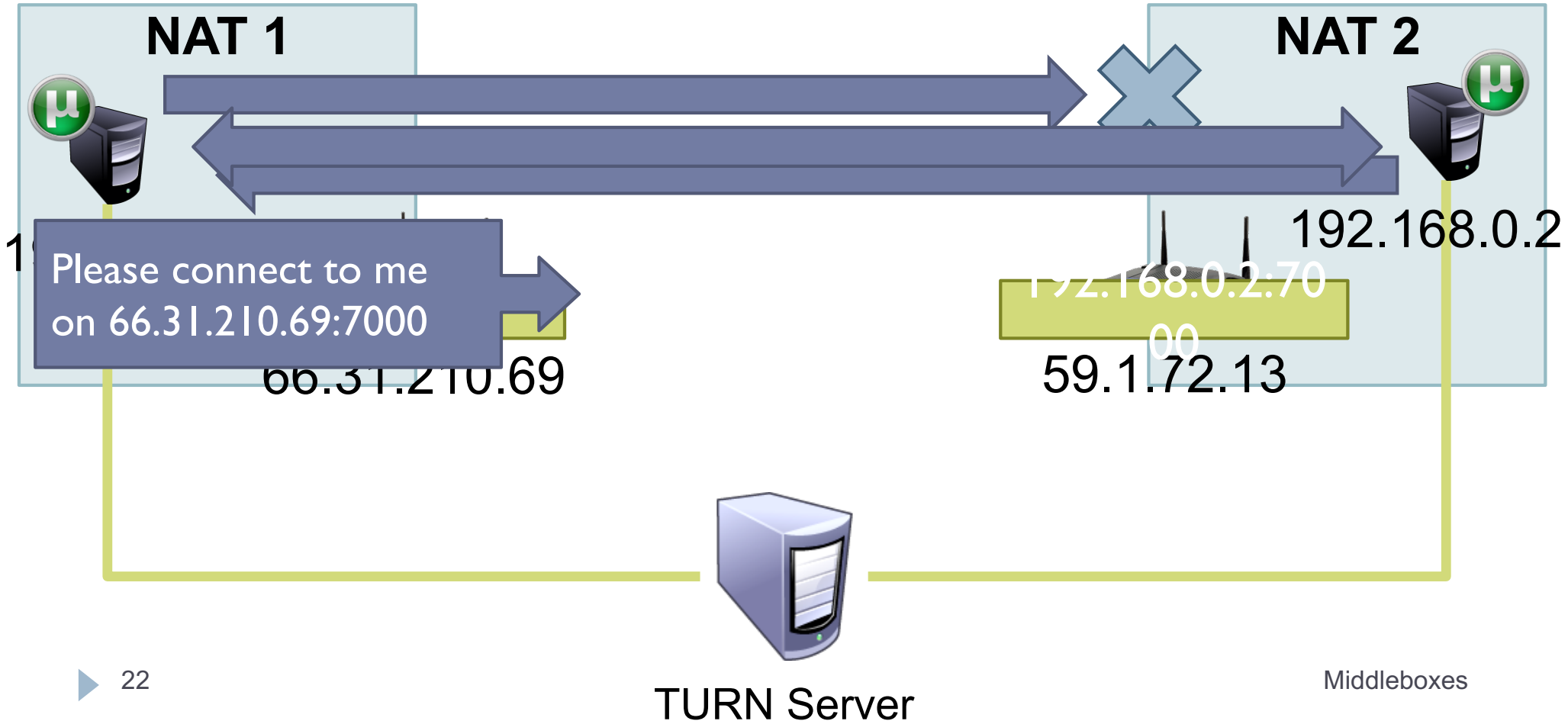
Problems With STUN

- ▶ Only useful in certain situations
 - ▶ One peer is behind a symmetric NAT
 - ▶ Both peers are behind partial NATs
- ▶ Not useful when both peers are fully behind full NATs



TURN

► Traversal **U**sing **R**elays around **N**AT



3: Middleboxes: Firewall. Web caching.
Proxies. Shaping.

Firewall

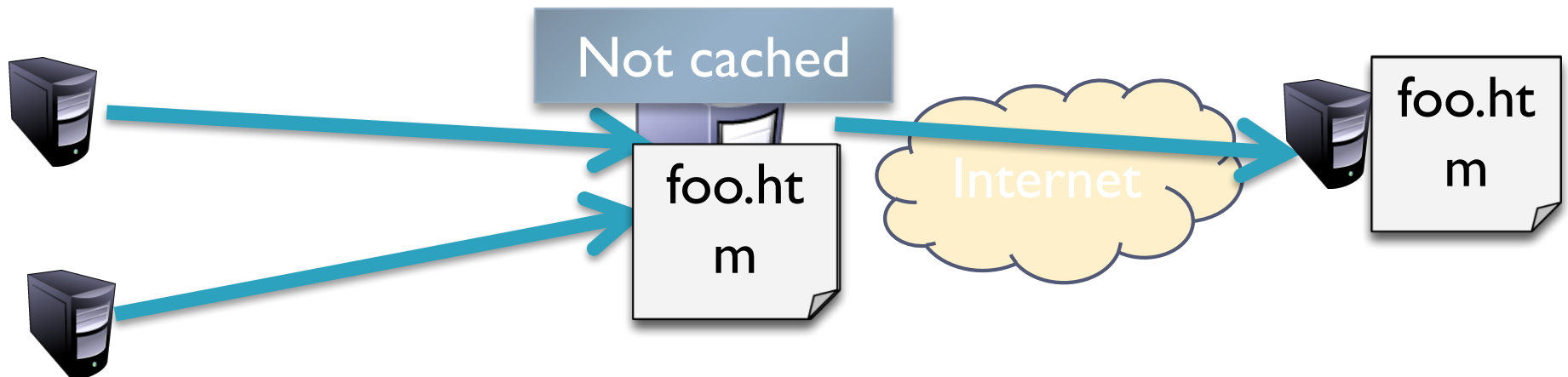
- ▶ A device that blocks traffic according to a set of rules
 - ▶ Why?
 - ▶ Services with vulnerabilities turned on by default
 - ▶ ISP policy forbidding certain traffic due to ToS
- ▶ Typically specified using a 5-tuple
 - ▶ E.g., block outbound SMTP; block inbound SQL server reqs
- ▶ GFC (Great Firewall of China)
 - ▶ Known to block based on IP, filter DNS requests, etc

Web caching

- ▶ ISP installs cache near network edge that caches copies of Web pages
 - ▶ Why?
 - ▶ **Performance:** Content is closer to clients, TCP will perform better with lower RTTs
 - ▶ **Cost:** “free” for the ISP to serve from inside the network
- ▶ Limitations
 - ▶ Much of today’s content is not static (why does this matter?)
 - ▶ Content ownership
 - ▶ Potential privacy issues
 - ▶ Long tail of content popularity

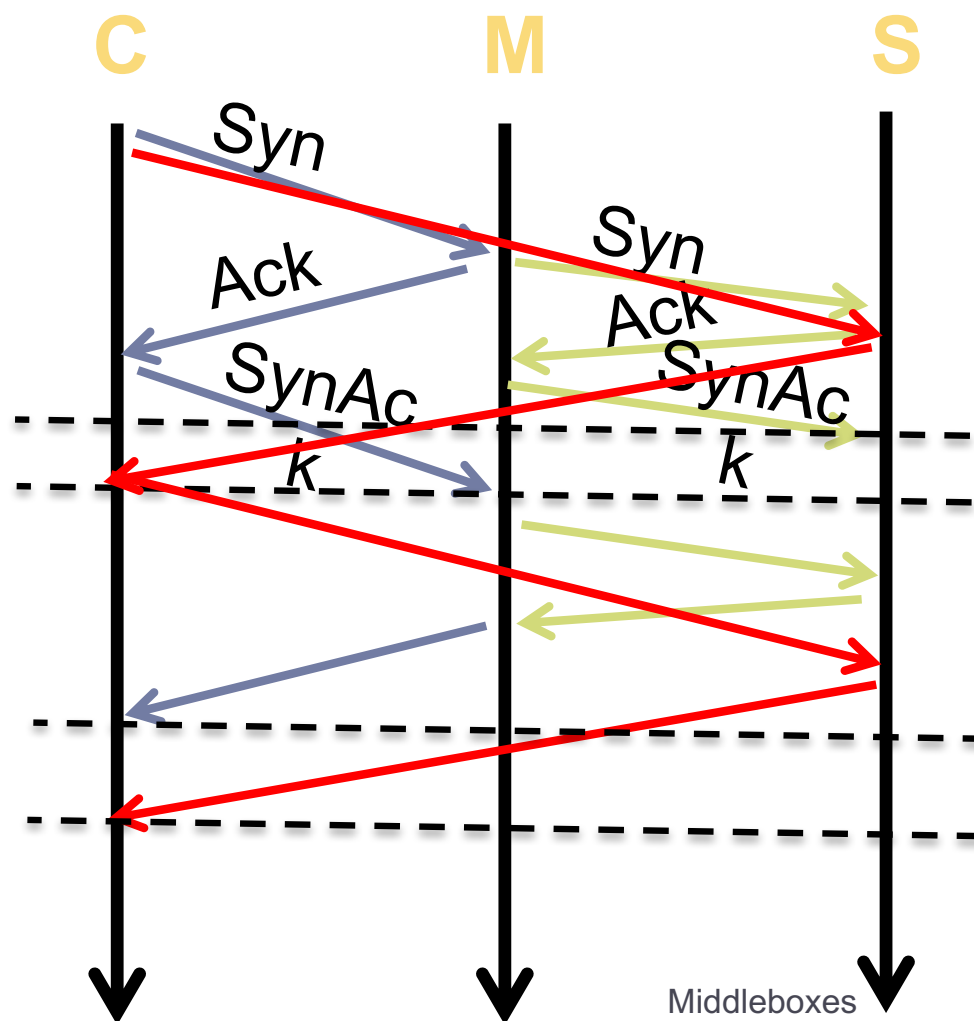
Web caching

- ▶ ISP installs cache near network edge that caches copies of Web pages
 - ▶ Why?
 - ▶ **Performance:** Content is closer to clients, TCP will perform better with lower RTTs
 - ▶ **Cost:** “free” for the ISP to serve from inside the network



Proxying

- ▶ **Non-split connections**
 - ▶ Forward traffic to a different IP/port (like a tunnel)
 - ▶ Can be local or remote
- ▶ **Split connections**
 - ▶ Middlebox maintains two flows:
C-M and M-S
 - ▶ Can be done transparently
 - ▶ How?



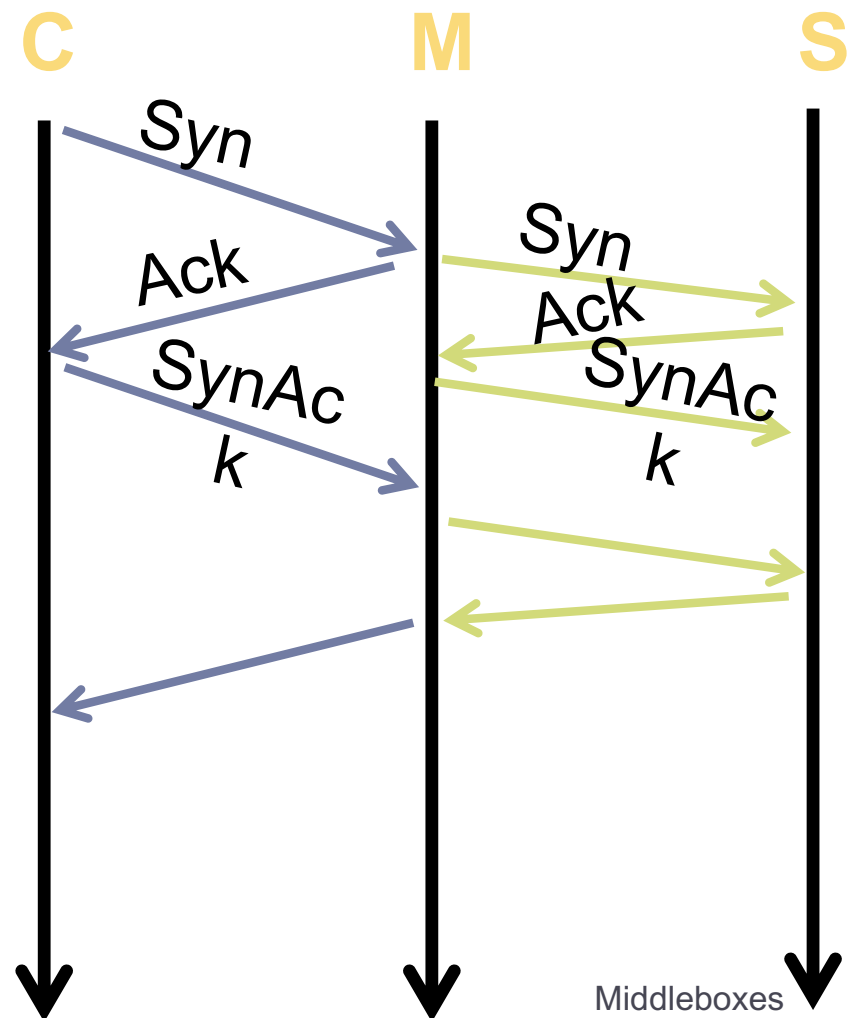
Proxying

► Advantages

- RTT is lower on each end
- Can use different MTUs
- Particularly useful in cell ntwks

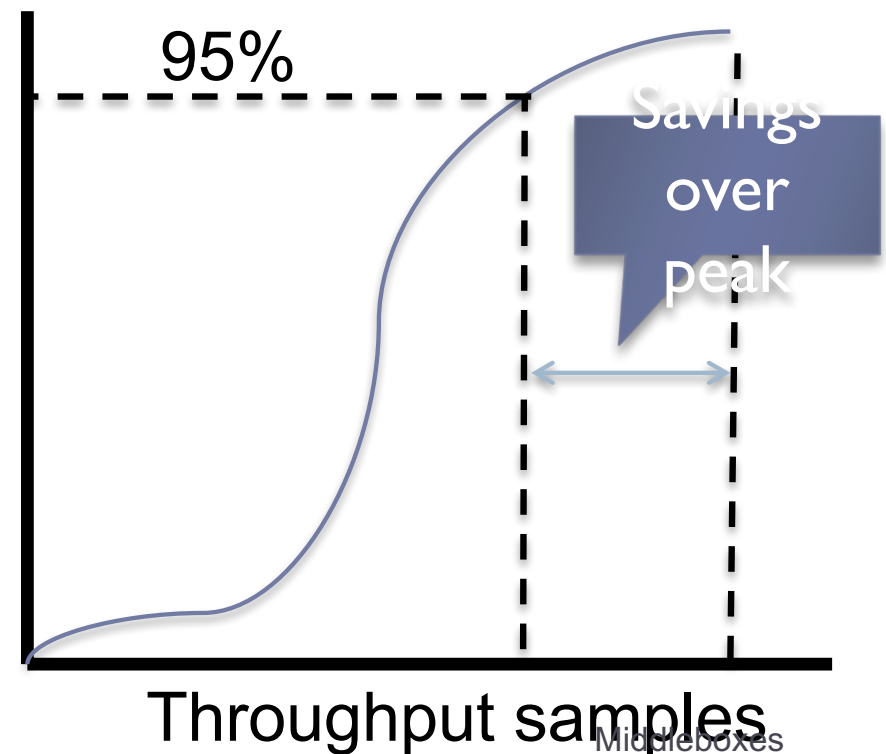
► Disadvantages

- Extra delay can be bad for small flows
- Buffering/state makes it potentially costly



Shaping

- ▶ ISPs are often charged according to 95% model
 - ▶ Internet usage is very “peaky”, e.g., at 5pm, or when House of Cards season 3 is released
- ▶ To control costs, ISPs such as Rogers **shape** client traffic
 - ▶ Time-of day
 - ▶ Traffic type
- ▶ Common implementations
 - ▶ Token Bucket (see next deck)
 - ▶ RSTs



Summary

- ▶ **Middleboxes are pervasive in today's networks**
 - ▶ Security
 - ▶ Performance
 - ▶ Network engineering
- ▶ **Pros**
 - ▶ Allow the network to provide functionality not provided by IP
 - ▶ Can protect the network from client misconfigurations
- ▶ **Cons**
 - ▶ Break the end-to-end model, by operating at layer 3
 - ▶ Can threaten net neutrality
 - ▶ One more point of failure



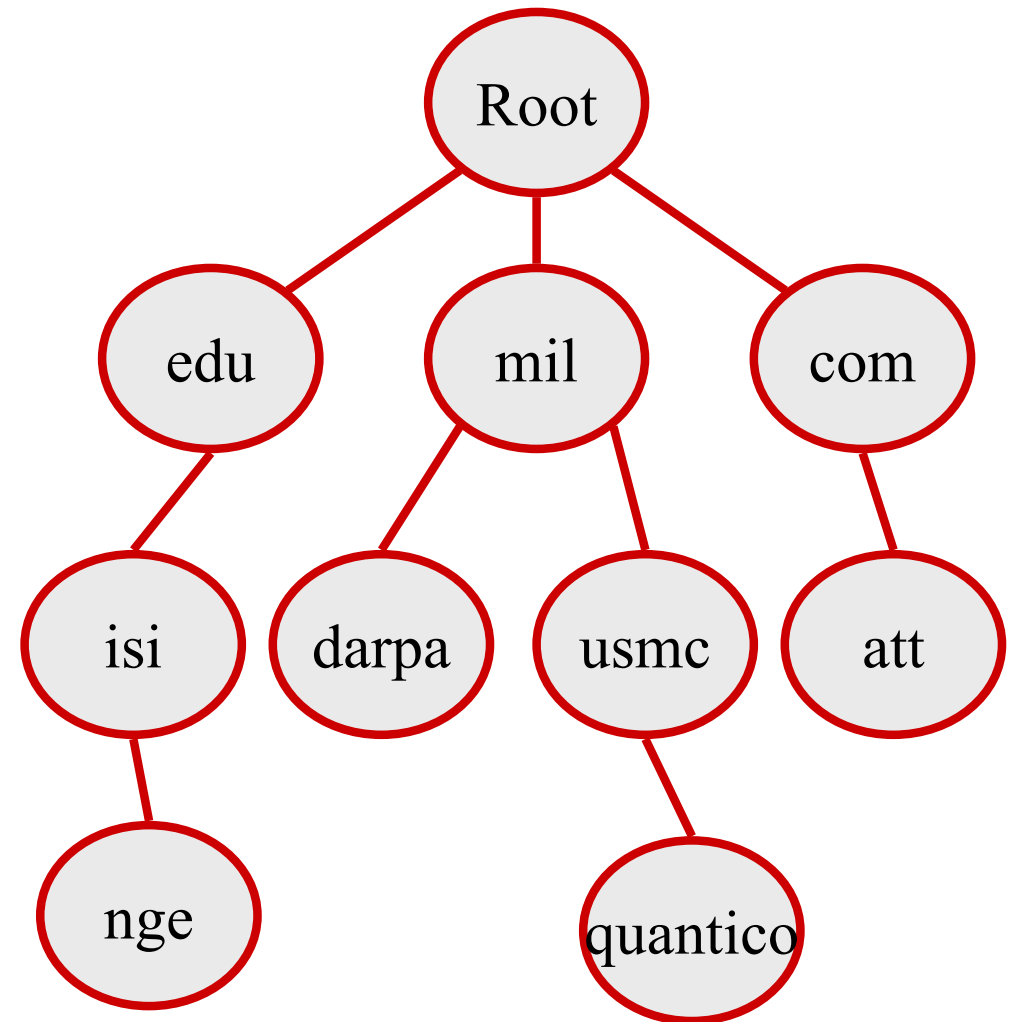
4: DNS Overview

Domain Name System (DNS)

- ▶ People prefer names to identify computers instead of numbers (IP addresses)
- ▶ DNS
 - ▶ Distributed database
 - ▶ Hierarchical name space as opposed to flat space
 - ▶ Maps host names with IP addresses
- ▶ Almost any application uses DNS
- ▶ If DNS is not working many applications will be crippled
- ▶ Uses UDP for communication (port 53), some implementations use TCP

Hierarchical name space

- ▶ Tree structure
 - ▶ Divided into zones
 - ▶ Delegating responsibilities
- ▶ CANN oversees the domain name assignments
- ▶ Top Level Domains (TLDs) are at the top
- ▶ Each Domain Name is a subtree
 - ▶ Maximum tree depth: 128



Server Hierarchy

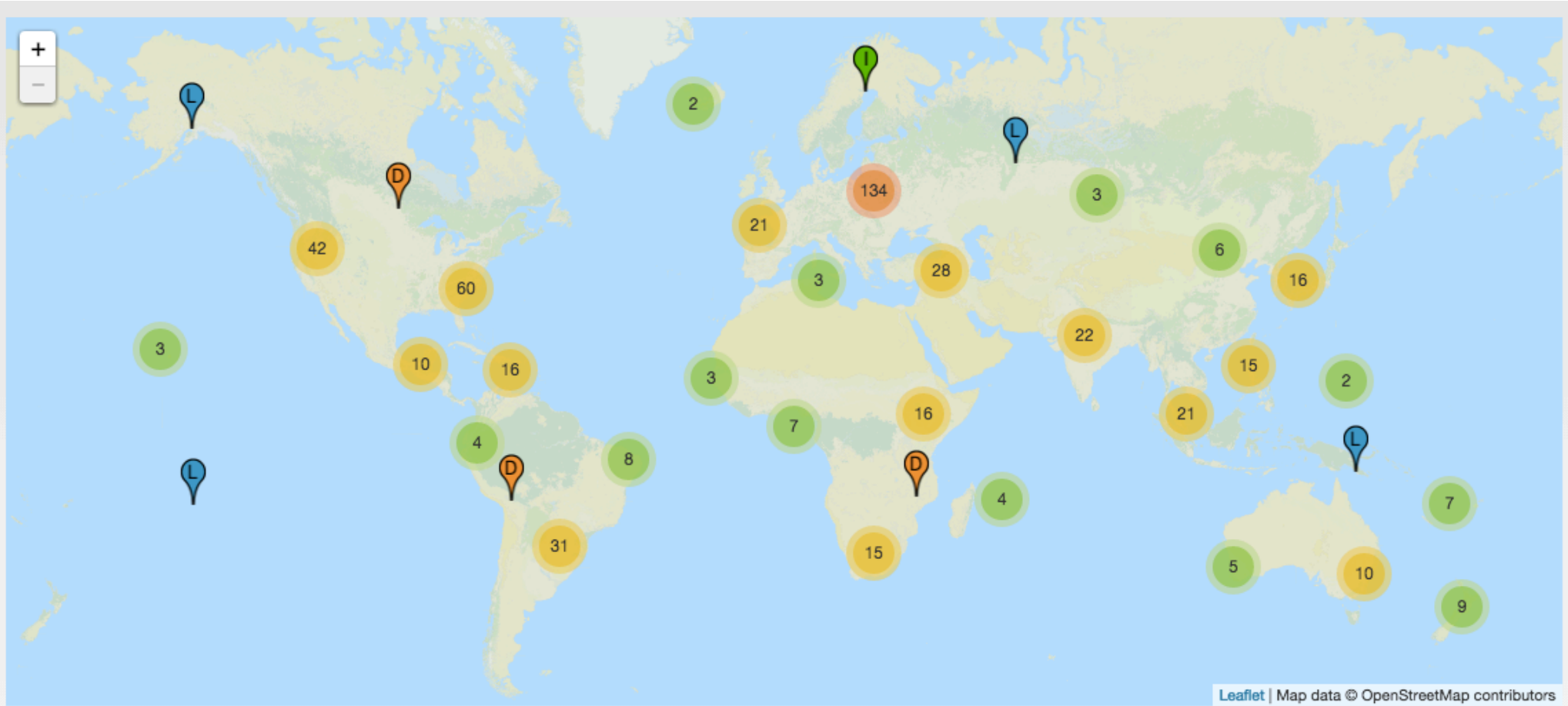
- ▶ **Functions of each DNS server:**
 - ▶ Authority over a portion of the hierarchy
 - ▶ No need to store all DNS names
 - ▶ Store all the records for hosts/domains in its zone
 - ▶ May be replicated for robustness
 - ▶ Know the addresses of the root servers
 - ▶ Resolve queries for unknown names
- ▶ **Root servers know about all TLDs**
 - ▶ The buck stops at the root servers

Root name servers

- ▶ **Responsible for the Root Zone File**
 - ▶ Lists the TLDs and who controls them
 - ▶ ~272KB in size
- ▶ 13 logical root servers , labeled A→M
 - ▶ operated by 12 independent organizations
 - ▶ 6 are anycasted, i.e. they are globally replicated
- ▶ Top Level Domain (TLD) operate “.com”, “.edu”, etc
- ▶ Contacted when names cannot be resolved
 - ▶ In practice, most systems cache this information

<http://www.root-servers.org>

Map of the roots (www.root-servers.org)



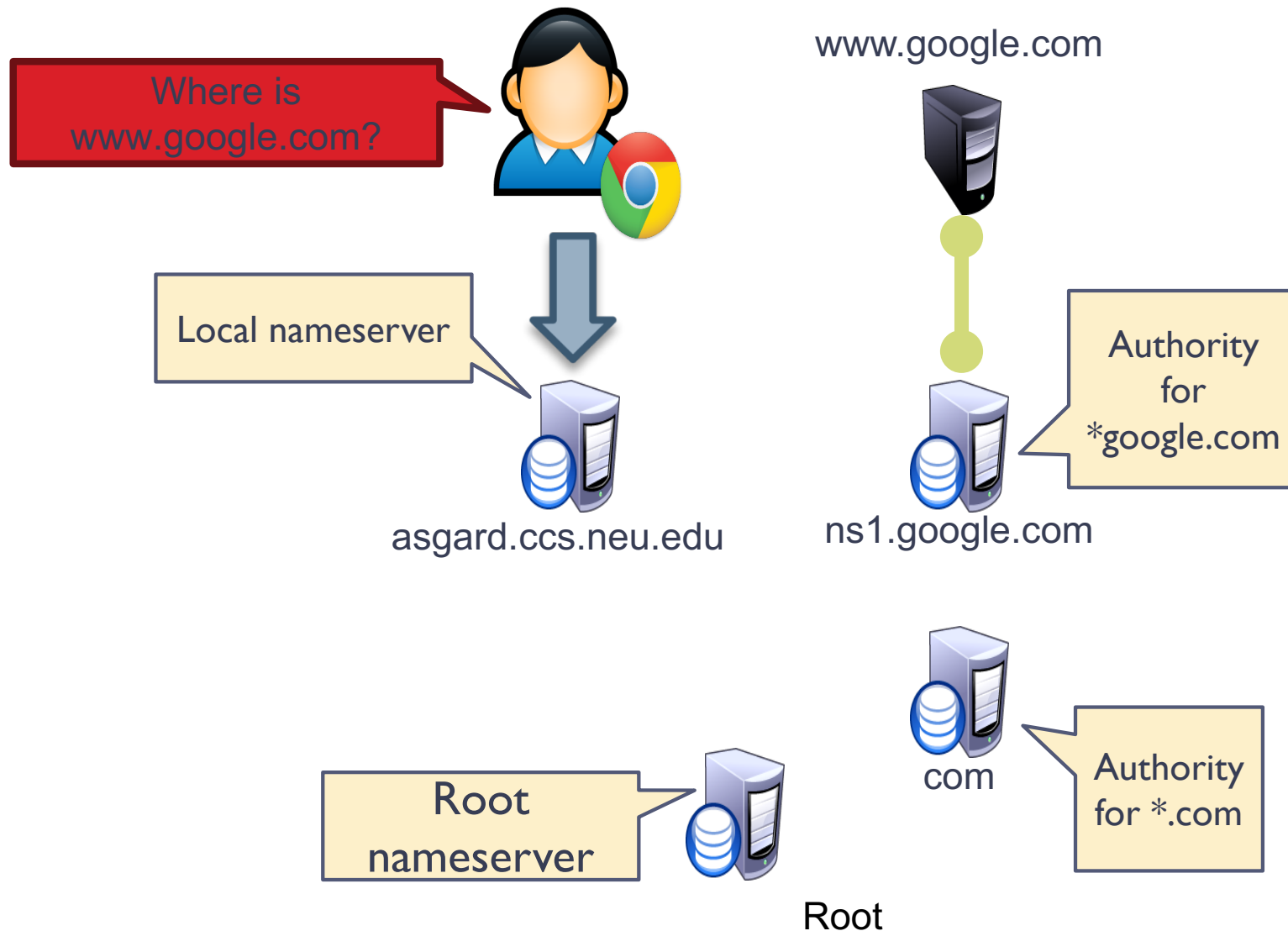
Domain Name Servers

- ▶ **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.

Local and Authoritative name server

- ▶ Resolver: client part that asks the questions about hostnames
- ▶ Name server: a server that can answer DNS queries
 - ▶ Local names server: handles queries on behalf of clients
 - ▶ Authoritative nameservers: know the zone mappings for a subset of the hierarchy; **Information needs to be updated when host info changes in the zone**
- ▶ Name servers cache answers:
 - ▶ time for caching depends on the TTL (set by the administrator of the DNS server handing out the response), from seconds to days or even weeks.

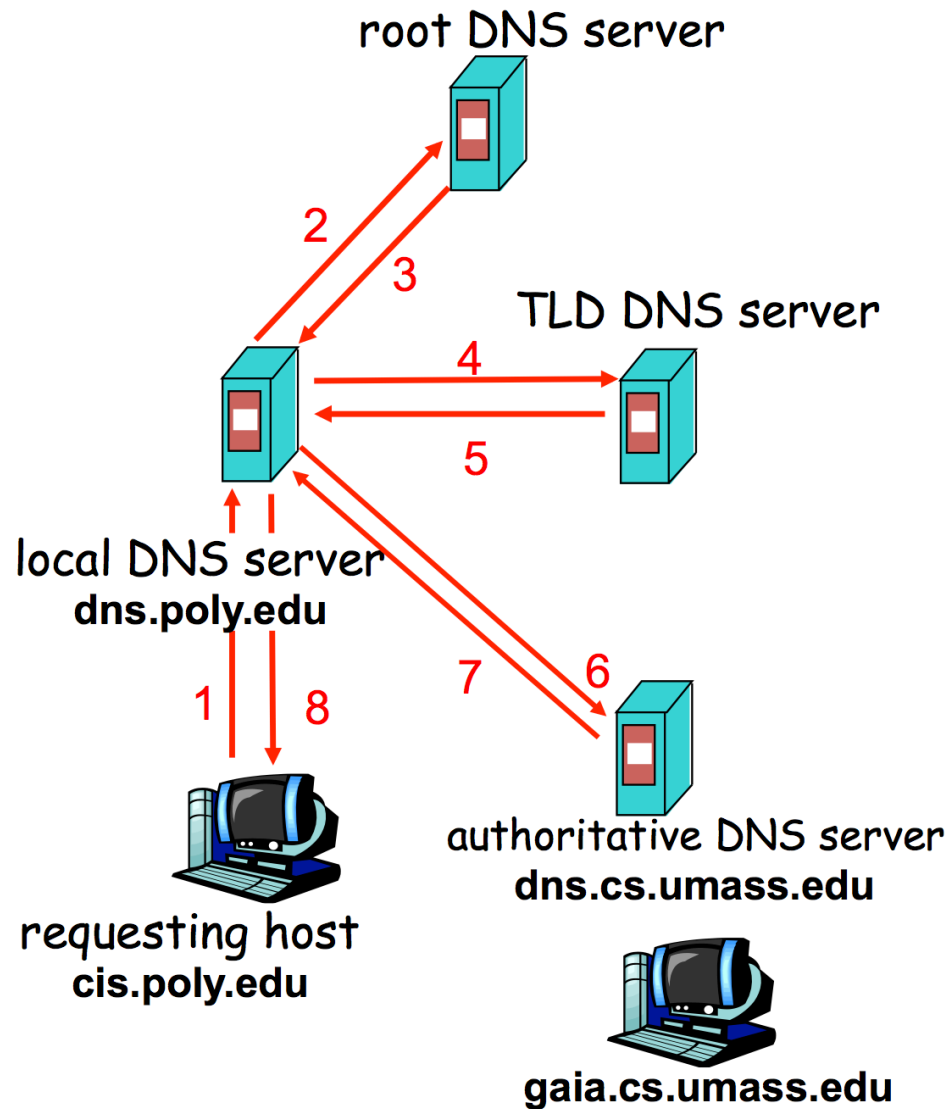
Local vs Authoritative Nameserver



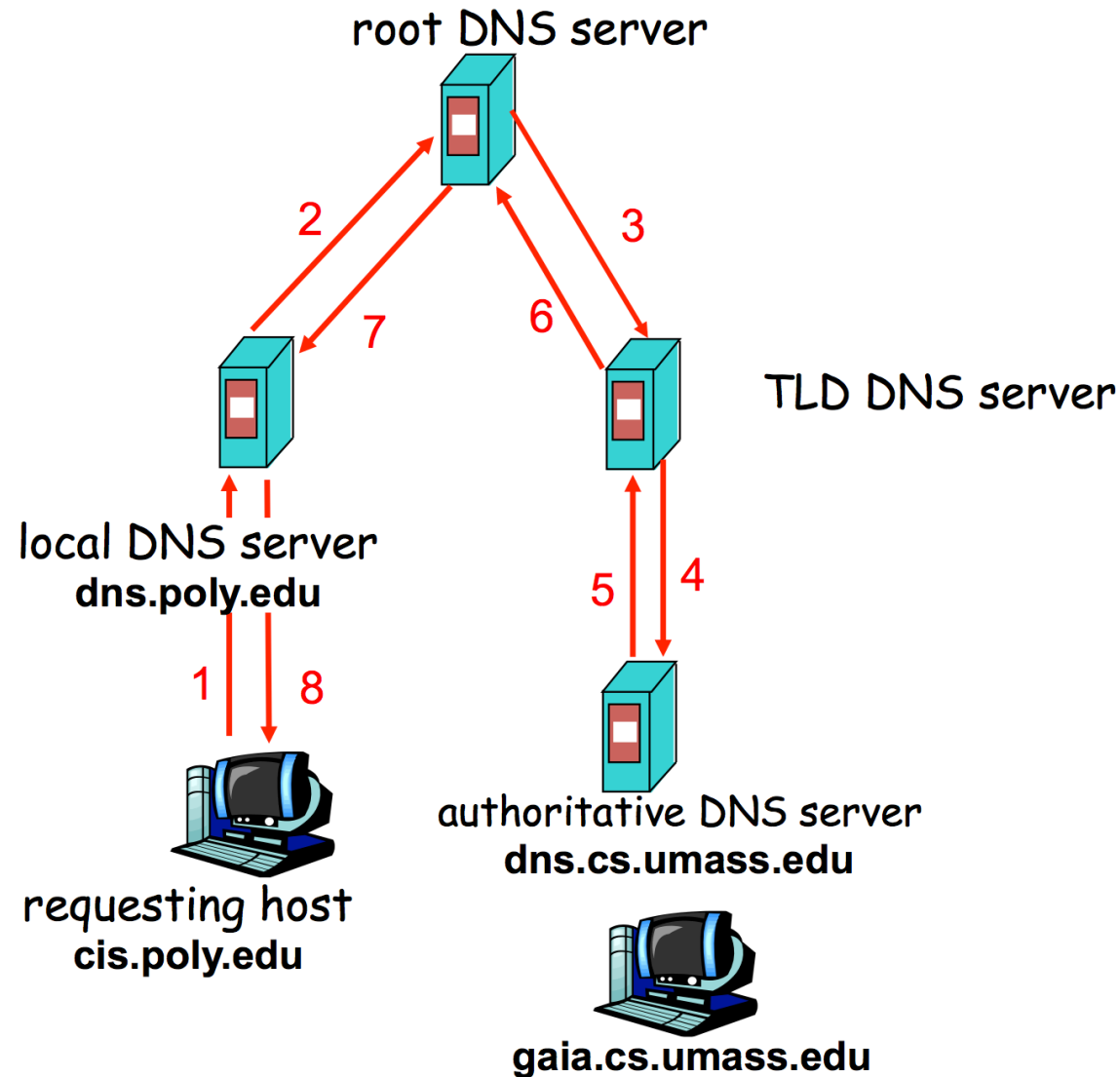
DNS Resolving

- ▶ **Every host knows a local nameserver**
 - ▶ Sends queries to it and expects name to be resolved
- ▶ **If the local nameserver knows the name**
 - ▶ Nameserver is also the authoritative server for that name
 - ▶ Nameserver has cached the record for that name and can answers immediately
- ▶ **Otherwise, local nameserver forwards query into hierarchy searching for the authoritative name server**
 - ▶ Every local nameserver knows the root servers
 - ▶ Use cache to skip steps if possible
 - ▶ e.g. skip the root and go directly to .edu if the .edu zone file is cached

DNS resolving: iterative



DNS resolving: recursive



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

DNS resource records

- ▶ DNS queries:
 - ▶ two fields: **name** and **type**
- ▶ Resource record is the response to a query
 - ▶ Four fields: (**name**, **value**, **type**, TTL)
 - ▶ There may be multiple records returned for one query
- ▶ What do the **name** and **value** mean?
 - ▶ Depends on the **type** of query and response

DNS types

- ▶ Type = A / AAAA
 - ▶ Name = domain name
 - ▶ Value = IP address
 - ▶ A is IPv4, AAAA is IPv6
- ▶ Type = NS
 - ▶ Name = partial domain
 - ▶ Value = name of DNS server for this domain
 - ▶ “Go send your query to this other server”

Query

Name:
www.ccs.neu.edu
Type: A

Resp.

Name:
www.ccs.neu.edu
Value: 129.10.116.81

Query

Name: ccs.neu.edu
Type: NS

Resp.

Name: ccs.neu.edu
Value: 129.10.116.51

DNS types (cont.)

▶ Type = CNAME

- ▶ Name = hostname
- ▶ Value = canonical hostname
- ▶ Useful for aliasing

Query

Name: foo.mysite.com
Type: CNAME

Resp.

Name: foo.mysite.com
Value: bar.mysite.com

▶ Type = MX

- ▶ Name = domain in email address
- ▶ Value = canonical name of mail server

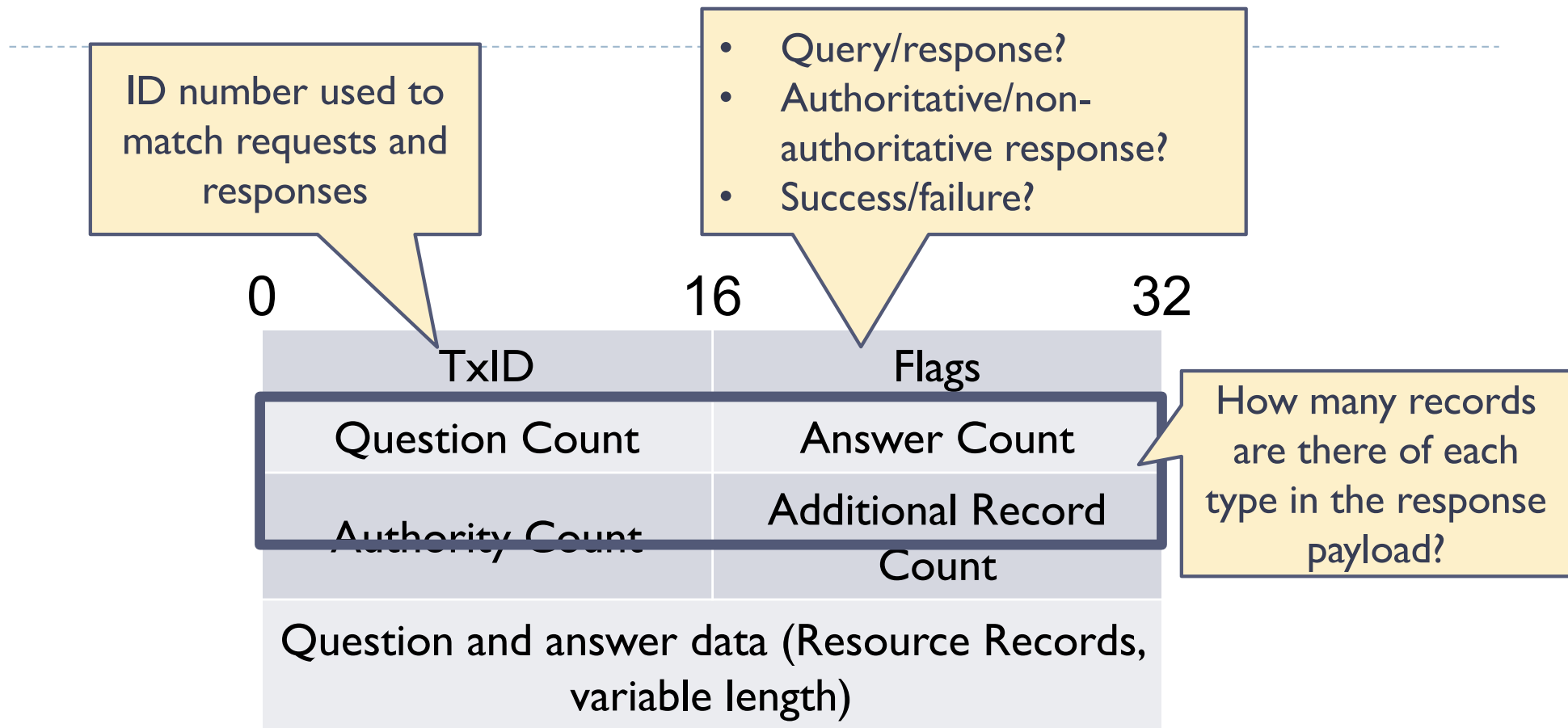
Query

Name: ccs.neu.edu
Type: MX

Resp.

Name: ccs.neu.edu
Value: amber.ccs.neu.edu

DNS packet format

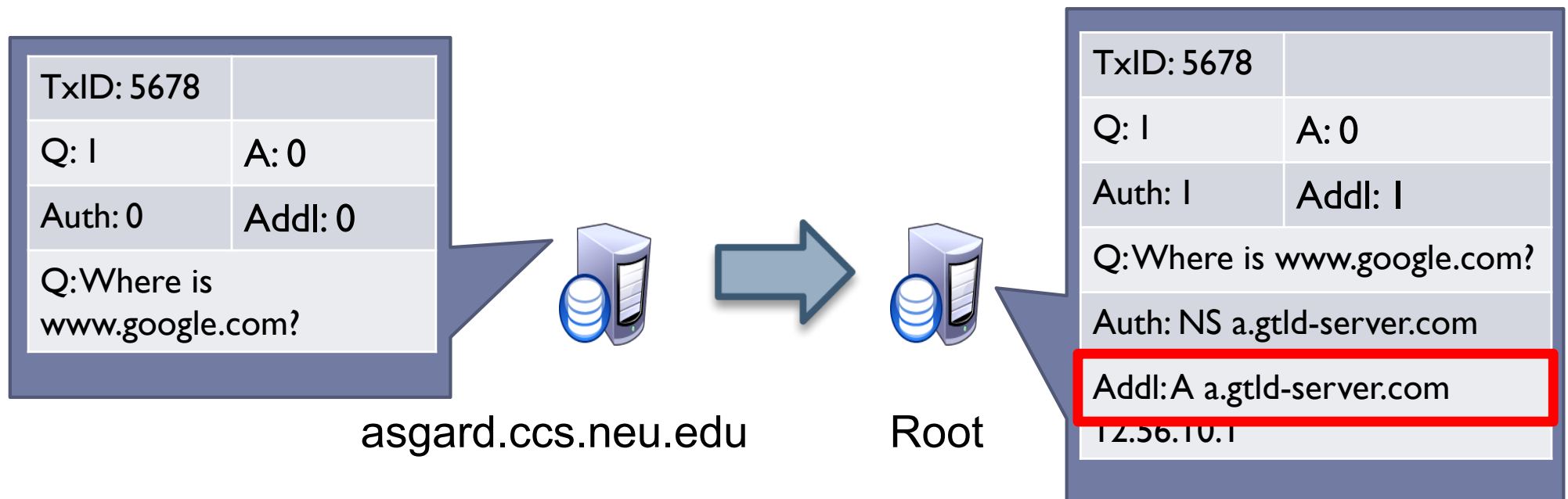


DNS is a UDP-based protocol on port 53

- TxIDs are needed to correlate requests and responses
- Serves as authentication for responses

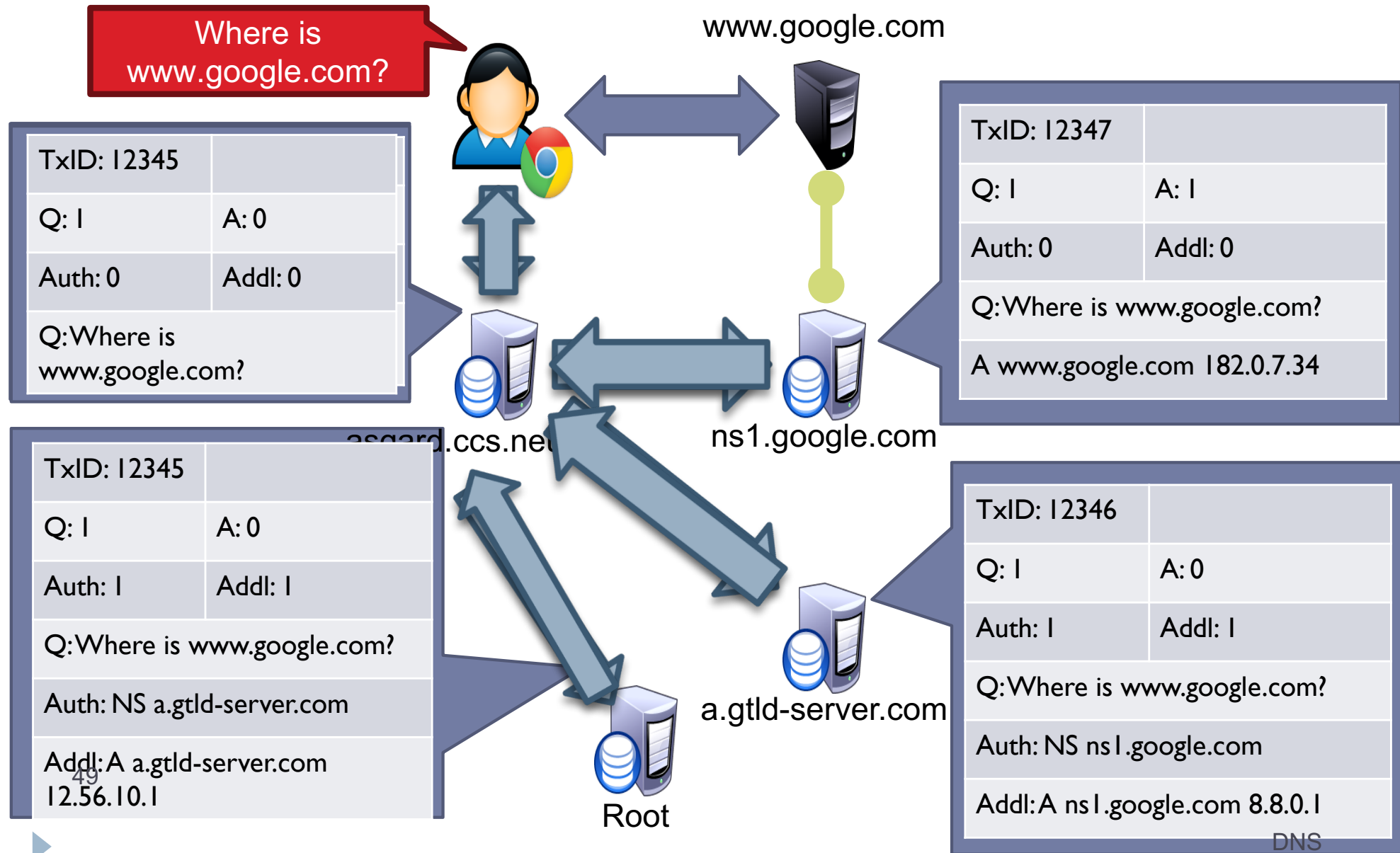
Glue records

- ▶ DNS responses may contain more than a single answer
- ▶ Example: resolving cyclic dependency



- Known as **glue records**
- Additional responses can contain any type of record (i.e.A, NS, etc.)

DNS query, revisited



NXDOMAIN

- ▶ If the domain name could not have been resolved, then the name server returns NXDOMAIN.
- ▶ Some ISPs ``hijacked'' such non-existent domains to make money and collect personal data by redirecting the user to some advertising web page.
- ▶ This practice is in contradiction with the RFC standard for DNS (NXDOMAIN) responses.

2: Attacks against DNS

Inherent DNS vulnerabilities

- ▶ Users/hosts typically trust the host-address mapping provided by DNS
 - ▶ What bad things can happen with wrong DNS info?
- ▶ DNS servers loaded so they can not answer the queries
- ▶ DNS resolvers trust responses received after sending out queries.
 - ▶ How to attack?
- ▶ Obvious problem
 - ▶ No authentication for DNS responses

DNS denial of service

- ▶ **DNS** servers bombarded with requests
 - ▶ **Defective implementations** RFC1918 (private addresses) that propagate requests/updates that were not supposed to happen (blackhole servers now collect and drop this traffic)
 - ▶ **Malicious attacks on the backbone**: Oct. 2002, DDoS, 9 of the root servers were affected (about 1 hour, ICMP flooding); Second attacks in Feb. 2007, 2 of the root servers were affected
 - ▶ Can target particular servers: for example something like 65Gbps DDoS is a big attack (Cloudflare)
 - ▶ Use Botnets to generate the attack traffic

Mitigating denial of service

- ▶ Root servers are deployed as clusters of machines
- ▶ Use load balancing
- ▶ Queries rate controlled, each source address is limited to a 10KBits/sec and queue size of 3 packets.

More DNS Attacks

- ▶ **DNS Spoofing:**
 - ▶ Guessing DNS queries Ids (man in the middle)
 - ▶ Compromise the DNS servers itself
- ▶ **Cache Poisoning:** False IP with a high TTL, which the DNS server will cache for a long time
- ▶ **Email Spoofing:** Registration with ICANN often done via email and authenticated by the email address. Return addresses can be falsified
- ▶ **Mis-configuration:** Administrator enters the DNS information incorrectly

A DNS Packet

- ▶ It's a UDP packet
- ▶ Query ID (16 bits): identifies each request
- ▶ Source/Destination IPs: machines that sent and should receive the packet.
- ▶ Source/Destination Ports:
 - ▶ DNS servers listen on port 53/udp for queries from the outside world;
 - ▶ The source port varies, 53, fixed port chosen at random by the operating system, or random port that changes every time.

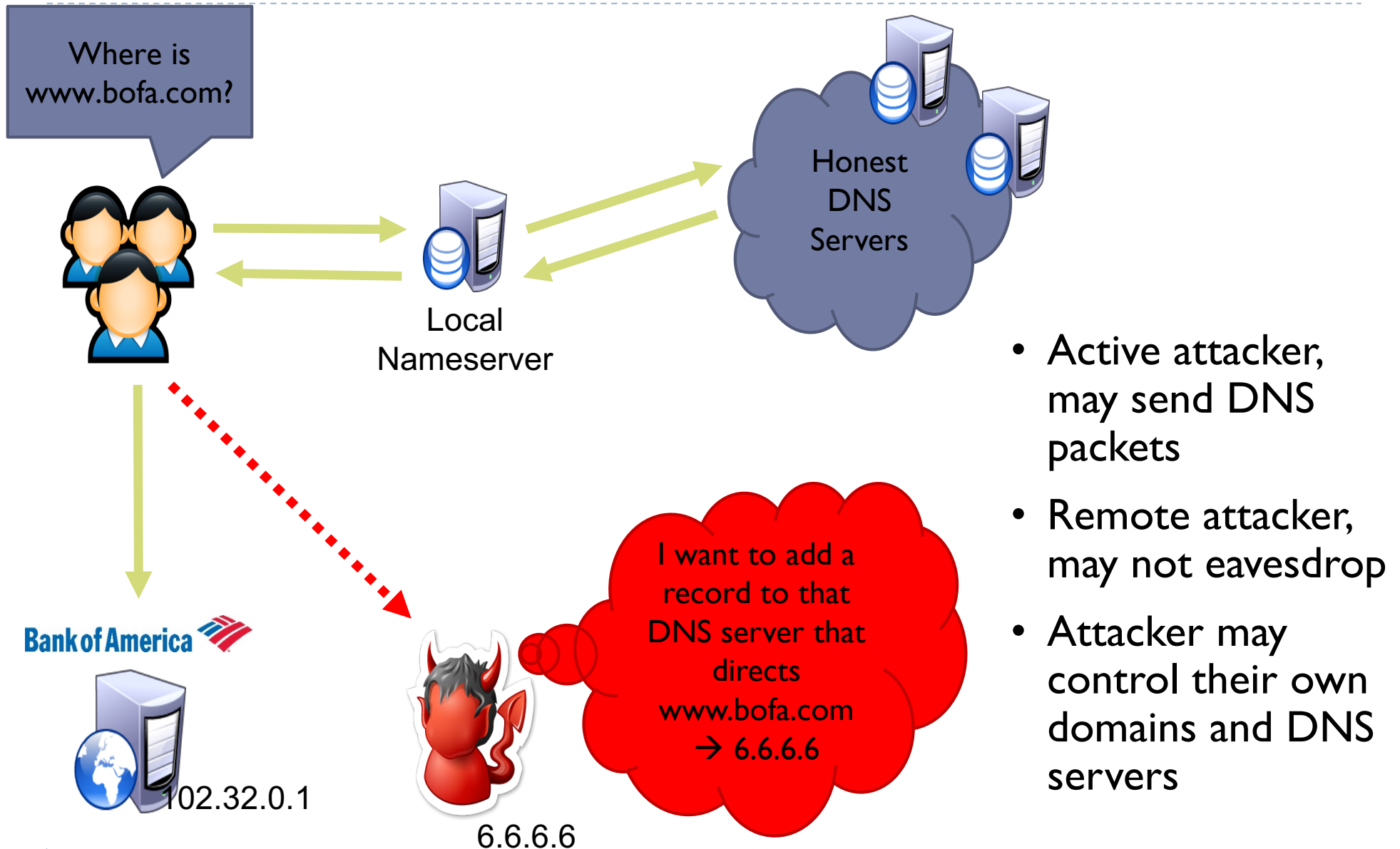
User Side Attack - Pharming

- ▶ **Exploit DNS poisoning attack**
 - ▶ Change IP addresses to redirect URLs to fraudulent sites
 - ▶ Potentially more dangerous than phishing attacks
- ▶ **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

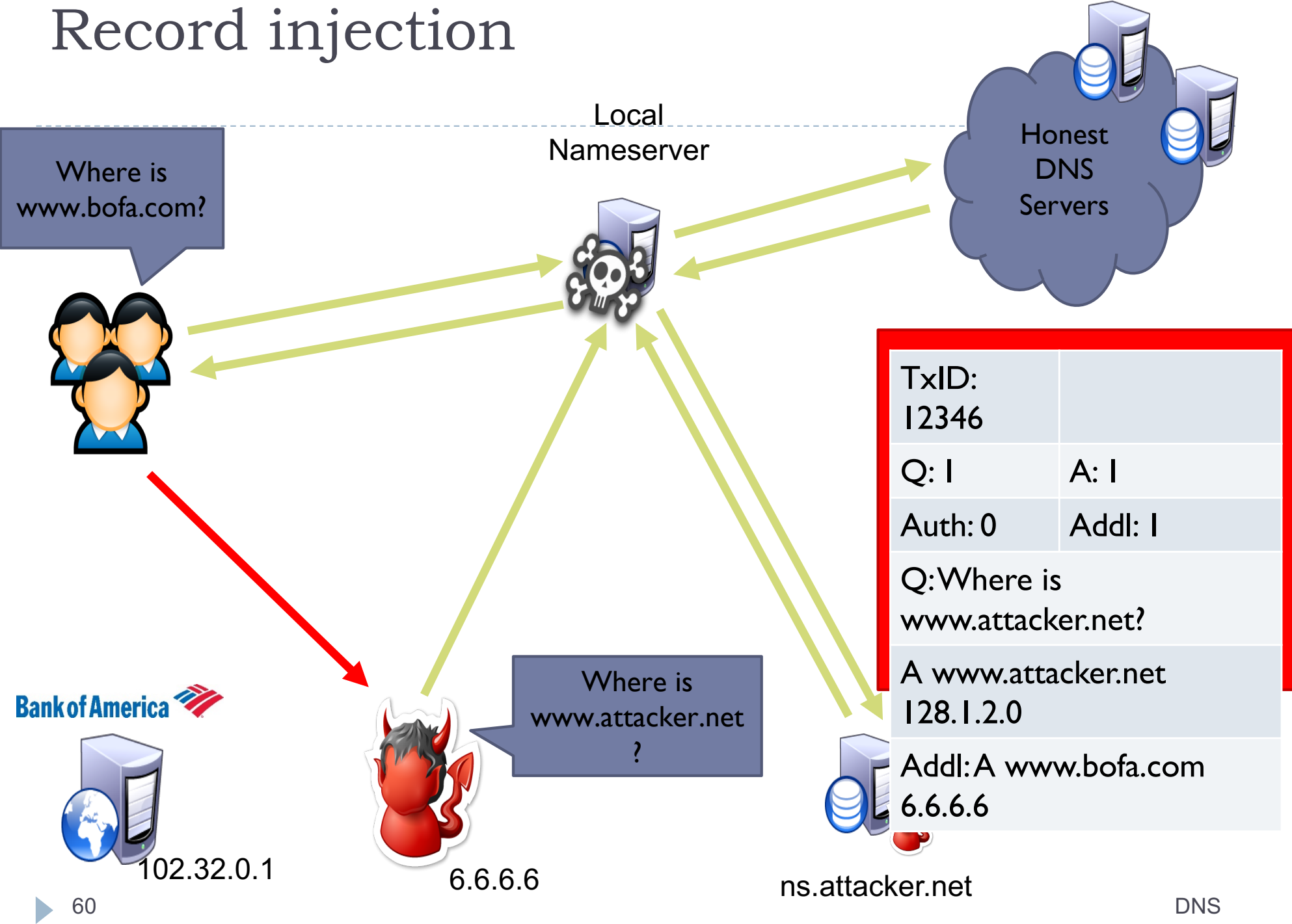
Cache poisoning

- ▶ **Three types of attacks**
 - ▶ Old school attack: record injection
 - ▶ Somewhat old school attack: response spoofing
 - ▶ New, deadly attack: The Kaminsky Attack

Threat model and attacker goals



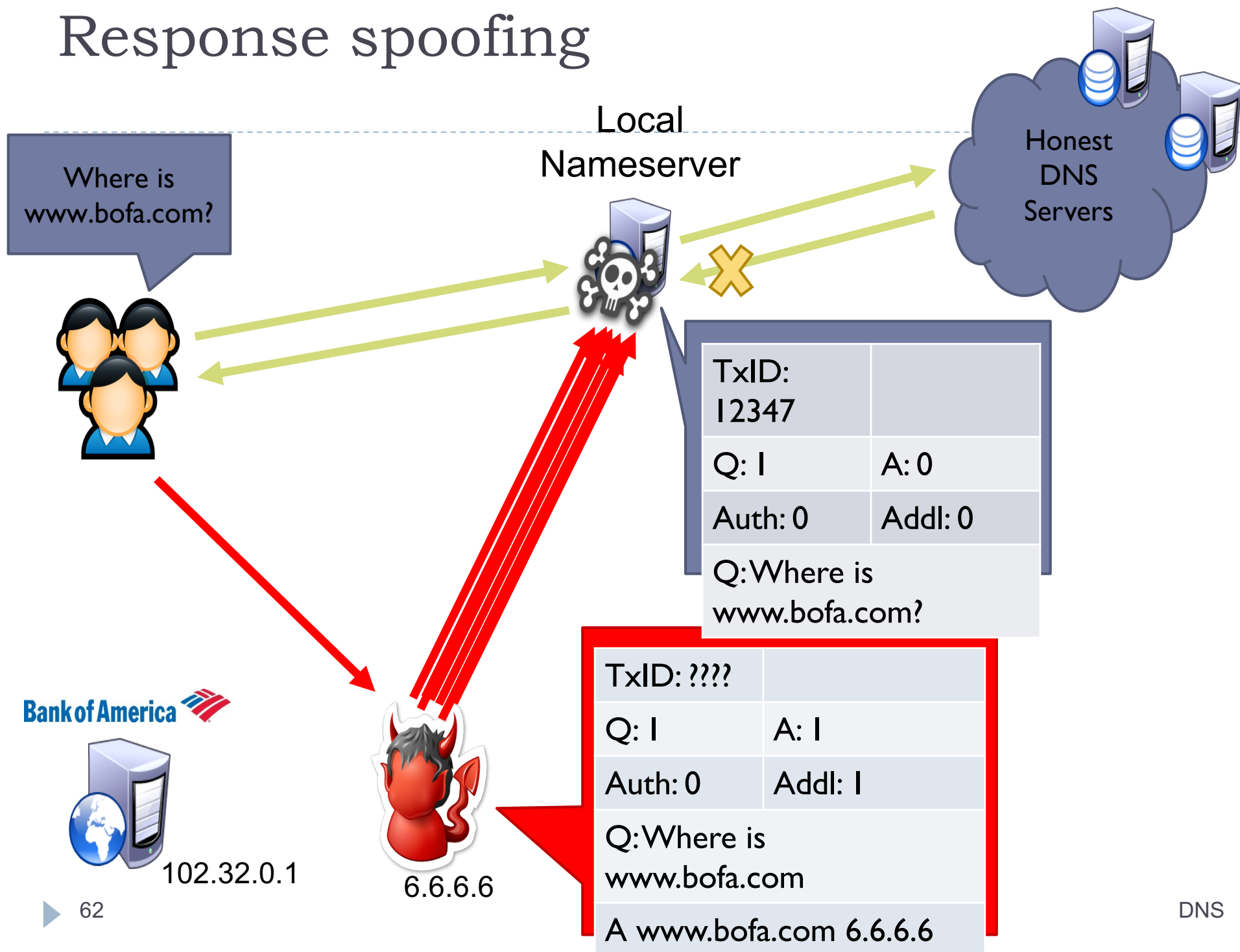
Record injection



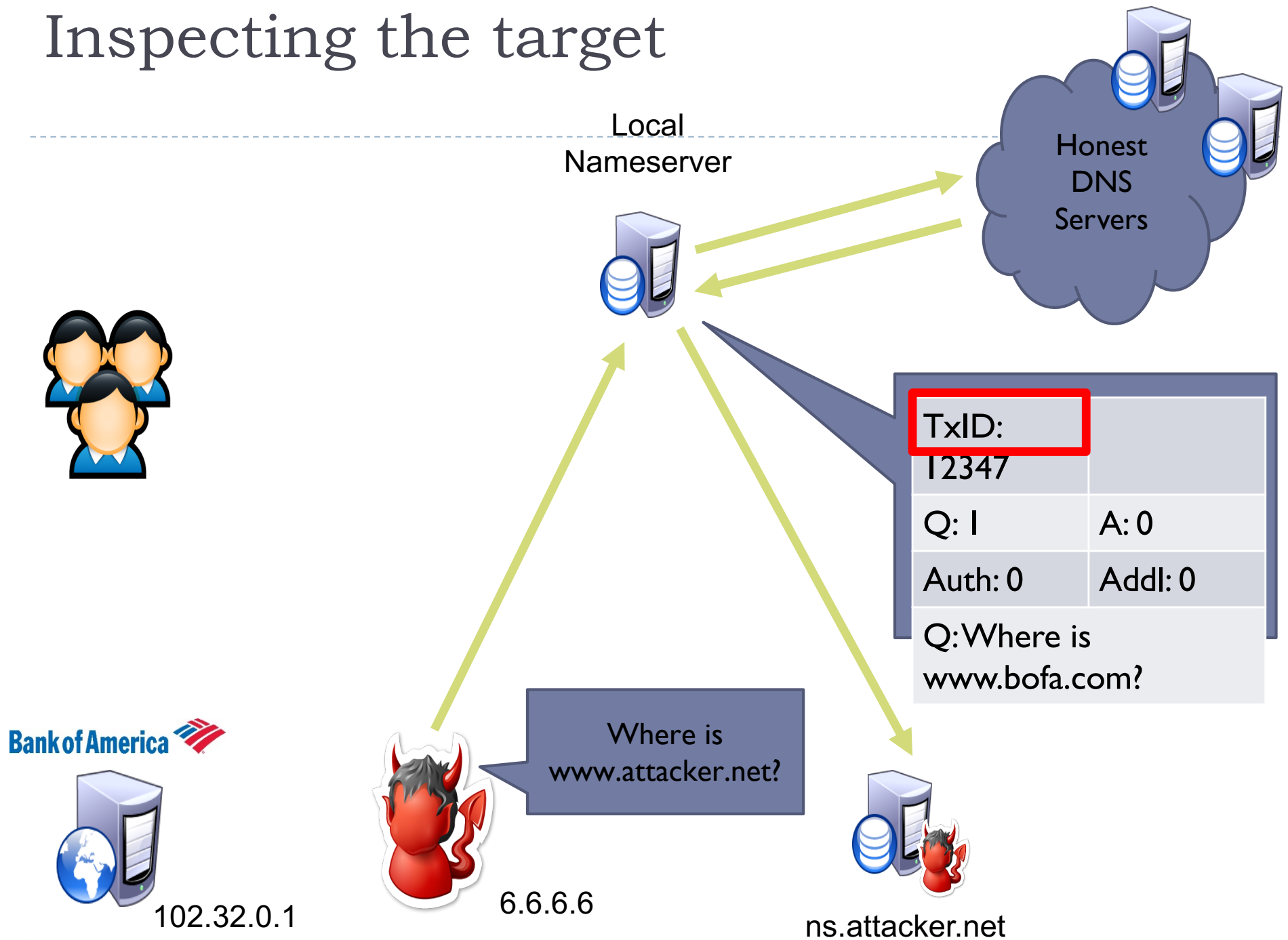
Bailiwick checking

- ▶ Record injection attacks no longer work in practice
- ▶ The bailiwick system prevents foo.com from declaring anything about “com”, or some other new TLD, or www.google.com
- ▶ Using the bailiwicks rules
 - ▶ The root servers can return any record
 - ▶ The com servers can return any record for com
 - ▶ The google.com servers can return any record for google.com
- ▶ All modern DNS servers implement **bailiwick checking**

Response spoofing



Inspecting the target



Implementing response spoofing

- ▶ What info does the attacker need to spoof a DNS response?
 - ▶ IP address of the target nameserver and true authoritative nameserver
 - ▶ Easy, both pieces of info are readily available
 - ▶ Source port used by the authoritative nameserver
 - ▶ Easy, it must be 53
 - ▶ The question in the query
 - ▶ Easy, the attacker can choose the targeted domain name
 - ▶ Response port used by the target when they made the request
 - ▶ TxID in the query

Challenges of response spoofing

- ▶ Attacker must infer the response port of the target nameserver and TxID
- ▶ Attacker's response must outrace the legitimate response
- ▶ The attack must be executed after the target nameserver is queried for a domain that is not in the cache
 - ▶ If the target domain name is already cached, no queries will be sent
 - ▶ The attacker can send the initial query to the nameserver, but if the attack fails the legitimate response will be cached until the TTL expires
- ▶ If the attack is successful, the record for a single domain is poisoned

DNS response spoofing: incremental TxID

- ▶ Old DNS servers used one port for all queries and incremented TxID monotonically
 - ▶ Attacker can query the target DNS server for a domain they control and observe the query at their own DNS server
 - ▶ The query reveals the port used by the target, as well as the approximate TxID
 - ▶ CERT reported in 1997 that BIND uses sequential transaction ID and is easily predicted
 - ▶ fixed by using random transaction IDs

DNS cache poisoning (Schuba and Spafford in 1993): exploits sequential TxID

- ▶ **DNS resource records (see RFC 1034)**
 - ▶ An “A” record supplies a host IP address
 - ▶ A “NS” record supplies name server for domain
- ▶ **First, guess TxID:**
 - ▶ Ask (dns.target.com) for www.evil.org
 - ▶ Request is sent to dns.evil.org (get TxID).
- ▶ **Second, attack:**
 - ▶ Ask (dns.target.com) for www.yahoo.com
 - ▶ Give responses from “dns.yahoo.com” to our chosen IP.

Non-cryptographic defenses to cache poisoning

- ▶ TxID is sequential: attacker floods with a window of future sequences (remember that everything is sent in clear)
- ▶ Proposed solution: **randomize the TxID**. It's harder for the attacker to guess what the TxID will be. (needs to inject thousands of packets)

DNS cache poisoning – Birthday attack

- ▶ Improve the chance of responding before the real nameserver (discovered by Vagner Sacramento in 2002)
 - ▶ Have many (say hundreds of) clients send the same DNS request to the name server
 - ▶ Each generates a query
 - ▶ Send hundreds of reply with random transaction IDs at the same time
 - ▶ Due to the Birthday Paradox, the success probability can be close to 1
 - ▶ 300 will give you 50%.
 - ▶ 700 will give you 1.07%

DNS response spoofing so far

- ▶ Early versions of DNS servers deterministically incremented the ID field
- ▶ Vulnerabilities were discovered in the random ID generation
 - ▶ Weak random number generator
 - ▶ The attacker is able to predict the ID if knowing several IDs in previous transactions
- ▶ Birthday attack
 - ▶ 16- bit (only 65,536 options).
 - ▶ Force the resolver to send many identical queries, with different IDs, at the same time
 - ▶ Increase the probability of making a correct guess

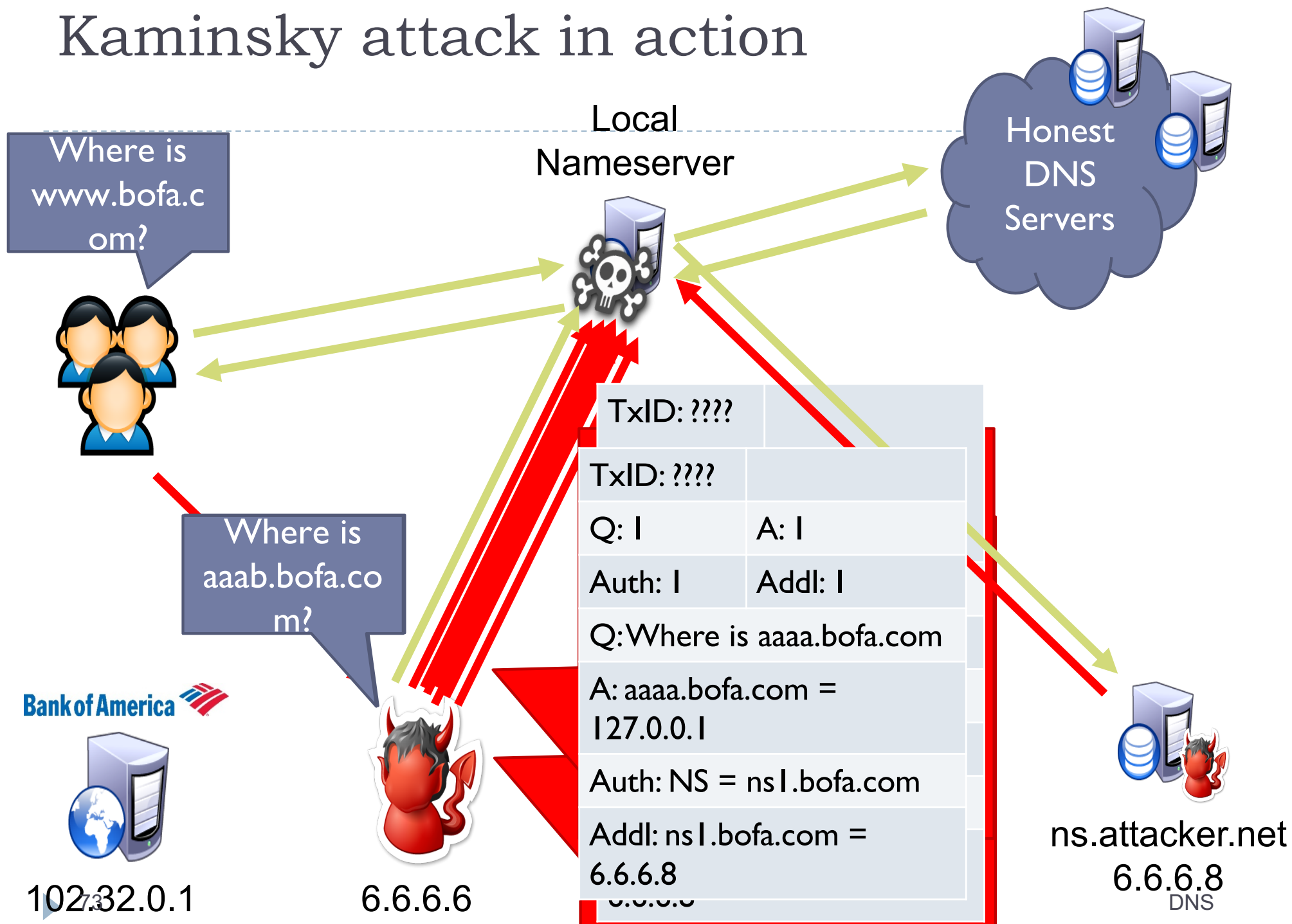
DNS cache poisoning - Kaminsky

- ▶ **Kaminsky Attack**
 - ▶ Big security news in summer of 2008
 - ▶ DNS servers worldwide were quickly patched to defend against the attack
- ▶ In previous attacks, when the attacker loses the race, the record is cached, with a TTL.
 - ▶ Before TTL expires, no attack can be carried out
 - ▶ Poisoning address for google.com in a DNS server is not easy.

Kaminsky attack

- ▶ Poisons glue records rather than A records
 - ▶ Attacker repeatedly makes queries for non-existent subdomains of the target domain
 - ▶ Since these subdomains do not exist, they are guaranteed to not be in the target nameservers cache
 - ▶ Attacker then attempts to spoof a response with a poison glue record
 - ▶ The attacker can attempt the attack an infinite number of times until success
- ▶ On success, entire zone is poisoned, rather than a single domain name

Kaminsky attack in action



What is new in the Kaminsky attack?

- ▶ The bad guy does not need to wait to try again
- ▶ Can start anytime; no waiting for old good cached entries to expire
- ▶ No “wait penalty” for racing failure
- ▶ The attack is only bandwidth limited

Mitigating the Kaminsky attack

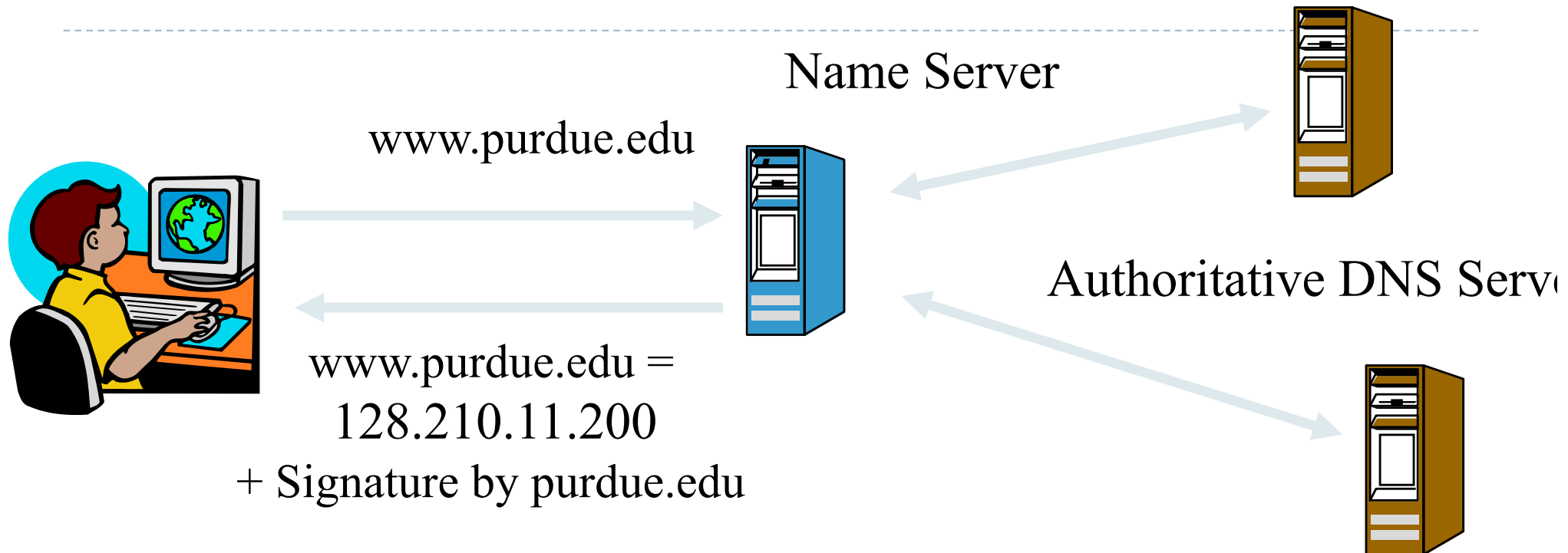
- ▶ The Kaminsky attack relies on fundamental properties of the DNS protocols
 - ▶ the ability to respond with NS records and glue to any query
 - ▶ the functionality is essential for DNS, it cannot be disabled
- ▶ How do you mitigate the Kaminsky attack?
 - ▶ Make it harder to spoof DNS responses by randomizing also the port
 - ▶ All modern DNS servers randomize the TxID and query port for every request
 - ▶ 2^{16} TxIDs * 2^{32} query ports = 2^{48} messages needed to spoof successful
- ▶ Despite this mitigation, almost all existing DNS servers
- ▶ ⁷⁵ are still fundamentally vulnerable to spoofing attacks

3: DNSSEC

DNSSEC

- ▶ Proposed solution: addressing authentication and integrity (digital signatures)
- ▶ Each DNS zone signs its data using its private key (signing can be done offline, in advance)
- ▶ Query for a record will return the requested resource and a digital signature of the requested resource record set
- ▶ Resolver will authenticate the response using the corresponding public key of the zone

Secure DNS



What are the Issues?

- ▶ How to obtain the public key to verify the digital signature (chicken-and-egg problem)
- ▶ Key management is critical (connected with flexibility, original design (RFC 2535) was fatally flawed because did not consider carefully key management)
- ▶ Denial of existence : prove a domain for which a query was made, does not exist
- ▶ Incremental deployment, flexible to add new domains
- ▶ Cryptography alone adds new DoS due to crypto errors and attacks

Key Validation

- ▶ How to obtain certified public keys of zones, to verify the digital signatures
- ▶ New DNS records KEY, signed by servers in other zones
- ▶ Approaches
 - ▶ **Tree structure**: each parents signs the keys of children
 - ▶ **PGP-style web of trust**
 - ▶ **Mesh**: combination between the above, specifies how to find a path of trust

Tree-Based Key Validation

- ▶ *A key is valid if it is signed by the parent and the parent key is valid*
- ▶ Resolvers configured to trust a master key
- ▶ The good:
 - ▶ Fits the DNS structure
 - ▶ Efficient, no problem to find path of trust
- ▶ The bad:
 - ▶ Difficult to deploy incrementally
 - ▶ Single point of failure
 - ▶ Undesirable trust relationships

Web of Trust Key Validation

- ▶ *A key is valid if it is signed by at least one other key, any chain of trust is possible*
- ▶ The good:
 - ▶ Ease of deployment
 - ▶ No single point of failure
 - ▶ Scalable key signing
- ▶ The bad:
 - ▶ Unauthorized signatures
 - ▶ Finding a chain of trust

Mesh-Based Key Validation

- ▶ Any zone may sign the public key of any other zone, the resolver decode which signatures are considered valid
- ▶ Each zone key has associated a routing record (lists the last link in a chain of trust);
- ▶ Default route record
- ▶ The good:
 - ▶ Ease of deployment
 - ▶ No single point of failure
 - ▶ Scalable key signing
- ▶ The bad ?

Signing the root

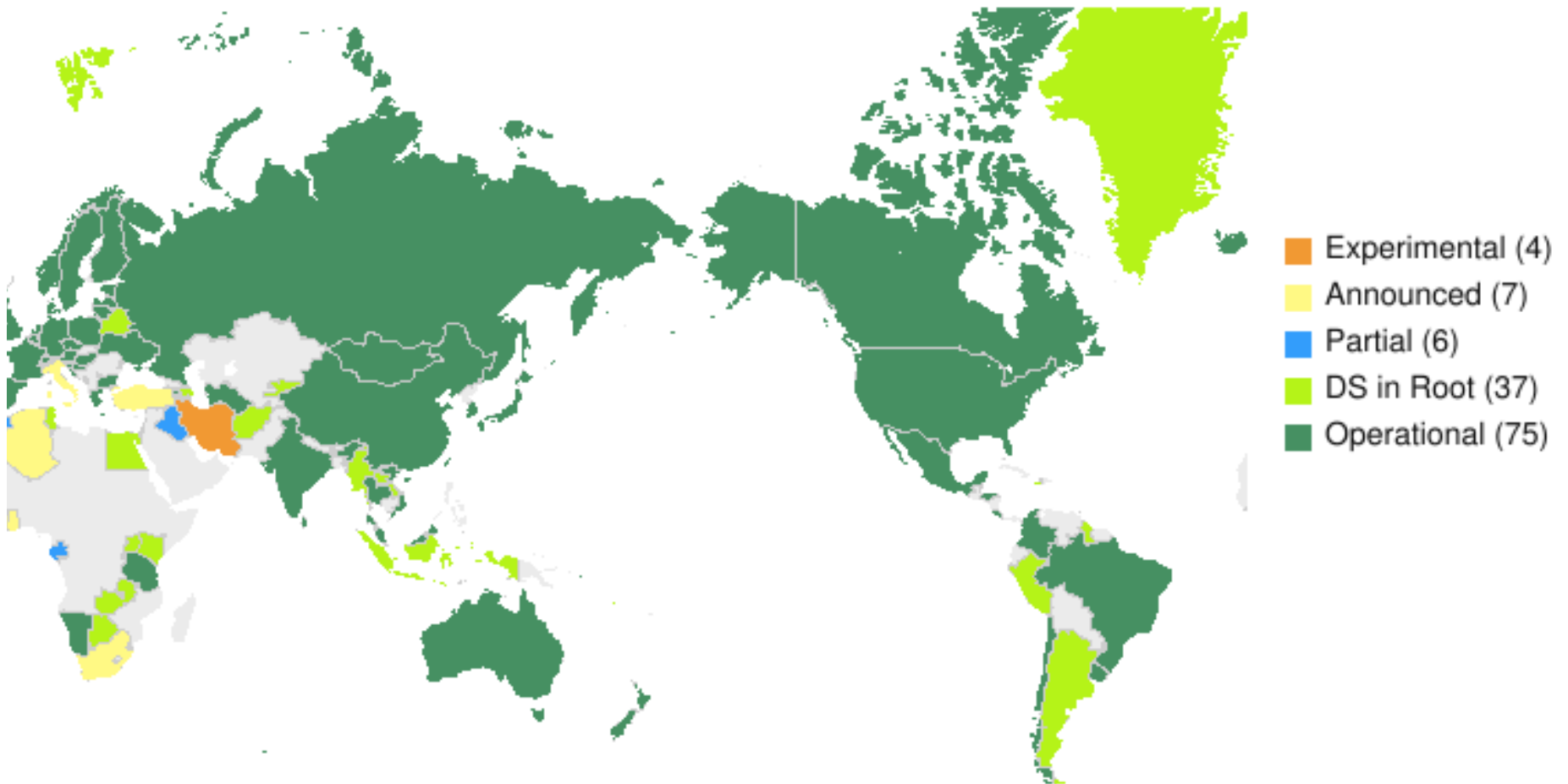
- ▶ **December 1, 2009:** Root zone signed for internal use by VeriSign and ICANN. ICANN and VeriSign exercise interaction protocols for signing the ZSK with the KSK.
- ▶ **January, 2010:** The first root server begins serving the signed root in the form of the DURZ (deliberately unvalidatable root zone). The DURZ contains unusable keys in place of the root KSK and ZSK to prevent these keys being used for validation.
- ▶ **Early May, 2010:** All root servers are now serving the DURZ. The effects of the larger responses from the signed root, if any, would now be encountered.
- ▶ **May and June, 2010:** The deployment results are studied and a final decision to deploy DNSSEC in the root zone is made.
- ▶ **June 16, 2010:** ICANN holds first KSK ceremony event in Culpeper, VA, USA
- ▶ **July 12, 2010:** ICANN holds second KSK ceremony event in El Segundo, CA, USA
- ▶ **July 15, 2010:** ICANN publishes the root zone trust anchor and root operators begin to serve the signed root zone with actual keys – **The signed root zone is available.**

DNS deployment

- ▶ On the roots since July 2010
- ▶ Verisign enabled it on .com and .net in January 2011
- ▶ Comcast is the first major ISP to support it (January 2012)

Country-based deployment

ccTLD DNSSEC Status on 2016-01-25



DNSSEC new resource types

- ▶ **DNSKEY**

- ▶ Public key for a zone
- ▶ Signed by the private key of the parent zone
- ▶ Signatures from the root servers are trusted by default

- ▶ **DS**

- ▶ Delegated signer

- ▶ **RRSIG**

- ▶ Digital signature of a specific resource record
- ▶ Signed by the private key of the zone

- ▶ **NSEC***

- ▶ Signed denial of record existence

Hierarchical key structure

- ▶ **Keys in DNSKEY records**
 - ▶ **Key signing keys (KSK)** which are used to sign other DNSKEY records.
 - ▶ **Zone signing keys (ZSK)** which are used to sign other records.
- ▶ ZSKs are under complete control and use by one particular DNS zone, they can be switched more easily and more often.
- ▶ For a new KSK
 - ▶ the DS record must be transferred to the parent zone and published there.

Zone enumeration

- ▶ In DNSSEC it is possible for an attacker to enumerate all the names in a zone by following the NSEC chain.
- ▶ NSEC RRs assert which names do not exist in a zone by linking from existing name to existing name along a canonical ordering of all the names within a zone.
- ▶ Allows an attacker to map network hosts or other resources

DNSSEC Example

