DNS AND DNSSEC

Instructor: Yizheng Chen

Credits: slides from Dave Levin

CMSC 414

NAMING

- IP addresses allow global connectivity
- But they’re pretty useless for humans!
  - Can’t be expected to pick their own IP address
  - Can’t be expected to remember another’s IP address
- **DHCP** : Setting IP addresses
- **DNS** : Mapping a memorable name to a routable IP address

HOSTNAMES AND IP ADDRESSES

```
gold:~ dml$ ping google.com
PING google.com (74.125.228.65): 56 data bytes
64 bytes from 74.125.228.65: icmp_seq=0 ttl=52 time=22.330 ms
64 bytes from 74.125.228.65: icmp_seq=1 ttl=52 time=6.304 ms
64 bytes from 74.125.228.65: icmp_seq=2 ttl=52 time=5.186 ms
64 bytes from 74.125.228.65: icmp_seq=3 ttl=52 time=12.805 ms
```

HOSTNAMES AND IP ADDRESSES

```
gold:~ dml$ ping google.com
PING google.com (74.125.228.65): 56 data bytes
64 bytes from 74.125.228.65: icmp_seq=0 ttl=52 time=22.330 ms
64 bytes from 74.125.228.65: icmp_seq=1 ttl=52 time=6.304 ms
64 bytes from 74.125.228.65: icmp_seq=2 ttl=52 time=5.186 ms
64 bytes from 74.125.228.65: icmp_seq=3 ttl=52 time=12.805 ms
```

HOSTNAMES AND IP ADDRESSES

```
gold:~ dml$ ping google.com
PING google.com (74.125.228.65): 56 data bytes
64 bytes from 74.125.228.65: icmp_seq=0 ttl=52 time=22.330 ms
64 bytes from 74.125.228.65: icmp_seq=1 ttl=52 time=6.304 ms
64 bytes from 74.125.228.65: icmp_seq=2 ttl=52 time=5.186 ms
64 bytes from 74.125.228.65: icmp_seq=3 ttl=52 time=12.805 ms
```

HOSTNAMES AND IP ADDRESSES

```
gold:~ dml$ ping google.com
PING google.com (74.125.228.65): 56 data bytes
64 bytes from 74.125.228.65: icmp_seq=0 ttl=52 time=22.330 ms
64 bytes from 74.125.228.65: icmp_seq=1 ttl=52 time=6.304 ms
64 bytes from 74.125.228.65: icmp_seq=2 ttl=52 time=5.186 ms
64 bytes from 74.125.228.65: icmp_seq=3 ttl=52 time=12.805 ms
```

google.com is easy to remember, but not routable

74.125.228.65 is routable

Name resolution:

The process of mapping from one to the other

TERMINOLOGY

- www.cs.umd.edu = "**domain name**"
  - www.cs.umd.edu is a "subdomain" of cs.umd.edu

- Domain names can map to a set of IP addresses

```
gold:~ dml$ dig google.com

; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35815
;; flags: qr rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             105     IN      A       74.125.228.70
google.com.             105     IN      A       74.125.228.66
google.com.             105     IN      A       74.125.228.64
google.com.             105     IN      A       74.125.228.69
google.com.             105     IN      A       74.125.228.78
google.com.             105     IN      A       74.125.228.73
google.com.             105     IN      A       74.125.228.68
google.com.             105     IN      A       74.125.228.65
google.com.             105     IN      A       74.125.228.72
```

We'll understand this more in a bit; for now, note that google.com is mapped to many IP addresses

TERMINOLOGY

- www.cs.umd.edu = "**domain name**"
  - www.cs.umd.edu is a "subdomain" of cs.umd.edu

- Domain names can map to a set of IP addresses

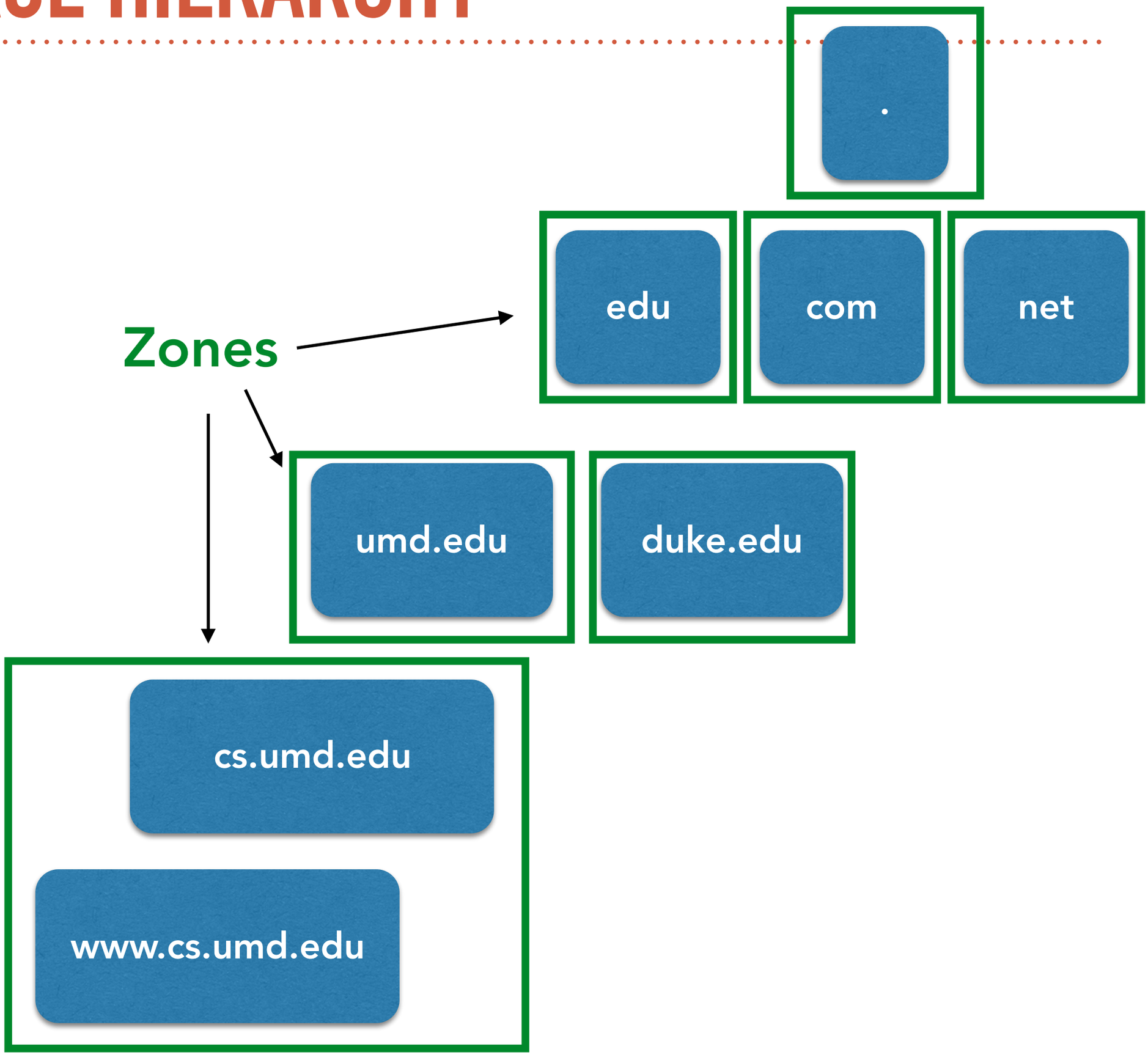
```
gold:~ dml$ dig google.com

; <<>> DiG 9.8.3-P1 <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 35815
;; flags: qr rd ra; QUERY: 1, ANSWER: 11, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;google.com.                    IN      A

;; ANSWER SECTION:
google.com.             105     IN      A       74.125.228.70
google.com.             105     IN      A       74.125.228.66
google.com.             105     IN      A       74.125.228.64
google.com.             105     IN      A       74.125.228.69
google.com.             105     IN      A       74.125.228.78
google.com.             105     IN      A       74.125.228.73
google.com.             105     IN      A       74.125.228.68
google.com.             105     IN      A       74.125.228.65
google.com.             105     IN      A       74.125.228.72
```

We'll understand this more in a bit; for now, note that google.com is mapped to many IP addresses

TERMINOLOGY

- "**zone**" = a portion of the DNS namespace, divided up for administrative reasons
  - Think of it like a collection of hostname/IP address pairs that happen to be lumped together
    - www.google.com, mail.google.com, dev.google.com, …
- Subdomains do not need to be in the same zone
  - Allows the owner of one zone (umd.edu) to delegate responsibility to another (cs.umd.edu)

NAMESPACE HIERARCHY



TERMINOLOGY

- "**Nameserver**" = A piece of code that answers queries of the form "What is the IP address for foo.bar.com?"
  - Every zone must run ≥2 nameservers
  - Several very common nameserver implementations: BIND, PowerDNS (more popular in Europe)

- "**Authoritative nameserver**":
  - Every zone has to maintain a file that maps IP addresses and hostnames ("www.cs.umd.edu is 128.8.127.3")
  - One of the name servers in the zone has the *master* copy of this file. It is the authority on the mapping.

TERMINOLOGY

- "**Resolver**" - while name servers *answer* queries, resolvers *ask* queries.

- Every OS has a resolver. Typically small and pretty dumb. All it typically does it forward the query to a local…

- "**Recursive nameserver**" - a nameserver which will do the heavy lifting, issuing queries on behalf of the client resolver until an authoritative answer returns.

- Prevalence
  - There is almost always a *local* (private) recursive name server
  - But very rare for name servers to support recursive queries otherwise

TERMINOLOGY

- "**Record**" (or "resource record") = usually think of it as a mapping between hostname and IP address

- But more generally, it can map virtually anything to virtually anything

- Many record types:
  - (**A**)ddress records (IP <-> hostname)
  - Mail server (**MX**, mail exchanger)
  - SOA (start of authority, to delineate different zones)
  - Others for DNSSEC to be able to share keys

- Records are the unit of information

TERMINOLOGY

Nameservers within a zone must be able to give:

- **Authoritative answers (A)** for hostnames in that zone
  - The umd.edu zone's nameservers must be able to tell us what the IP address for umd.edu is

"A" record: umd.edu = 54.84.241.99

54.84.241.99 is a valid IP address for umd.edu

TERMINOLOGY

Nameservers within a zone must be able to give:

- **Authoritative answers (A)** for hostnames in that zone
  - The umd.edu zone's nameservers must be able to tell us what the IP address for umd.edu is

"A" record: umd.edu = 54.84.241.99

54.84.241.99 is a valid IP address for umd.edu

- Pointers to **name servers (NS)** who host zones in its subdomains
  - The umd.edu zone's nameservers must be able to tell us what the name and IP address of the cs.umd.edu zone's nameservers is

"NS" record: cs.umd.edu = ipa01.cs.umd.edu.

Ask ipa01.cs.umd.edu for all cs.umd.edu subdomains

DNS

Domain Name Service at a very high level

Requesting host

What is an IP address for cs.umd.edu?

DNS

Domain Name Service at a very high level

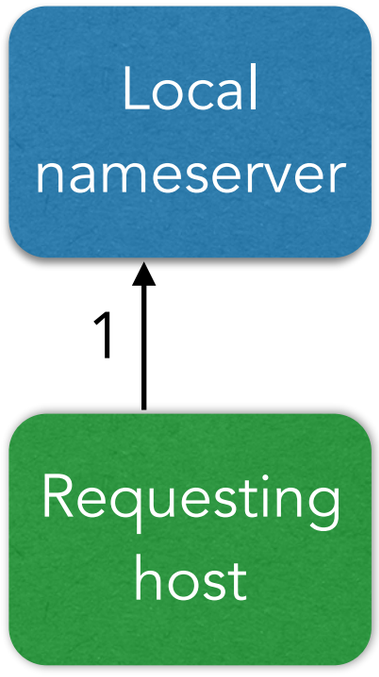
Local nameserver

Requesting host

What is an IP address for cs.umd.edu?

DNS

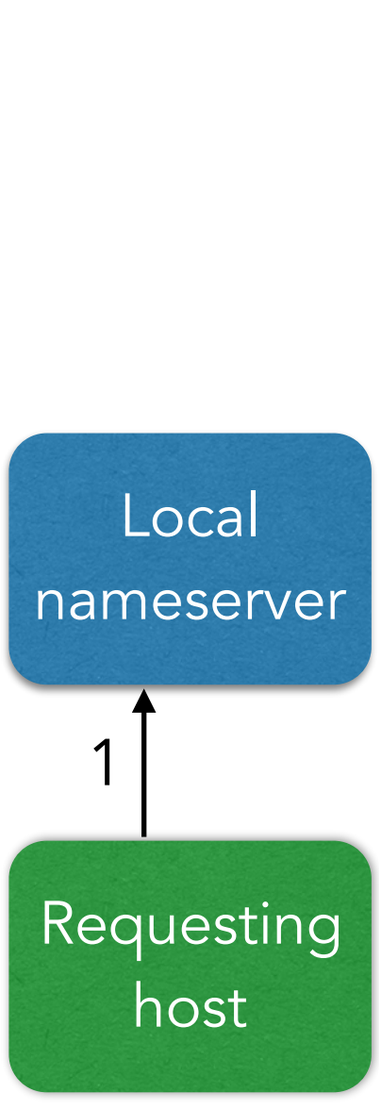
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

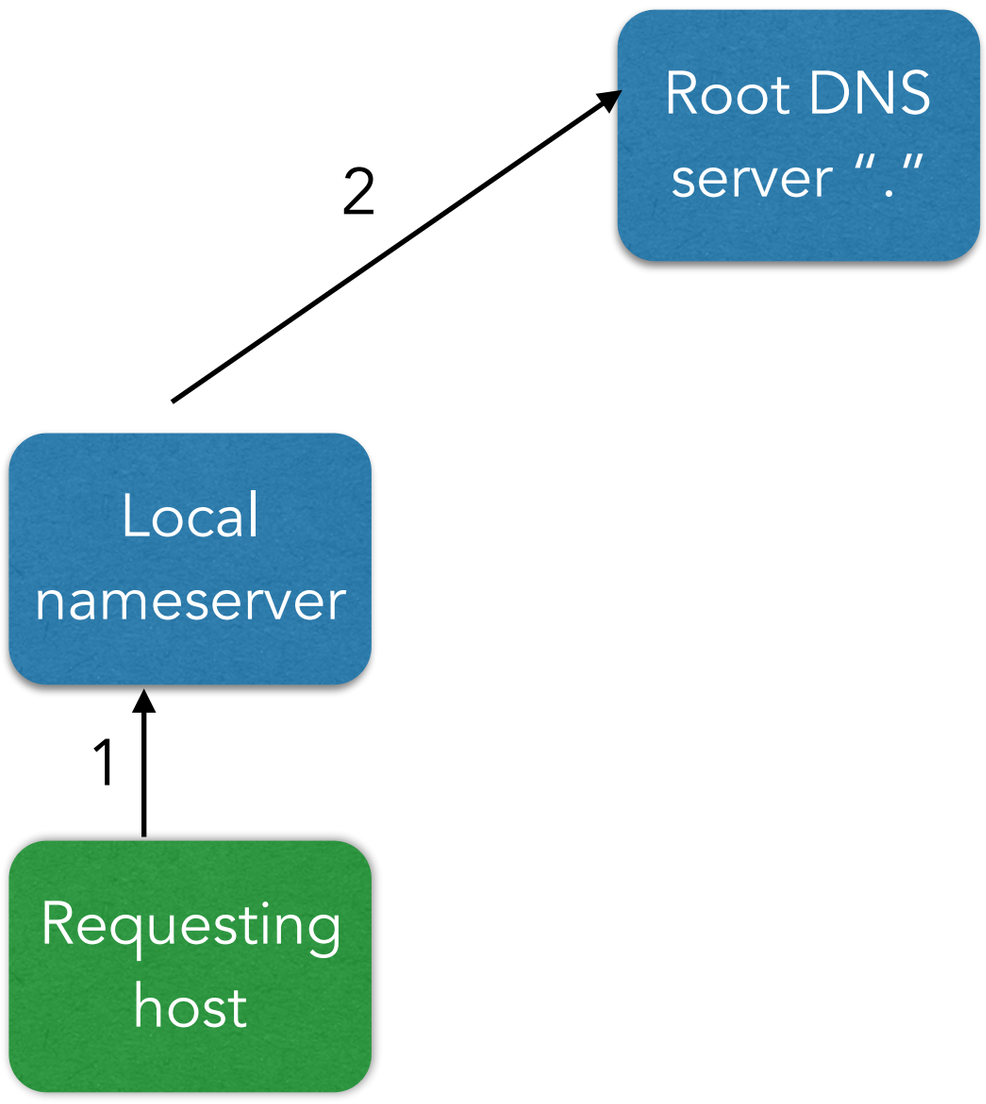
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

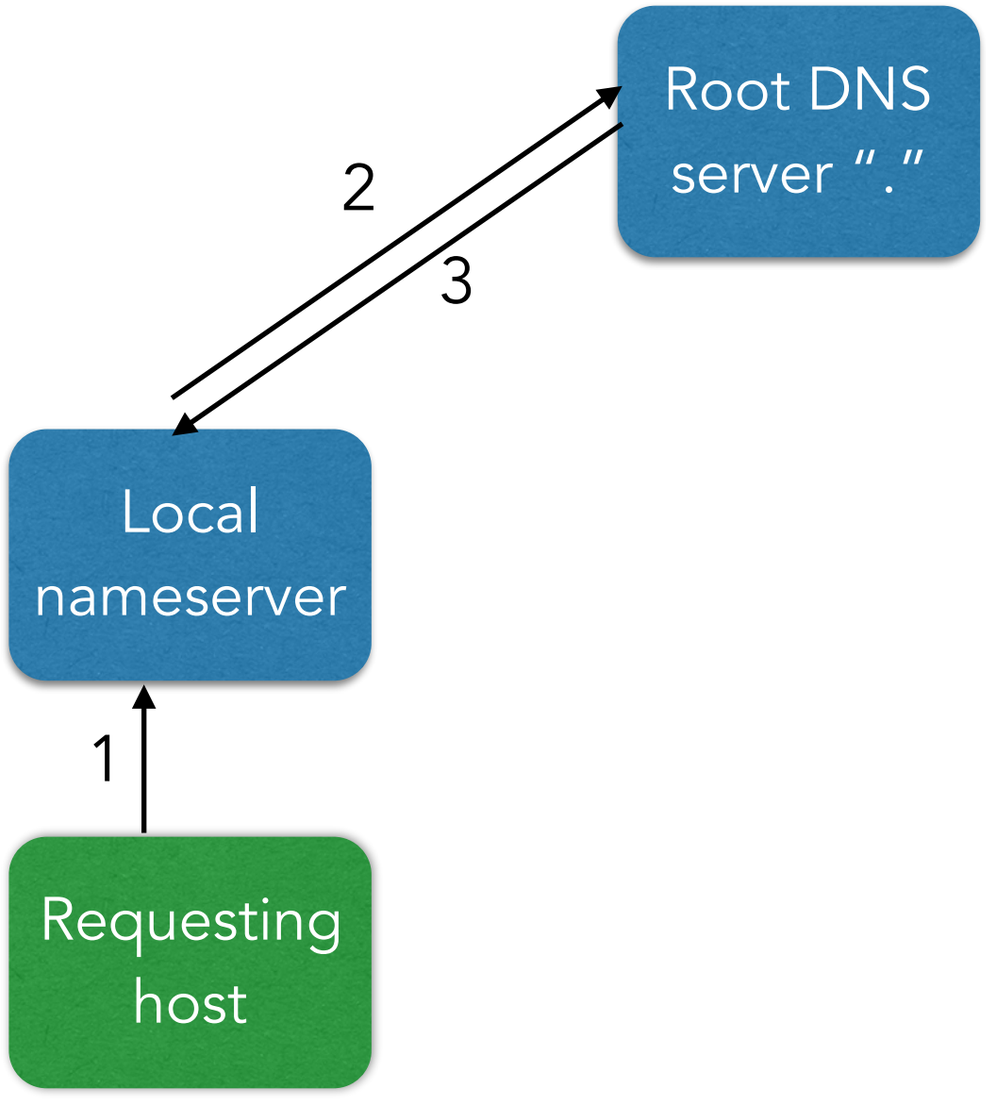
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

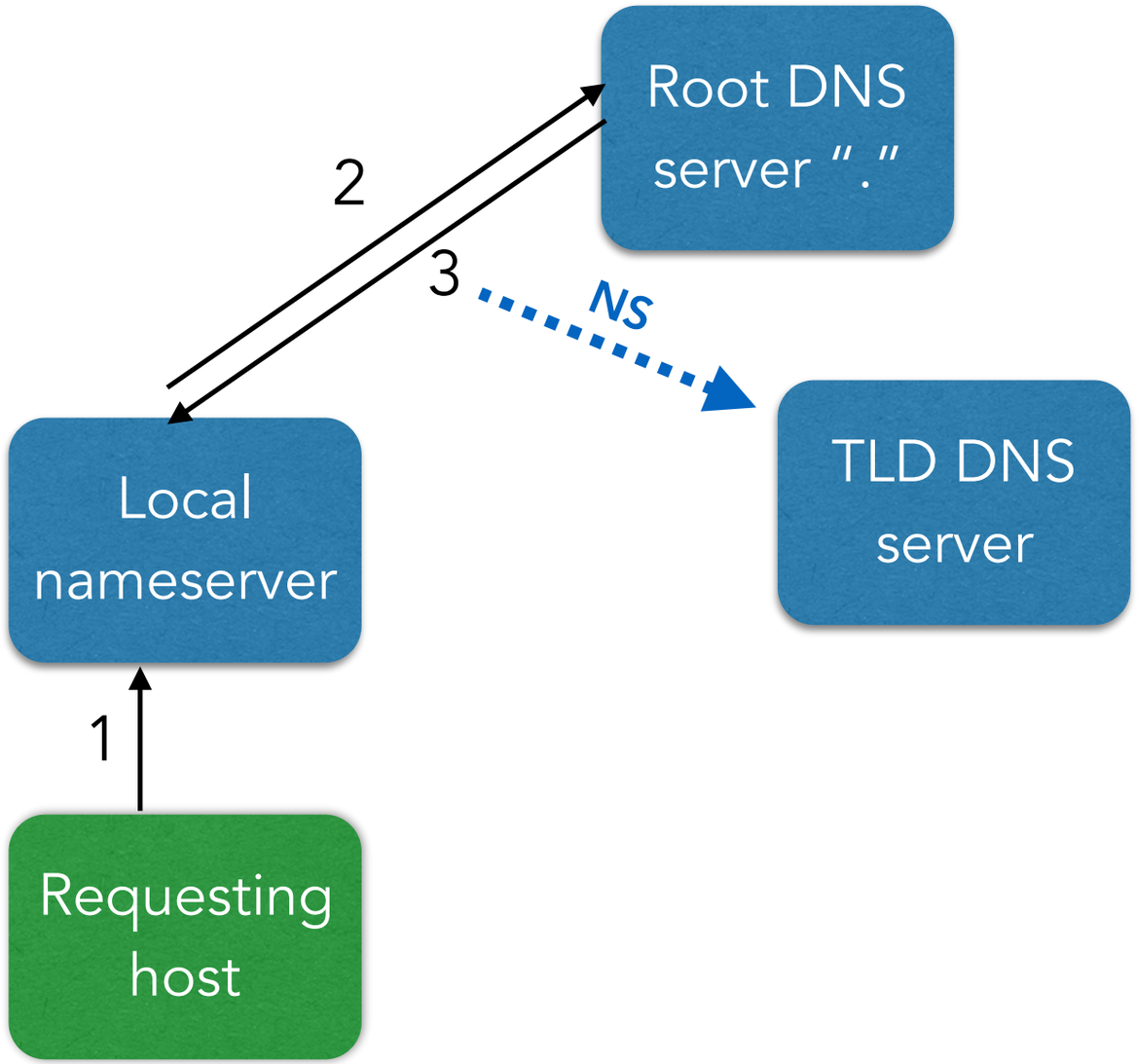
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

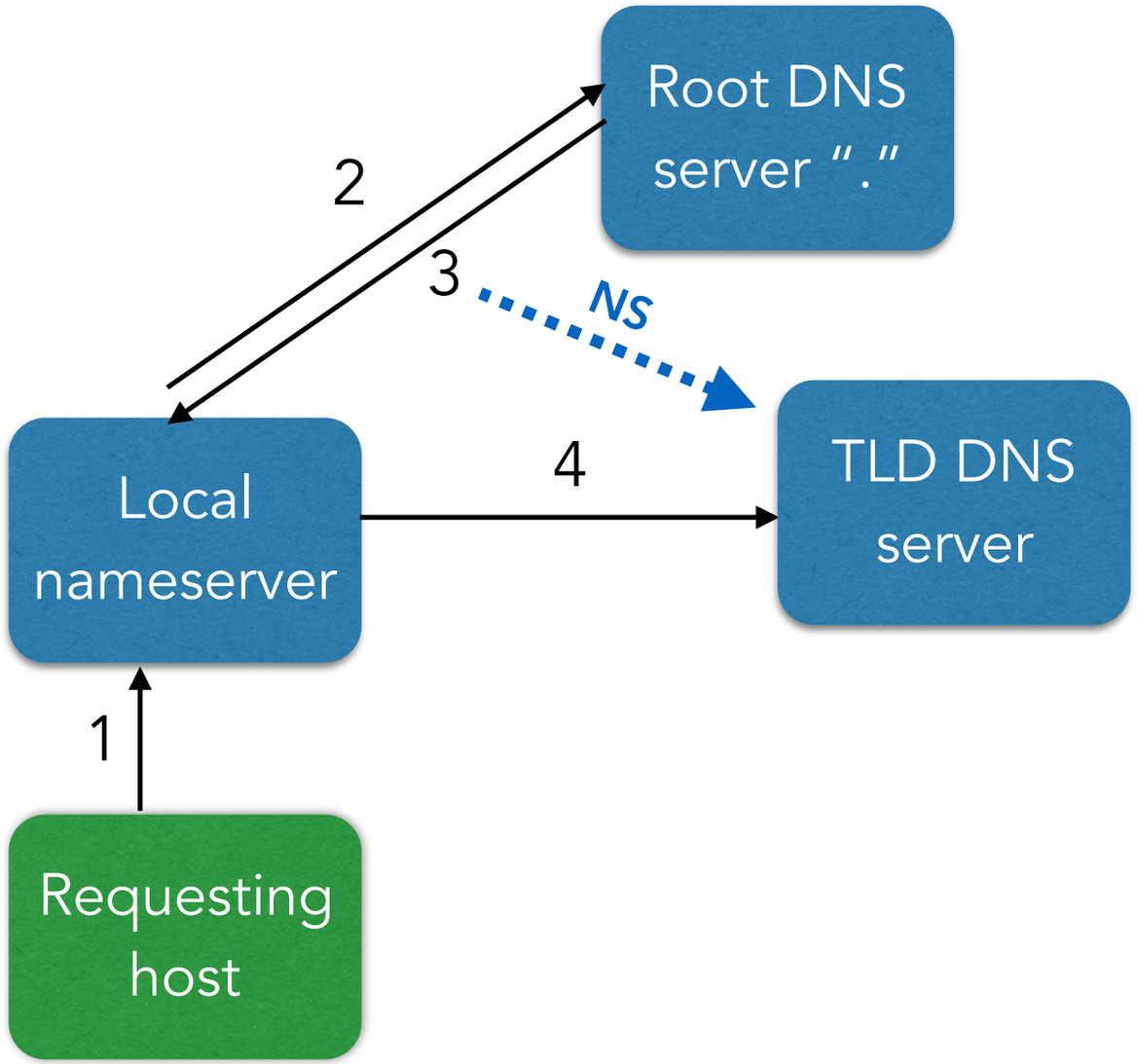
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

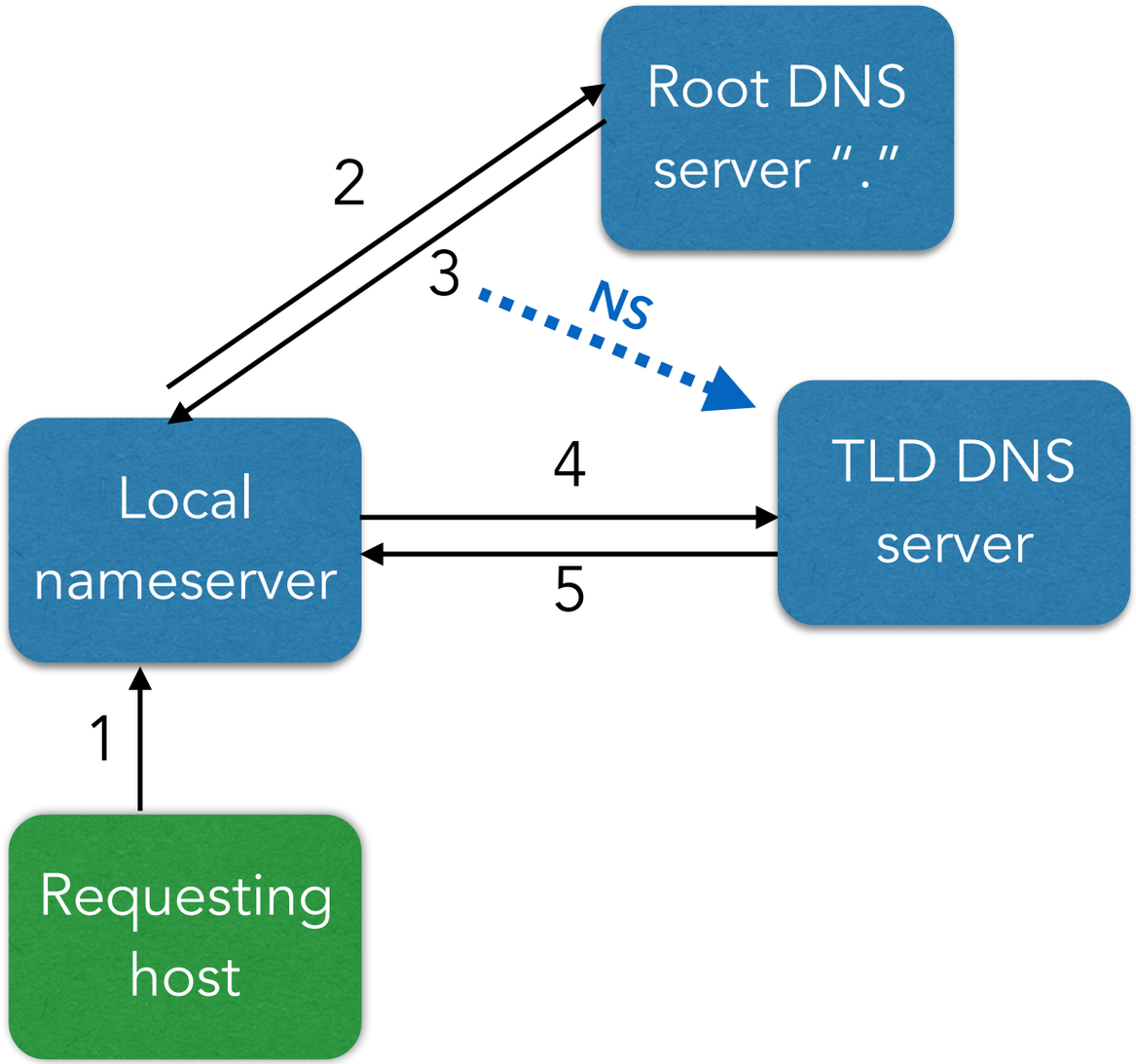
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

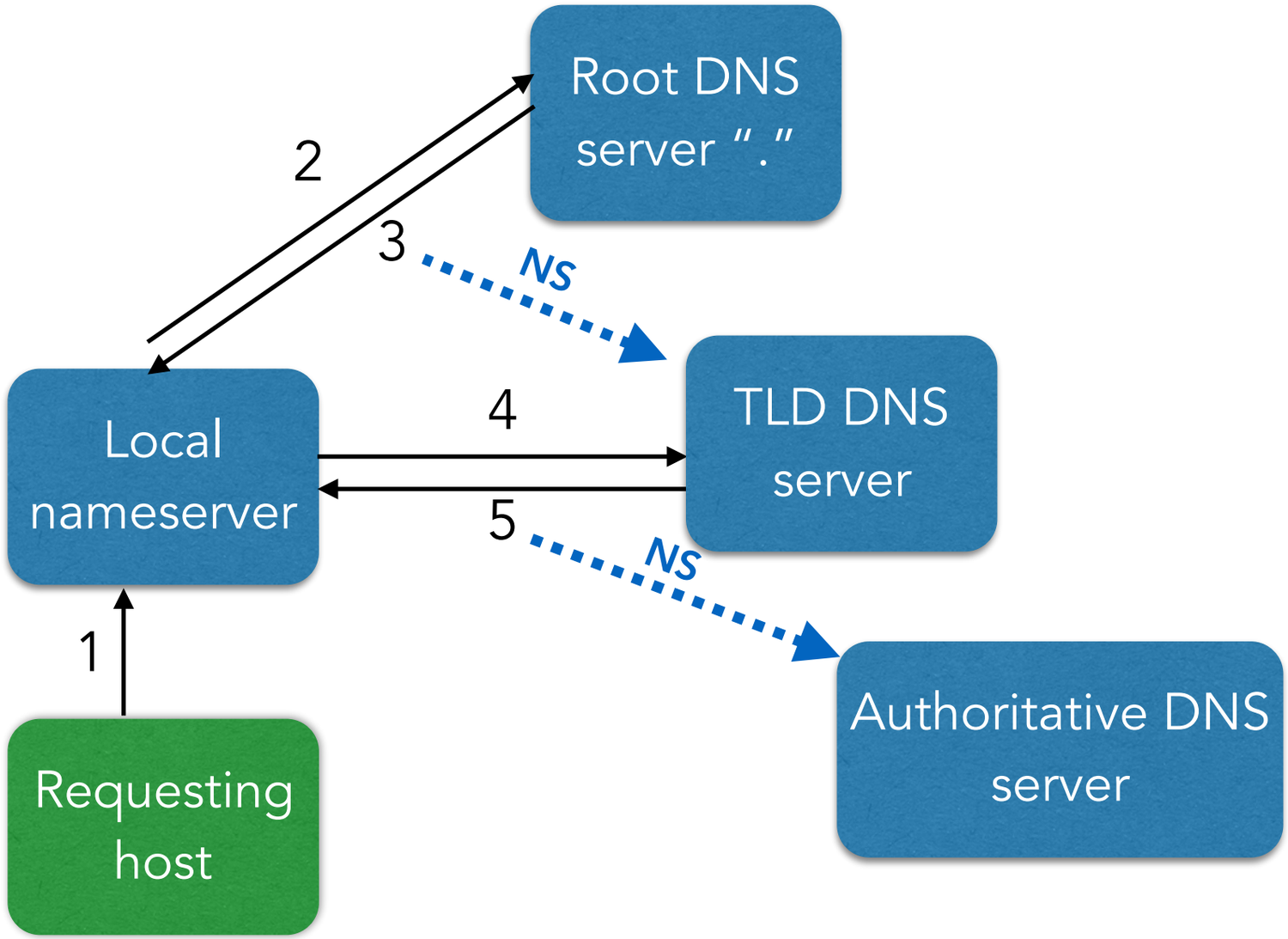
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

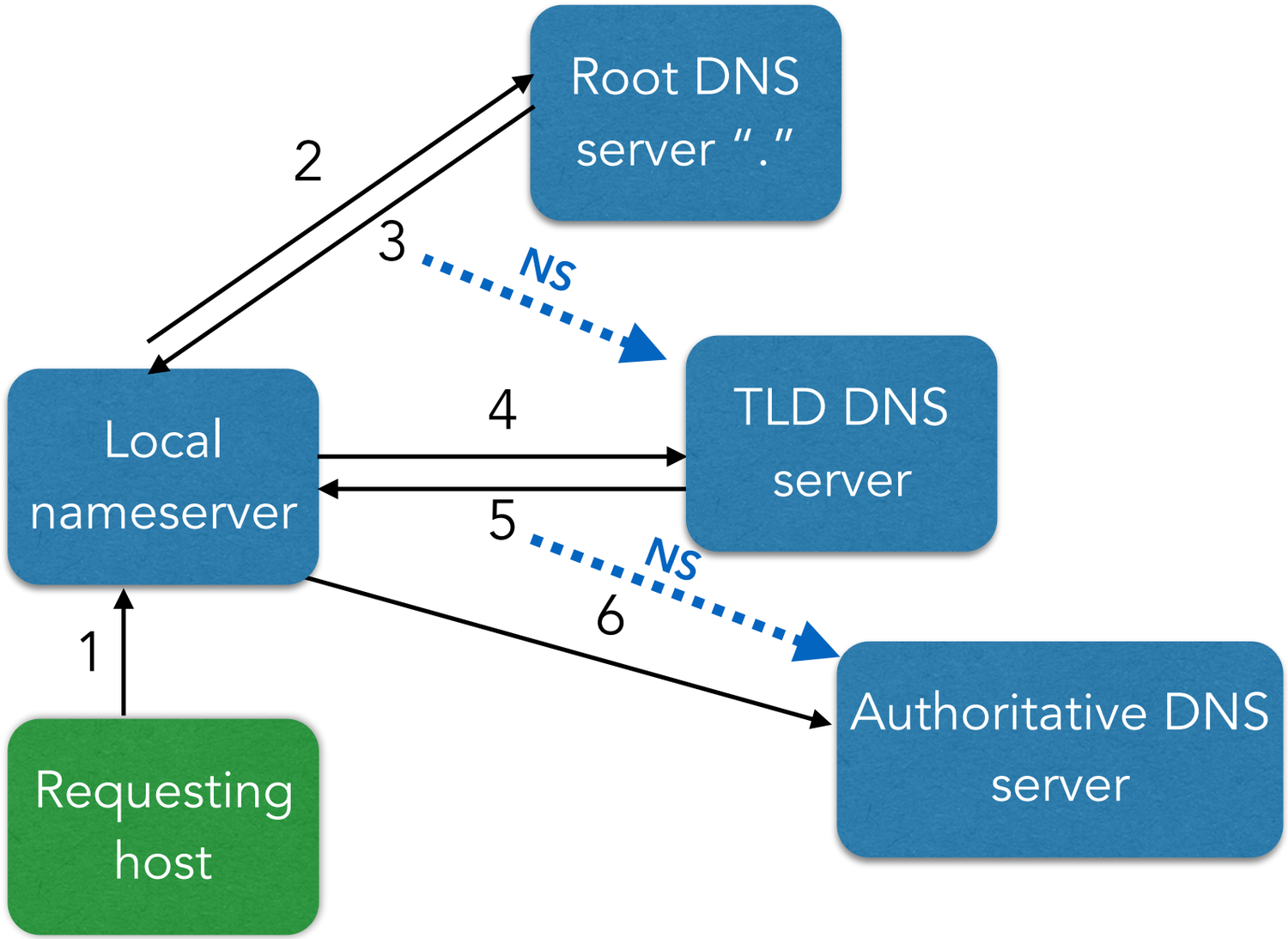
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

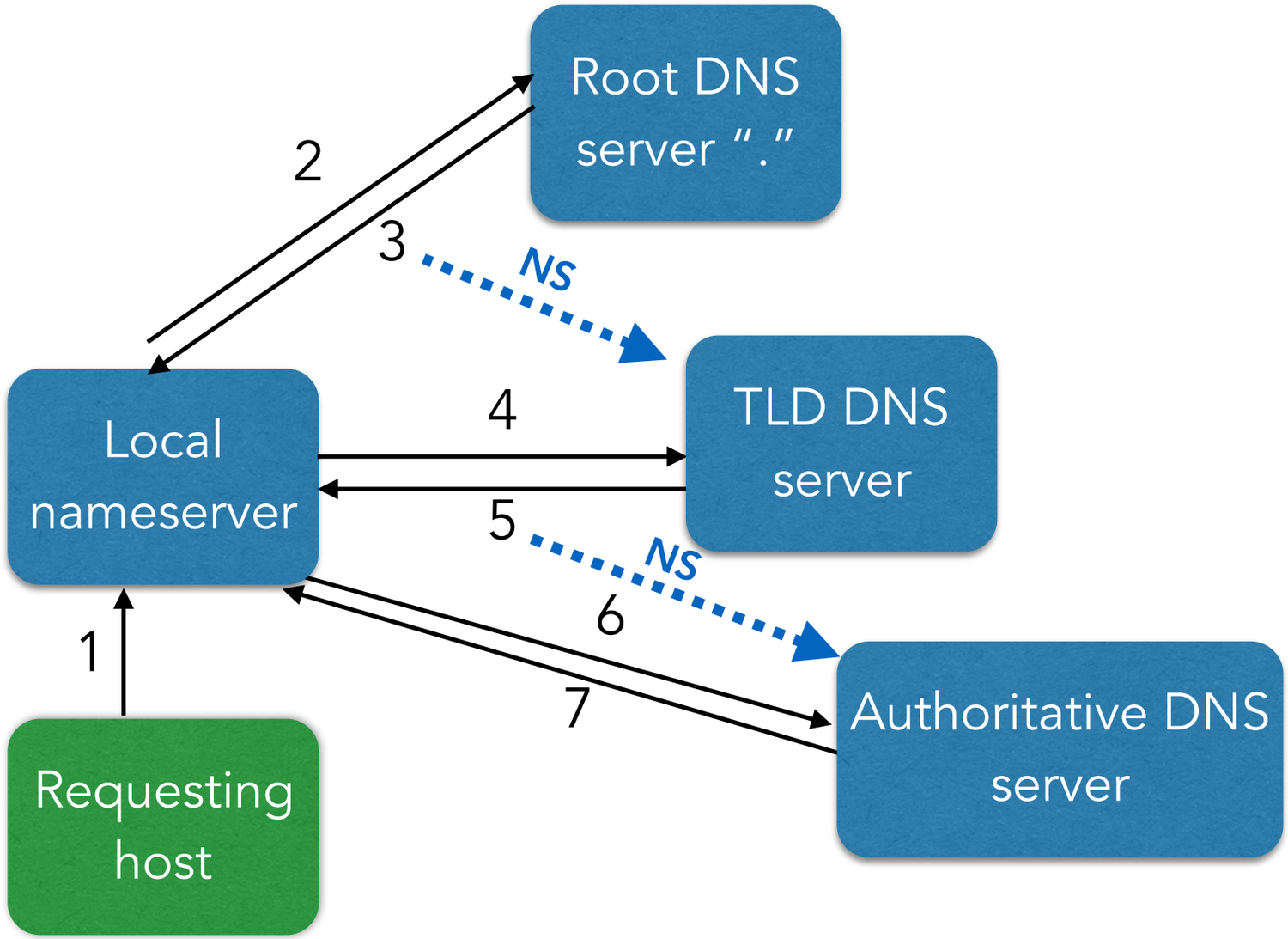
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

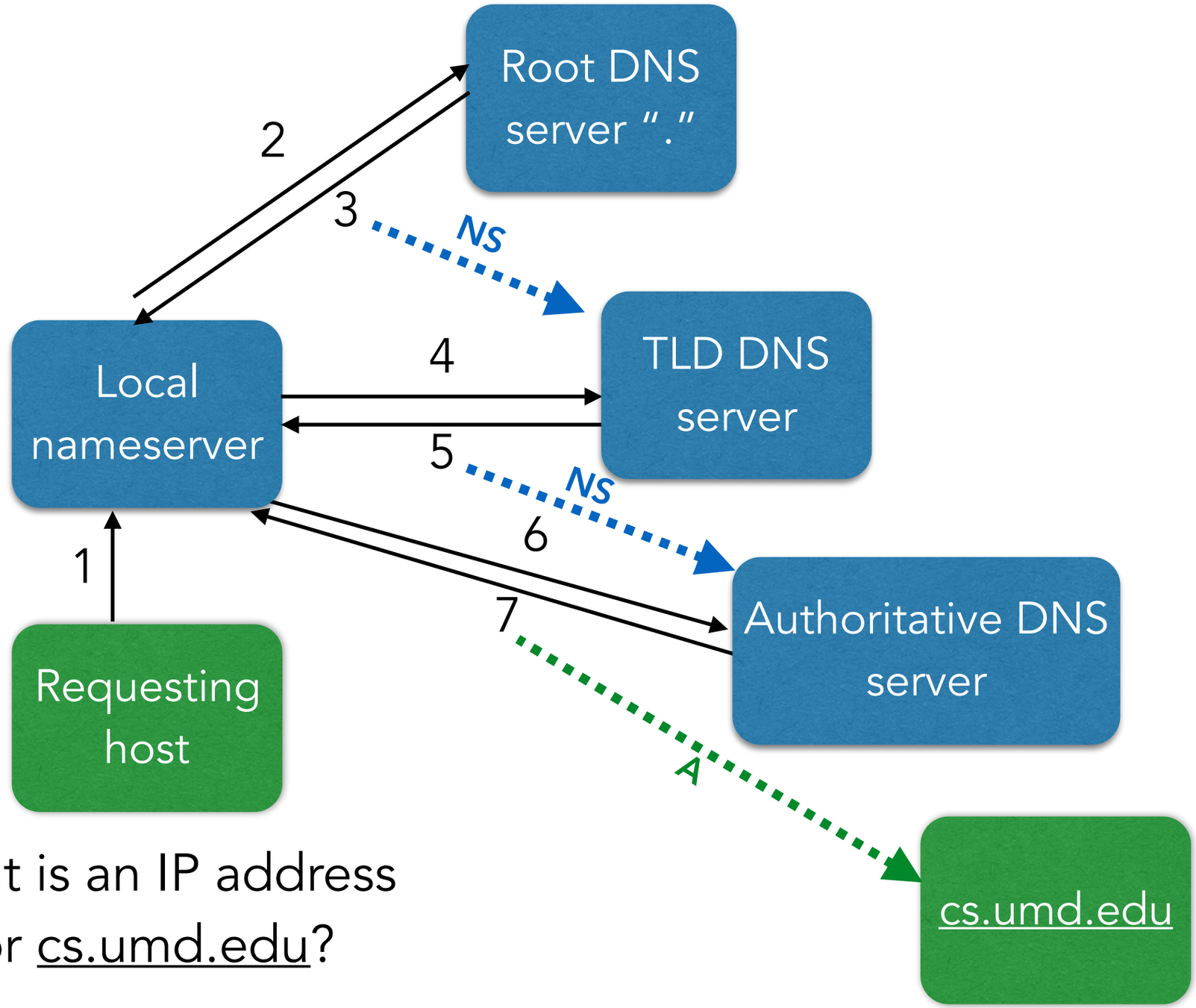
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

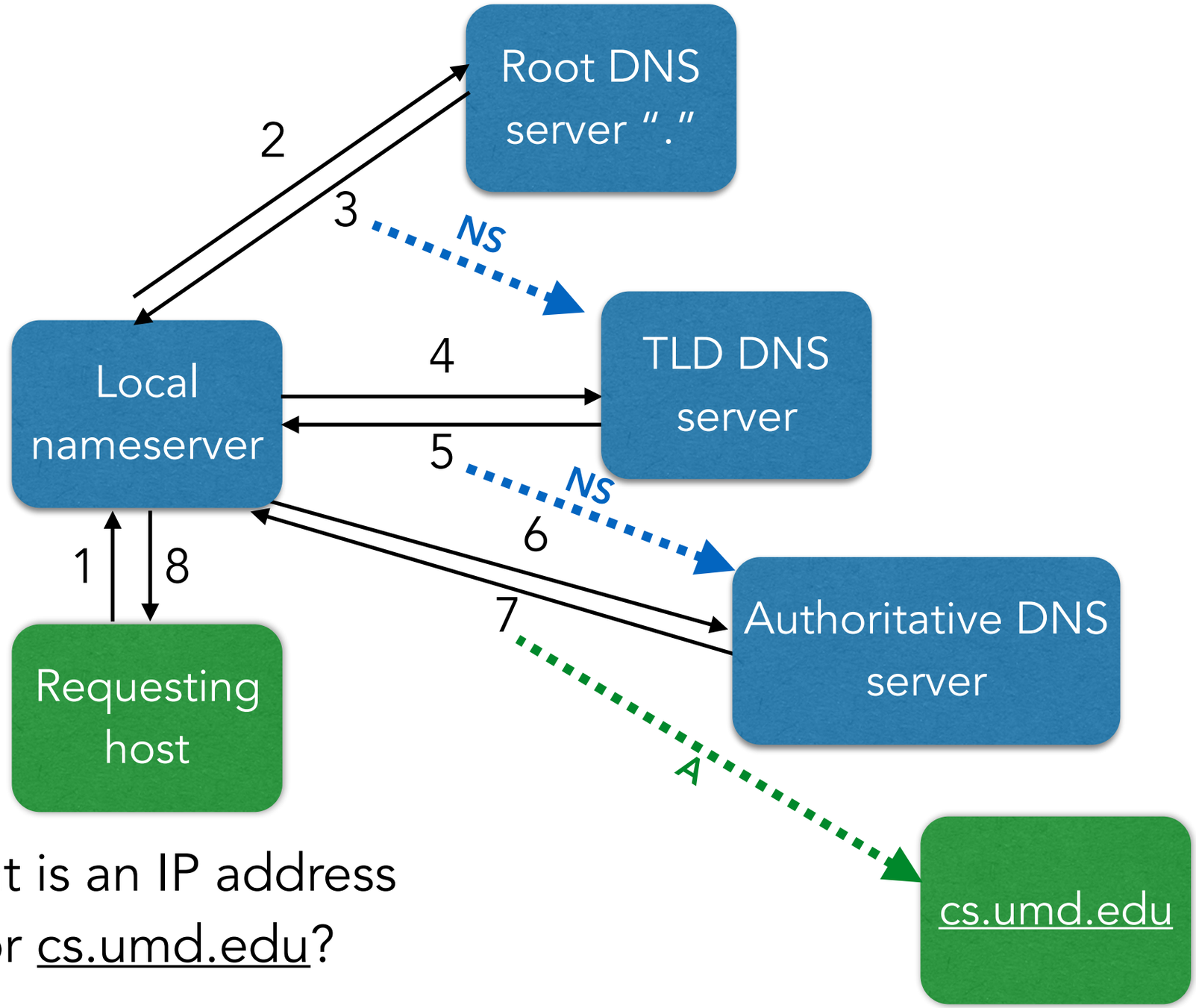
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

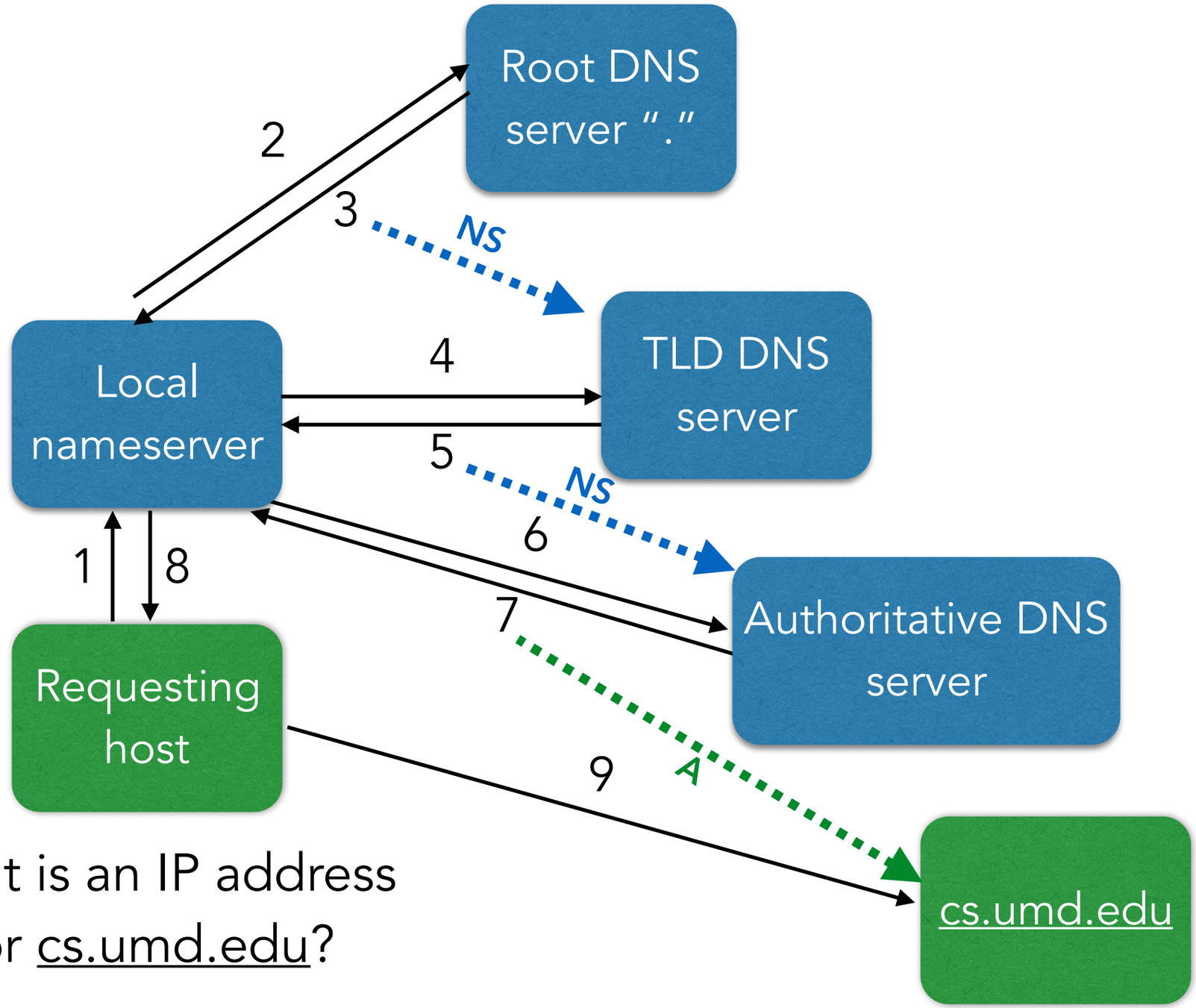
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

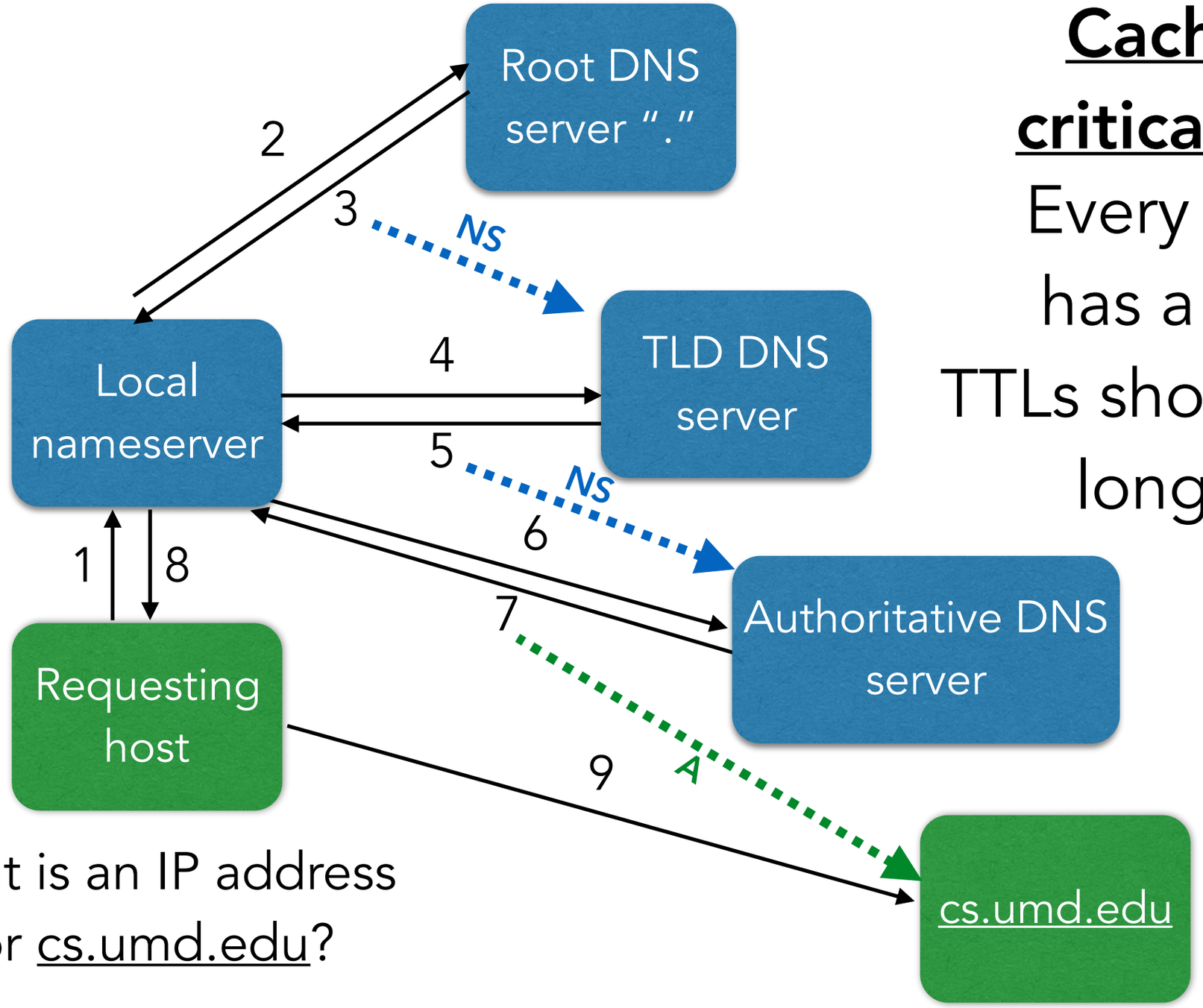
Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

DNS

Domain Name Service at a very high level



What is an IP address for cs.umd.edu?

Caching responses is critical to DNS's success

Every response (3,5,7,8) has a time-to-live (TTL). TTLs should be reasonably long (days), but some are minutes.

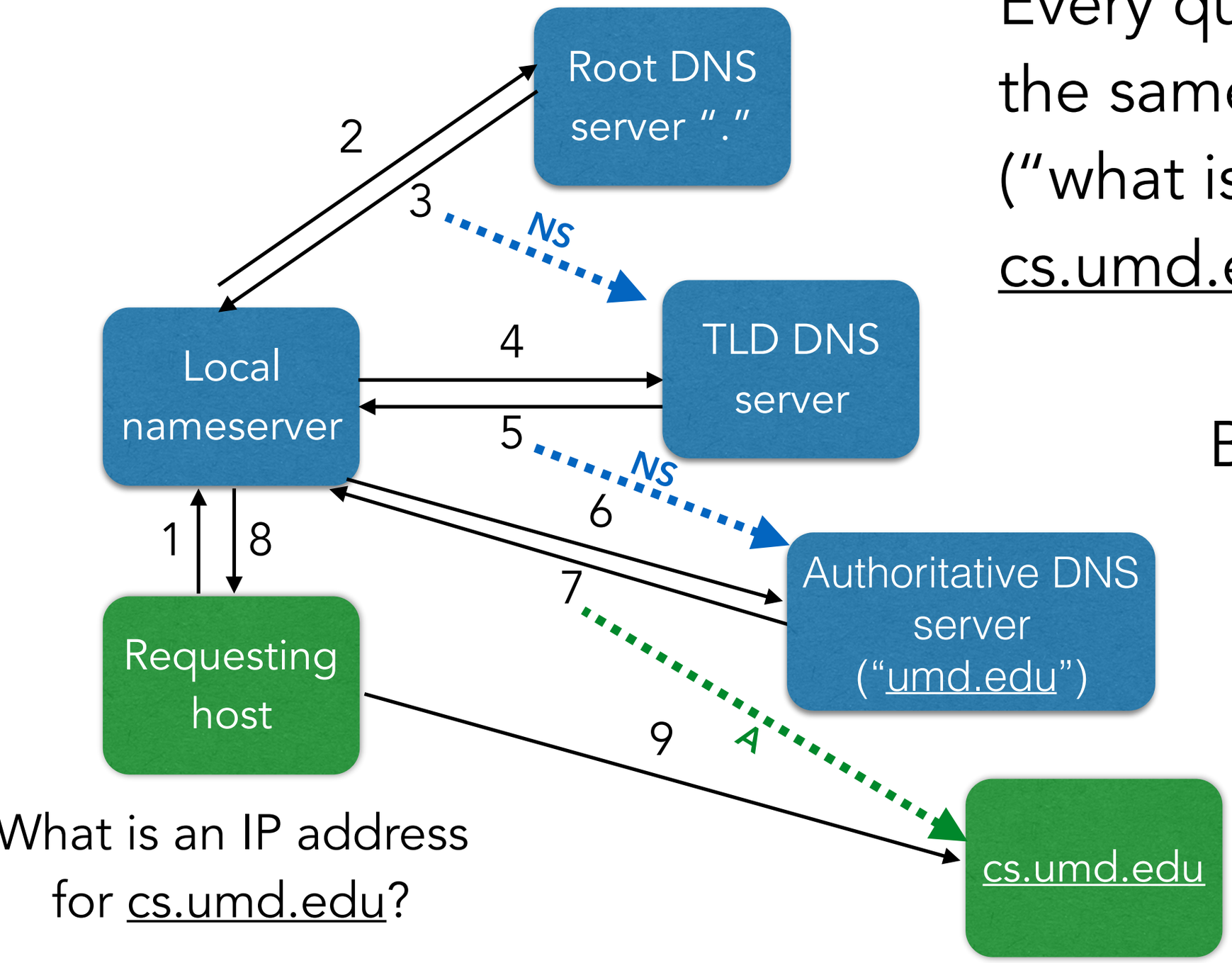
HOW DO THEY KNOW THESE IP ADDRESSES?

- Local DNS server: host learned this via DHCP
- A parent knows its children: part of the registration process
- Root nameserver: *hardcoded* into the local DNS server (and every DNS server)
  - 13 root servers (logically): A-root, B-root, …, M-root
  - These IP addresses change *very* infrequently
  - **UMD runs D-root.**
    - IP address changed beginning of 2013!!
    - For the most part, the change-over went alright, but Lots of weird things happened — ask me some time.

CACHING

- Central to DNS's success
- Also central to attacks
- "Cache poisoning": filling a victim's cache with false information

QUERIES



Every query (2,4,6) has the same request in it ("what is the IP address for cs.umd.edu?")

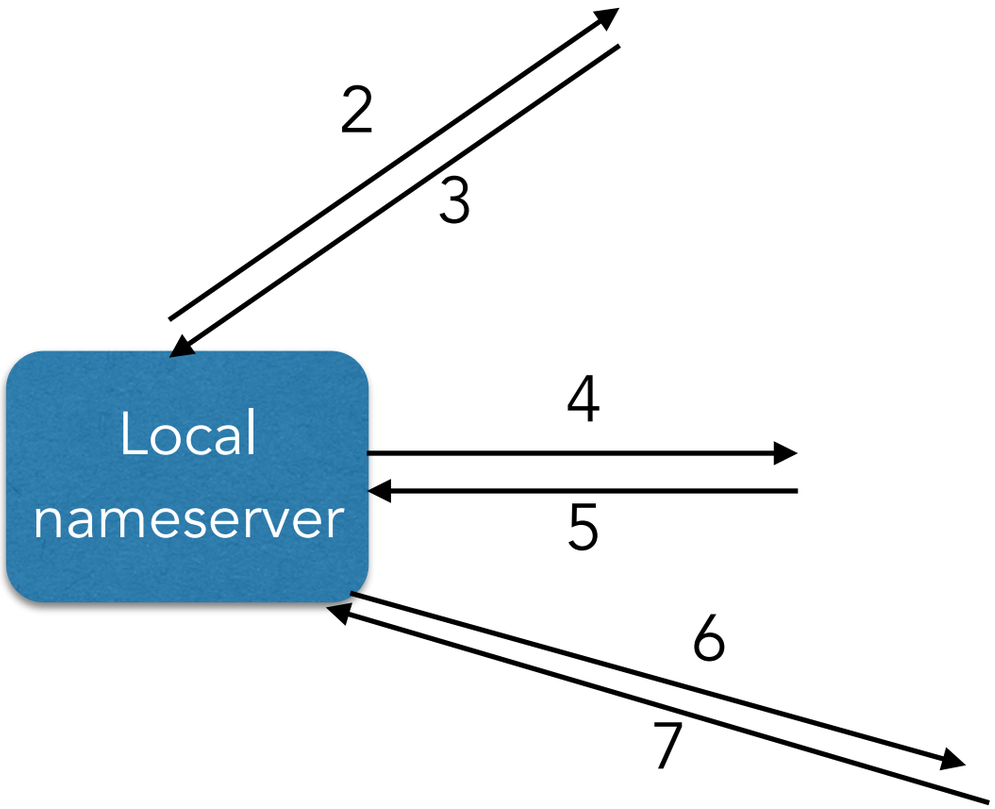
But **different**:

- dst IP (port = 53)
- query ID

WHAT'S IN A RESPONSE?

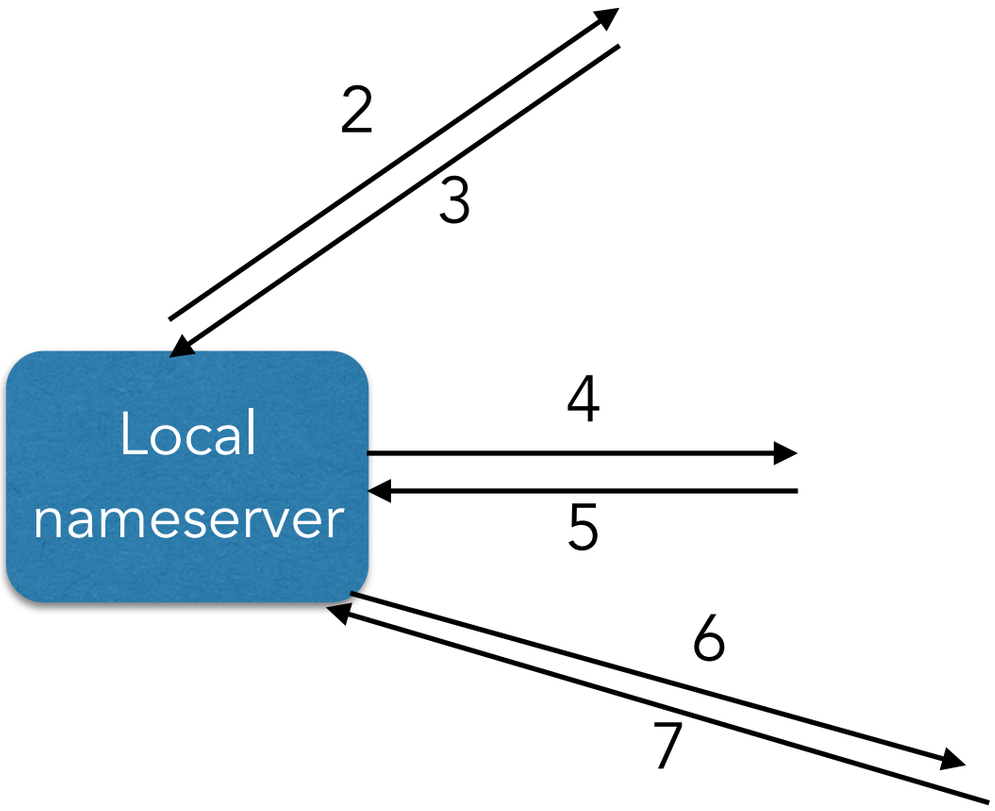
- Many things, but for the attacks we're concerned with…
- A record: gives "the authoritative response for the IP address of this hostname"
- NS record: describes "this is the name of the nameserver who should know more about how to answer this query than I do"
  - Often also contains "glue" records (IP addresses of those name servers to avoid chicken and egg problems)
  - Resolver will generally cache all of this information

QUERY IDS



- The local resolver has a lot of incoming/outgoing queries at any point in time.
- To determine which response maps to which queries, it uses a *query ID*
- Query ID: 16-bit field in the DNS header
  - Requester sets it to whatever it wants
  - Responder must provide the same value in its response

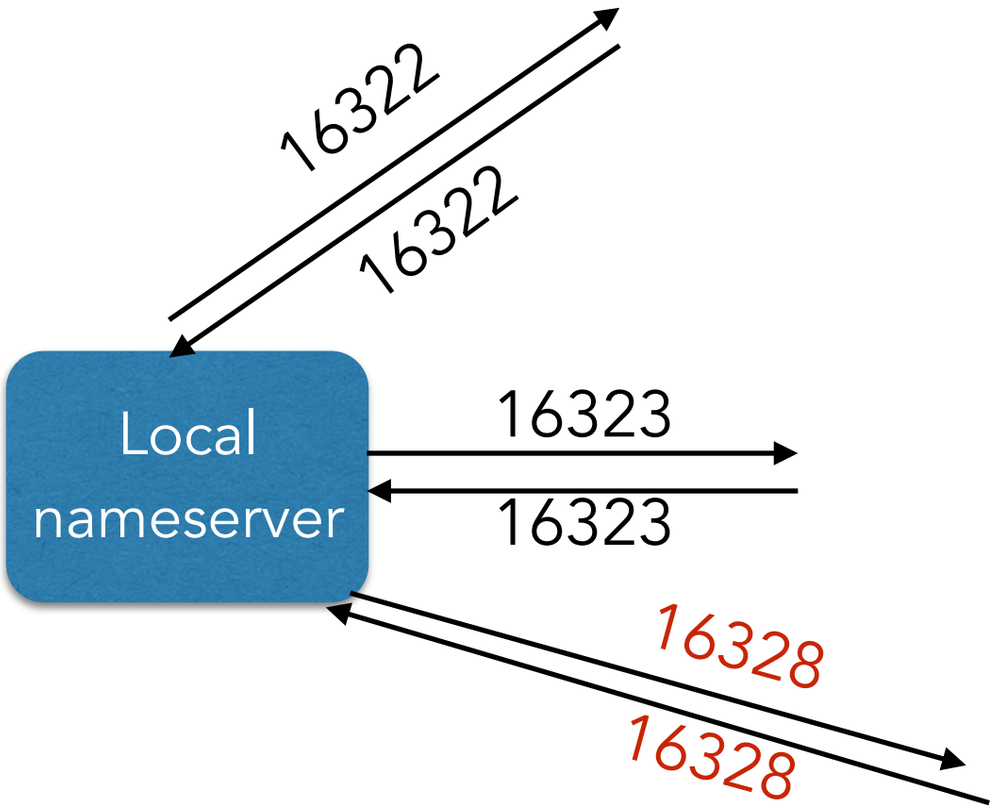
QUERY IDS



- The local resolver has a lot of incoming/outgoing queries at any point in time.
- To determine which response maps to which queries, it uses a *query ID*
- Query ID: 16-bit field in the DNS header
  - Requester sets it to whatever it wants
  - Responder must provide the same value in its response

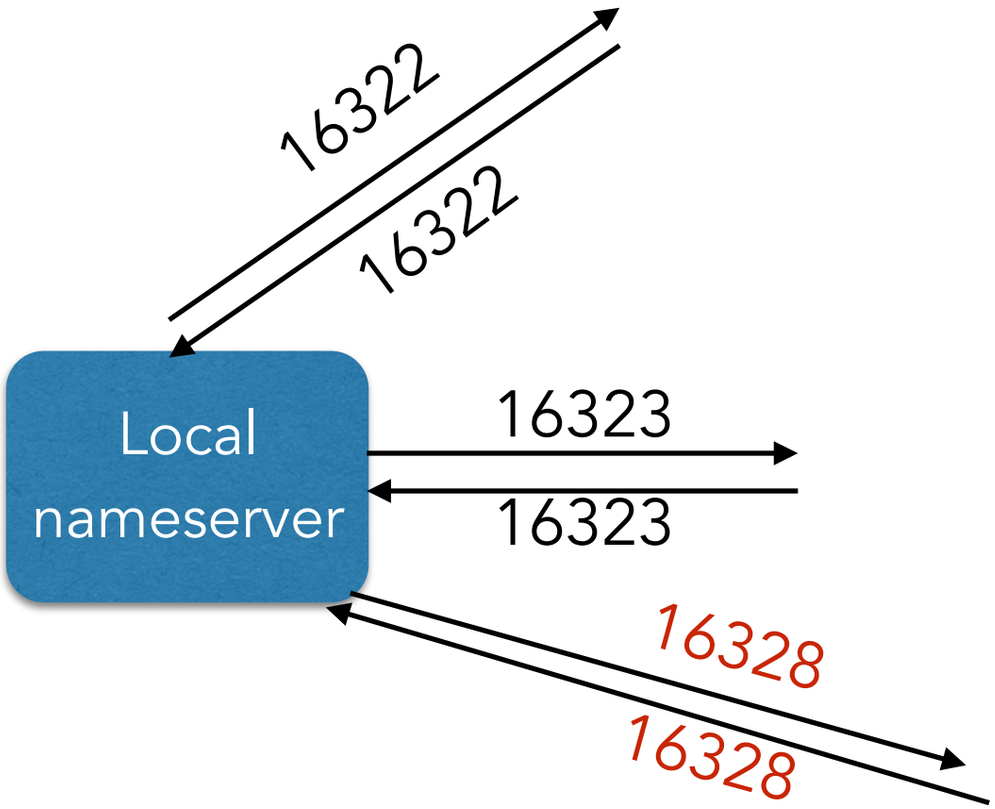
How would you implement query IDs at a resolver?

QUERY IDS USED TO INCREMENT



- Global query ID value
- Map outstanding query ID to local state of who to respond to (the client)
- Basically:
  new Packet(queryID++)

QUERY IDS USED TO INCREMENT



- Global query ID value
- Map outstanding query ID to local state of who to respond to (the client)
- Basically:
  new Packet(queryID++)

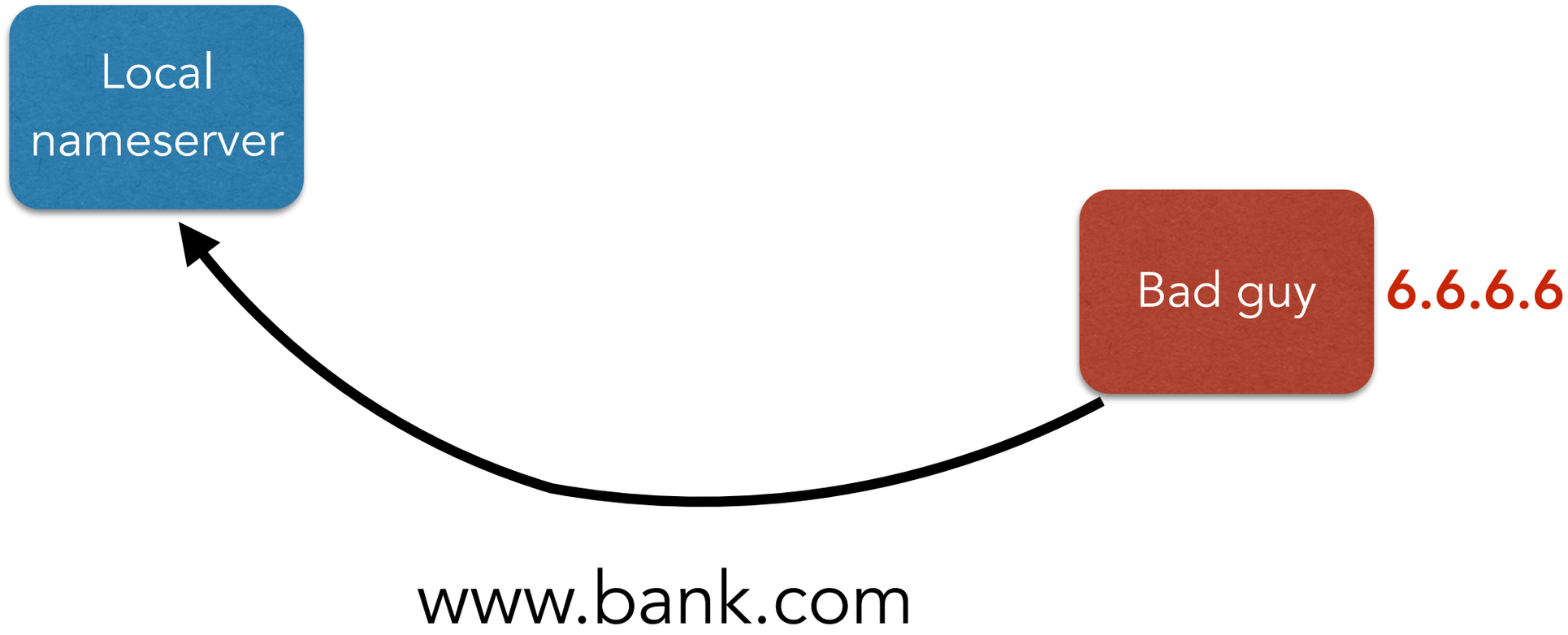
How would you attack this?

CACHE POISONING

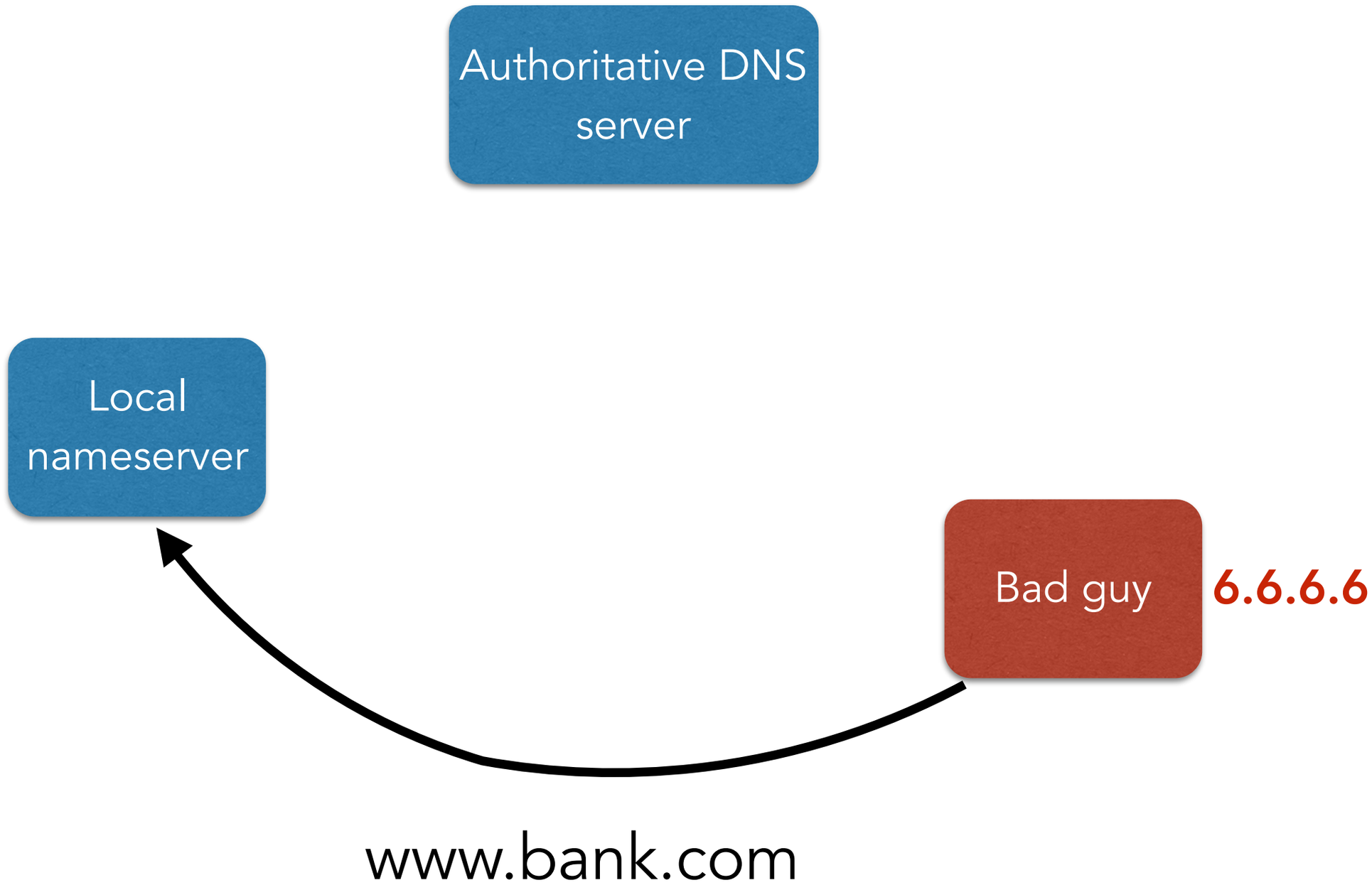
Local nameserver

Bad guy **6.6.6.6**

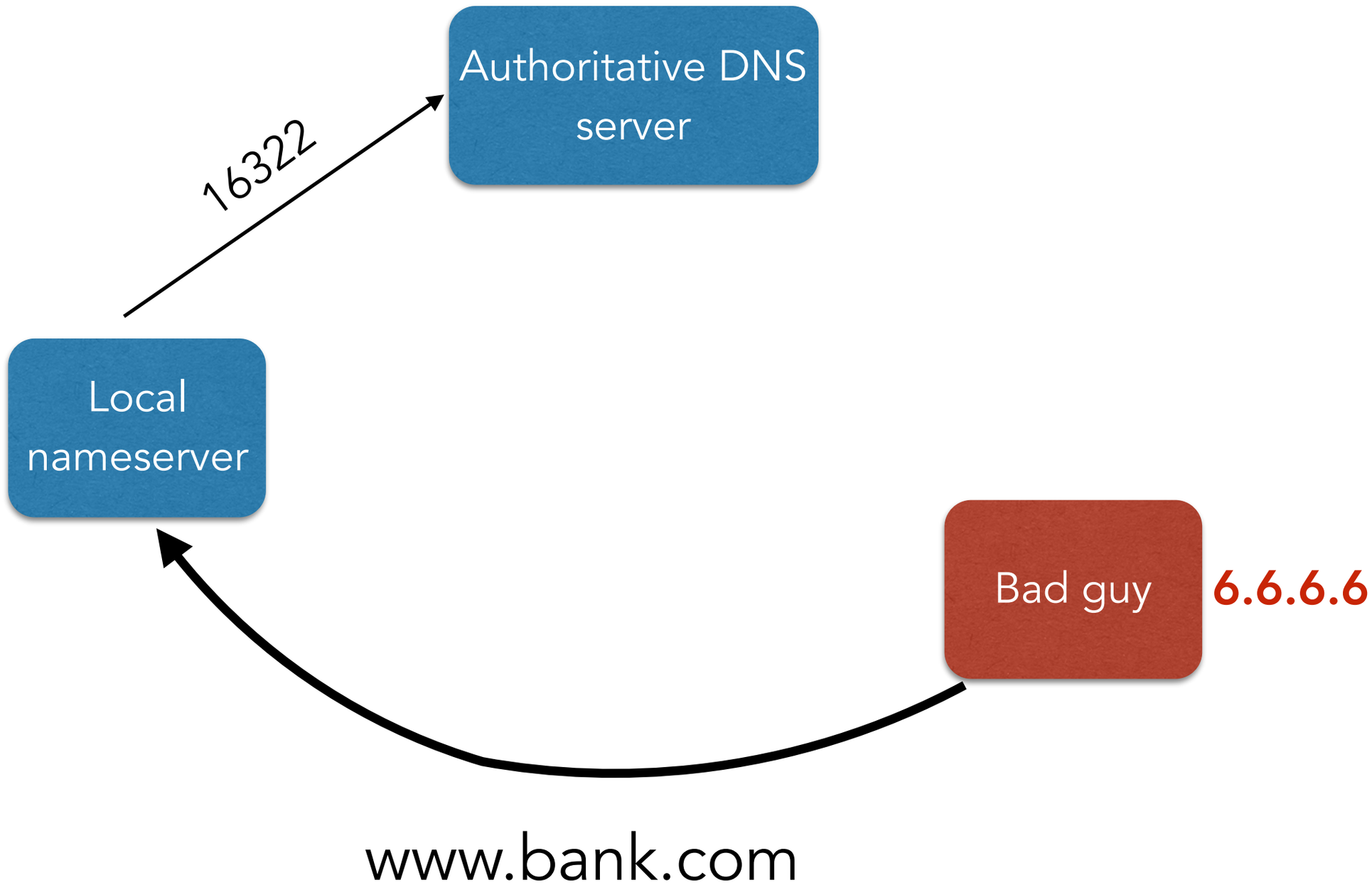
CACHE POISONING



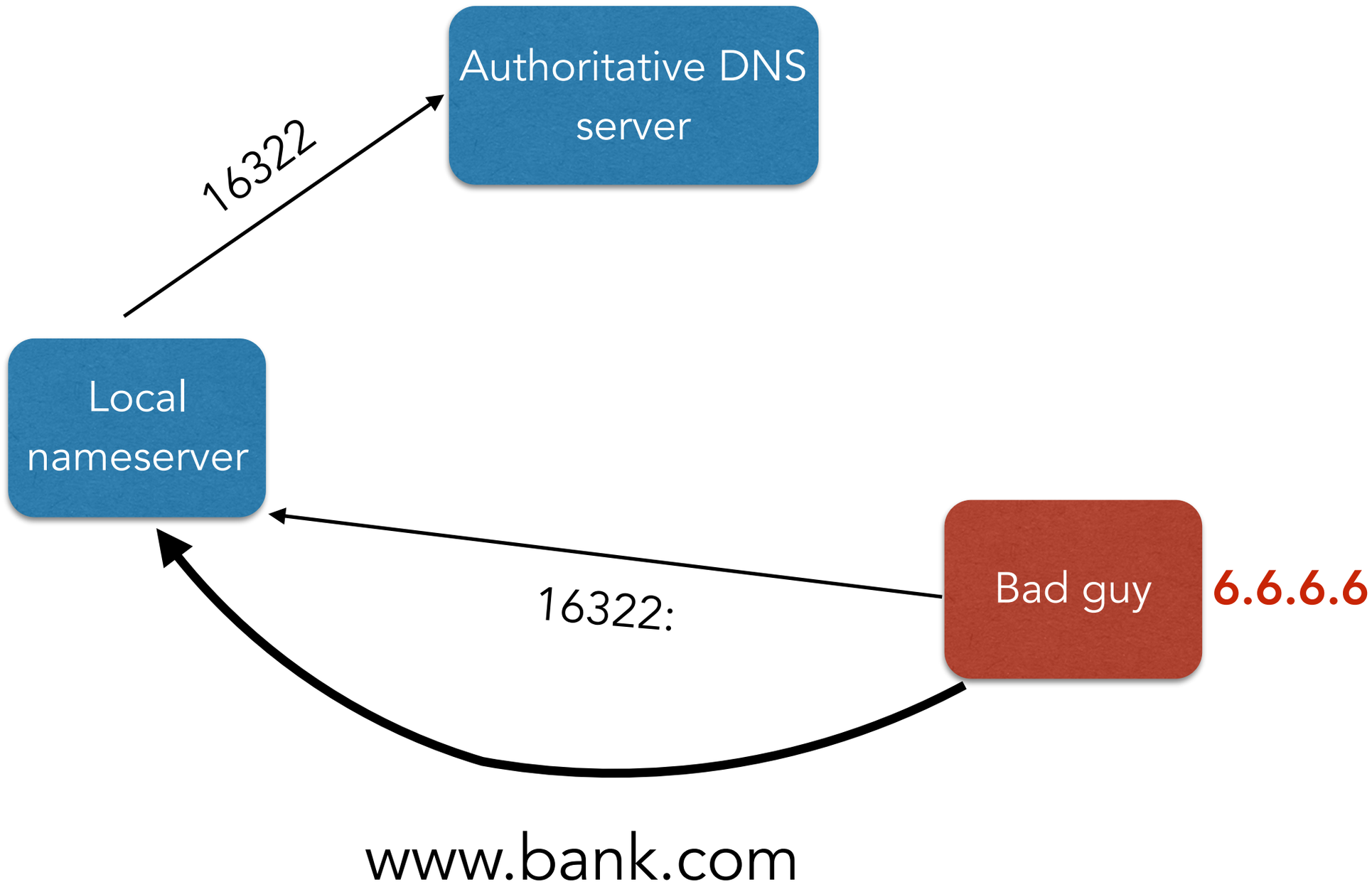
CACHE POISONING



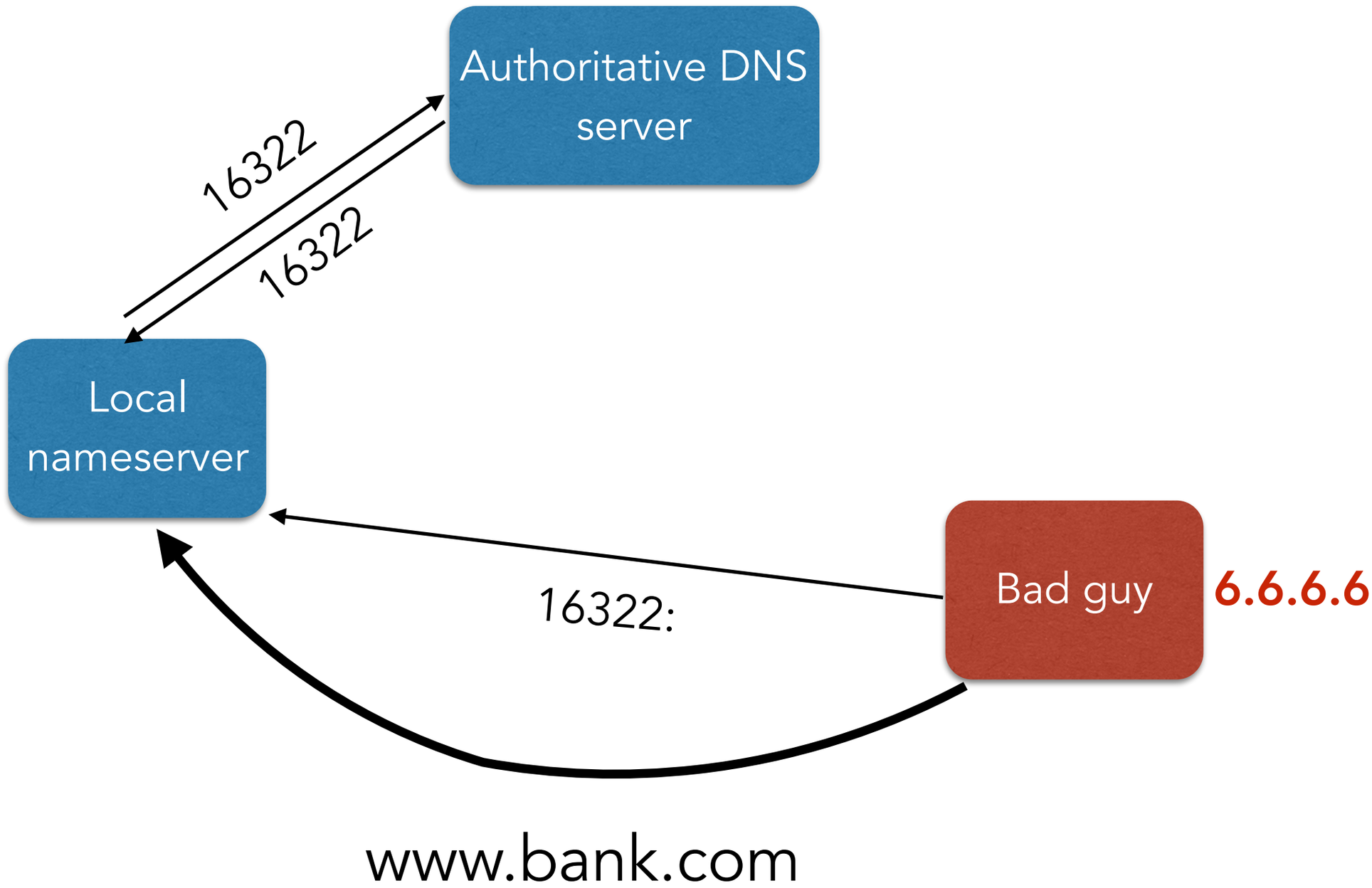
CACHE POISONING



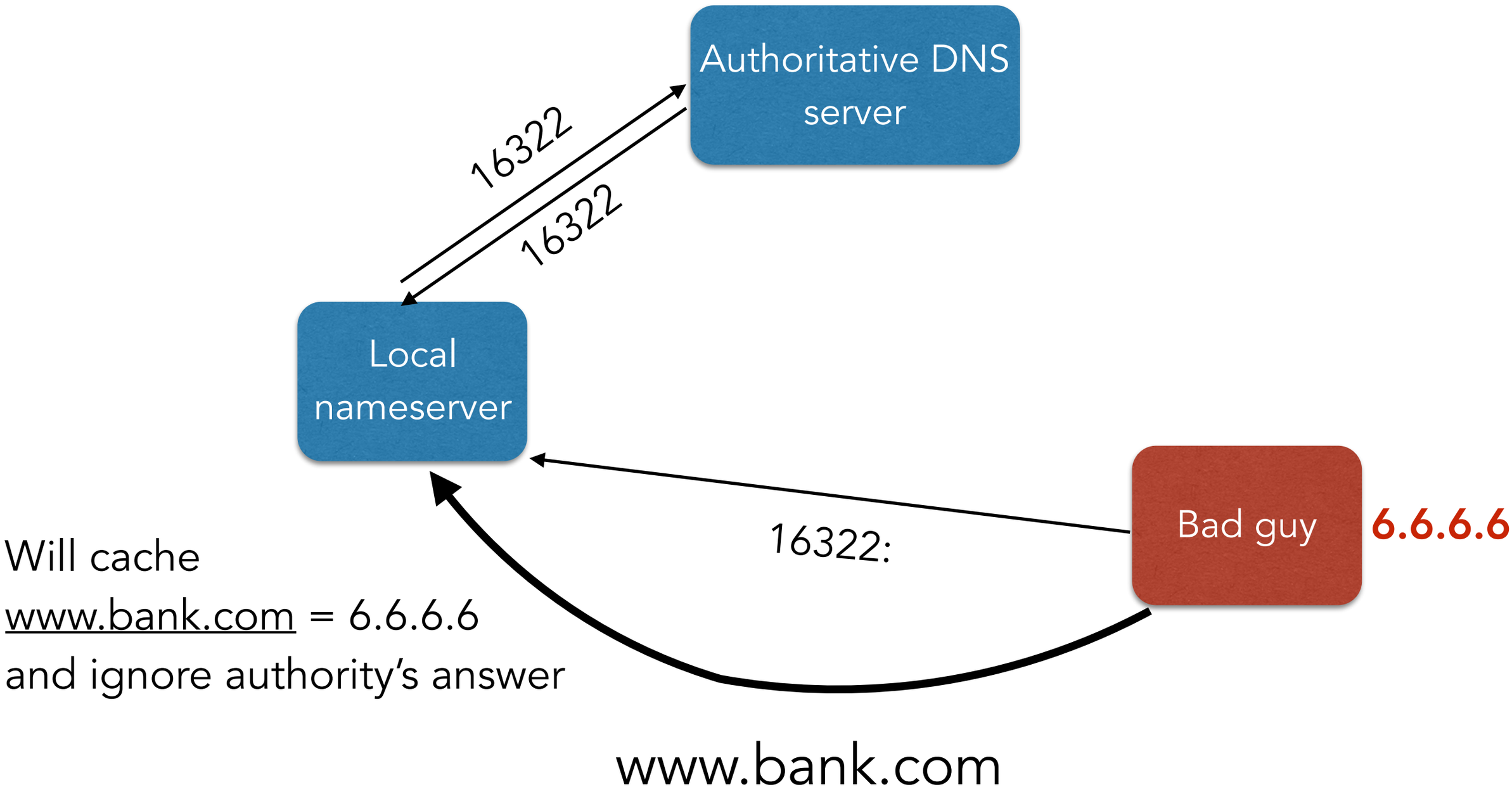
CACHE POISONING



CACHE POISONING

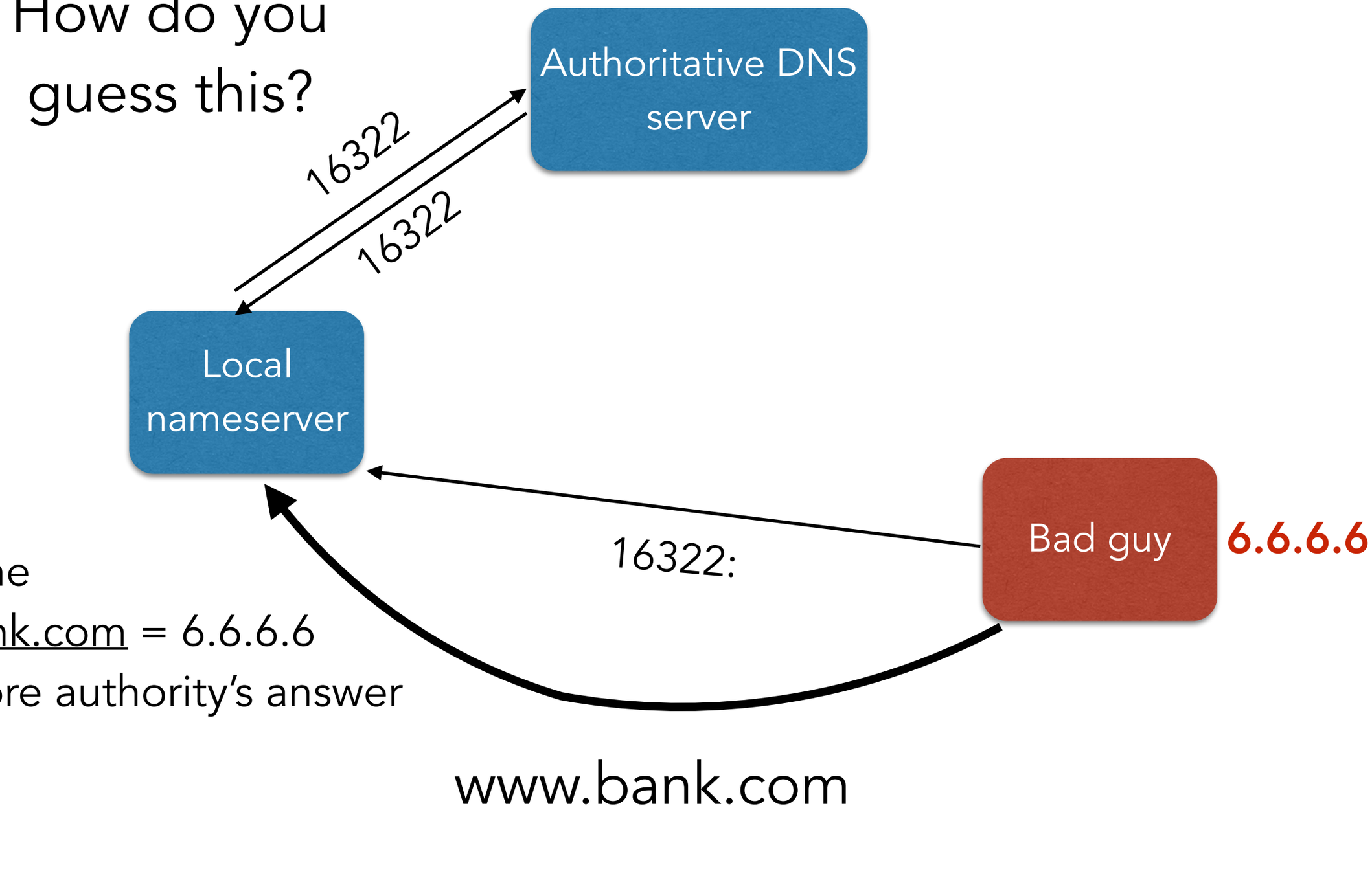


CACHE POISONING



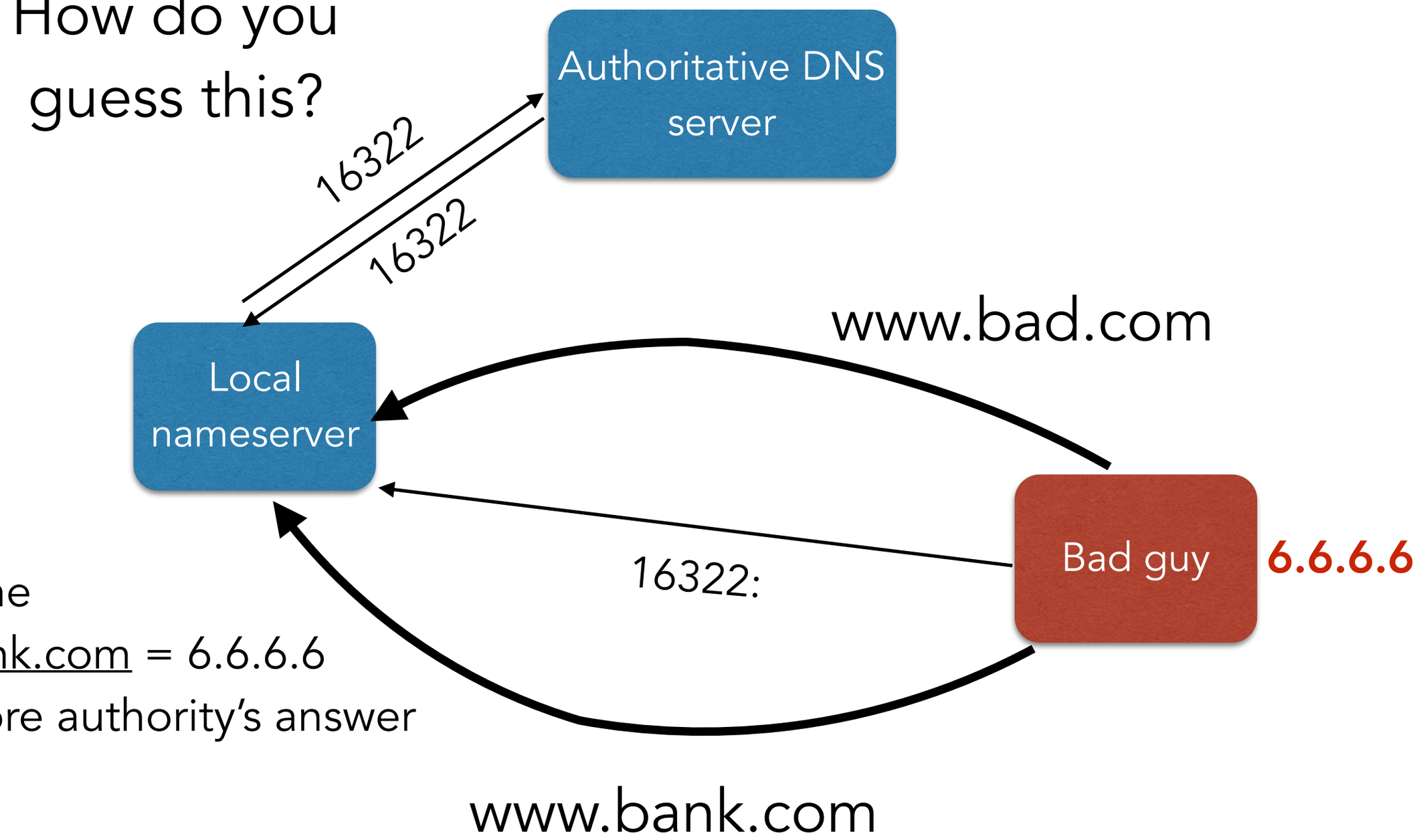
CACHE POISONING

How do you guess this?



CACHE POISONING

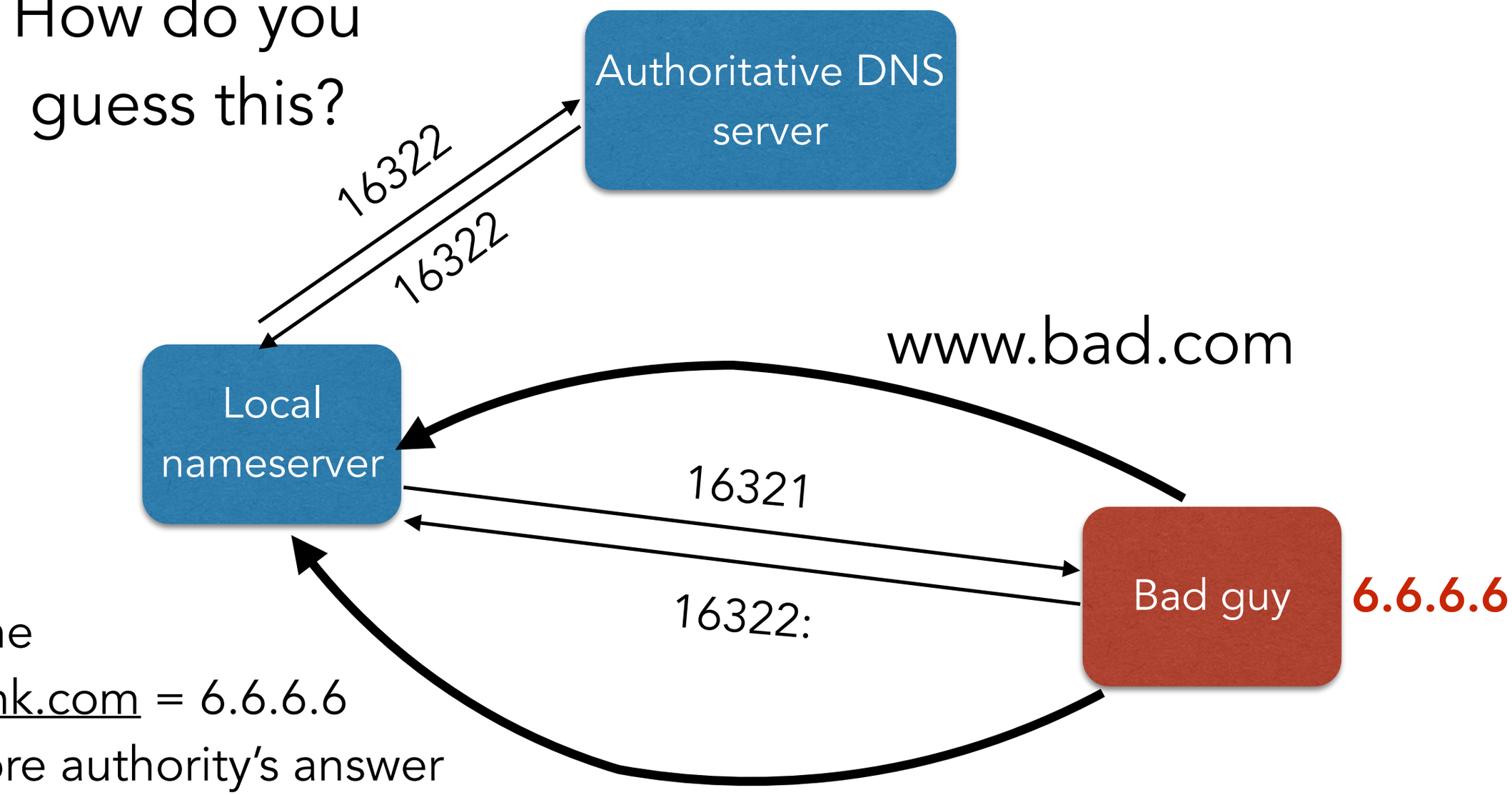
How do you guess this?



Will cache
www.bank.com = 6.6.6.6
and ignore authority's answer

CACHE POISONING

How do you guess this?

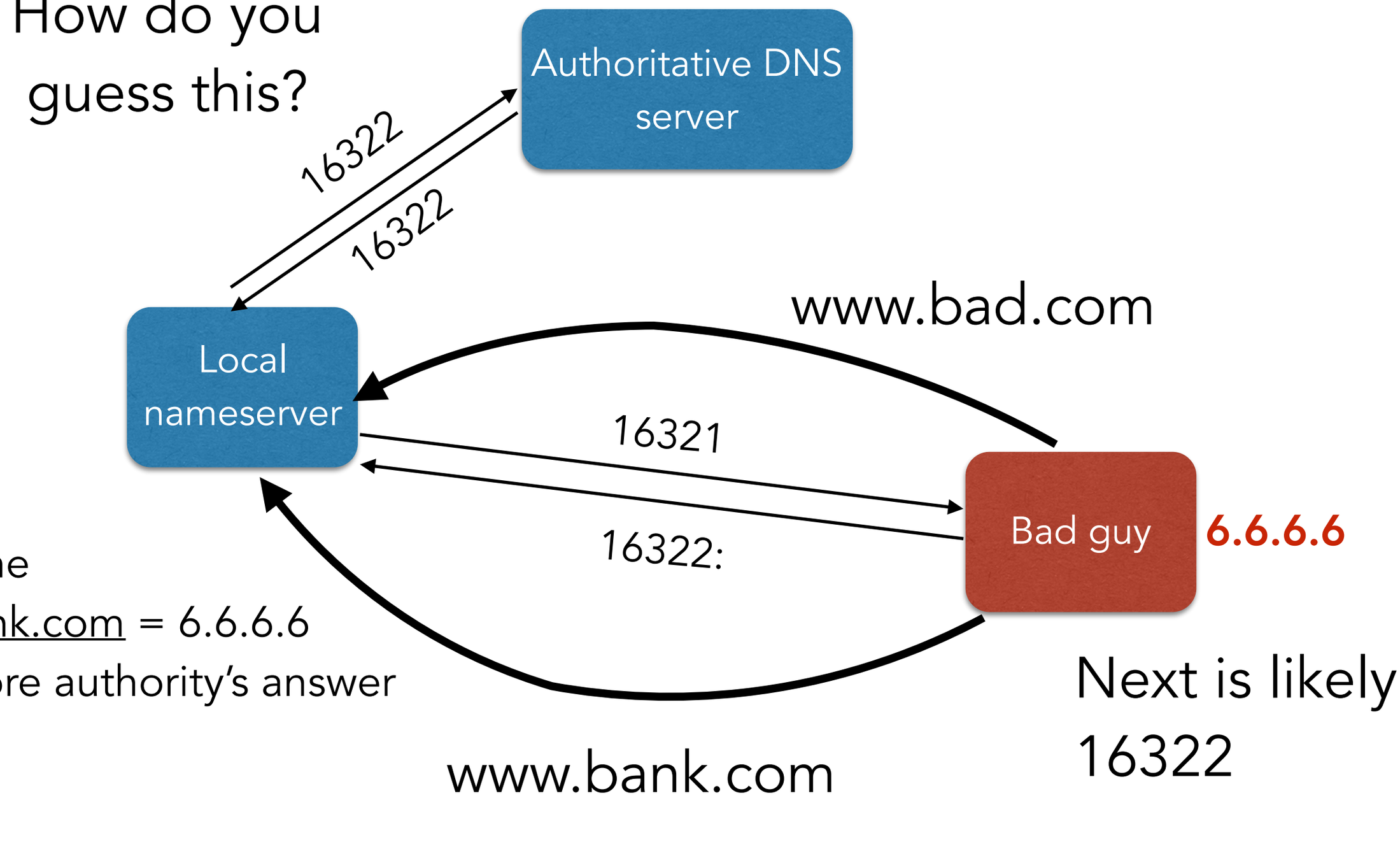


Will cache
www.bank.com = 6.6.6.6
and ignore authority's answer

www.bank.com

CACHE POISONING

How do you guess this?



DETAILS OF GETTING THE ATTACK TO WORK

- Must guess query ID: ask for it, and go from there
  - Partial fix: randomize query IDs
  - Problem: small space
  - Attack: issue a Lot of query IDs

- Must guess source port number
  - Typically constant for a given server (often always 53)

- The answer must not already be in the cache
  - It will avoid issuing a query in the first place

CACHE POISONING

Can we do more harm than a single record?

com. TLD

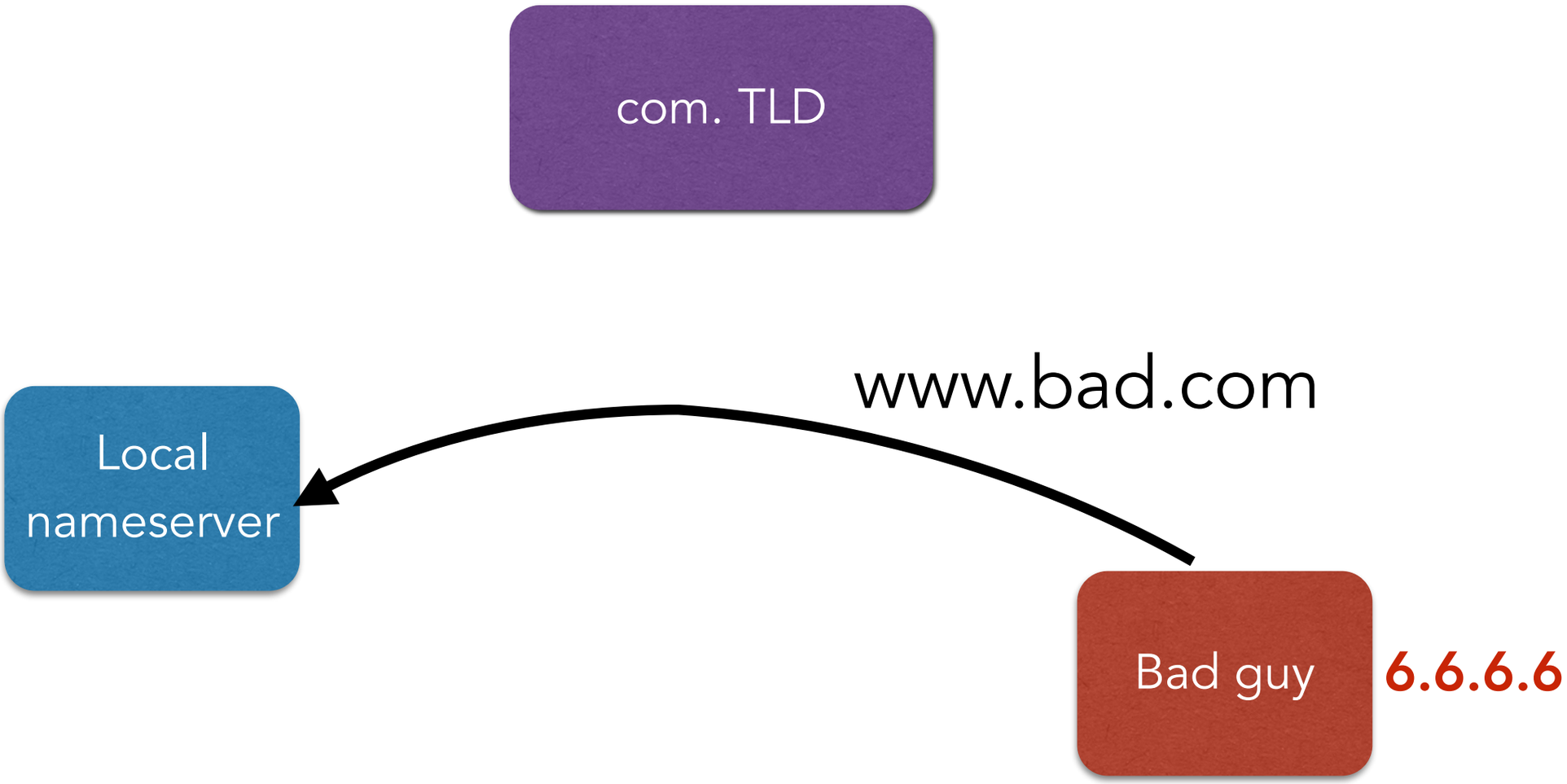
Local nameserver

Bad guy

6.6.6.6

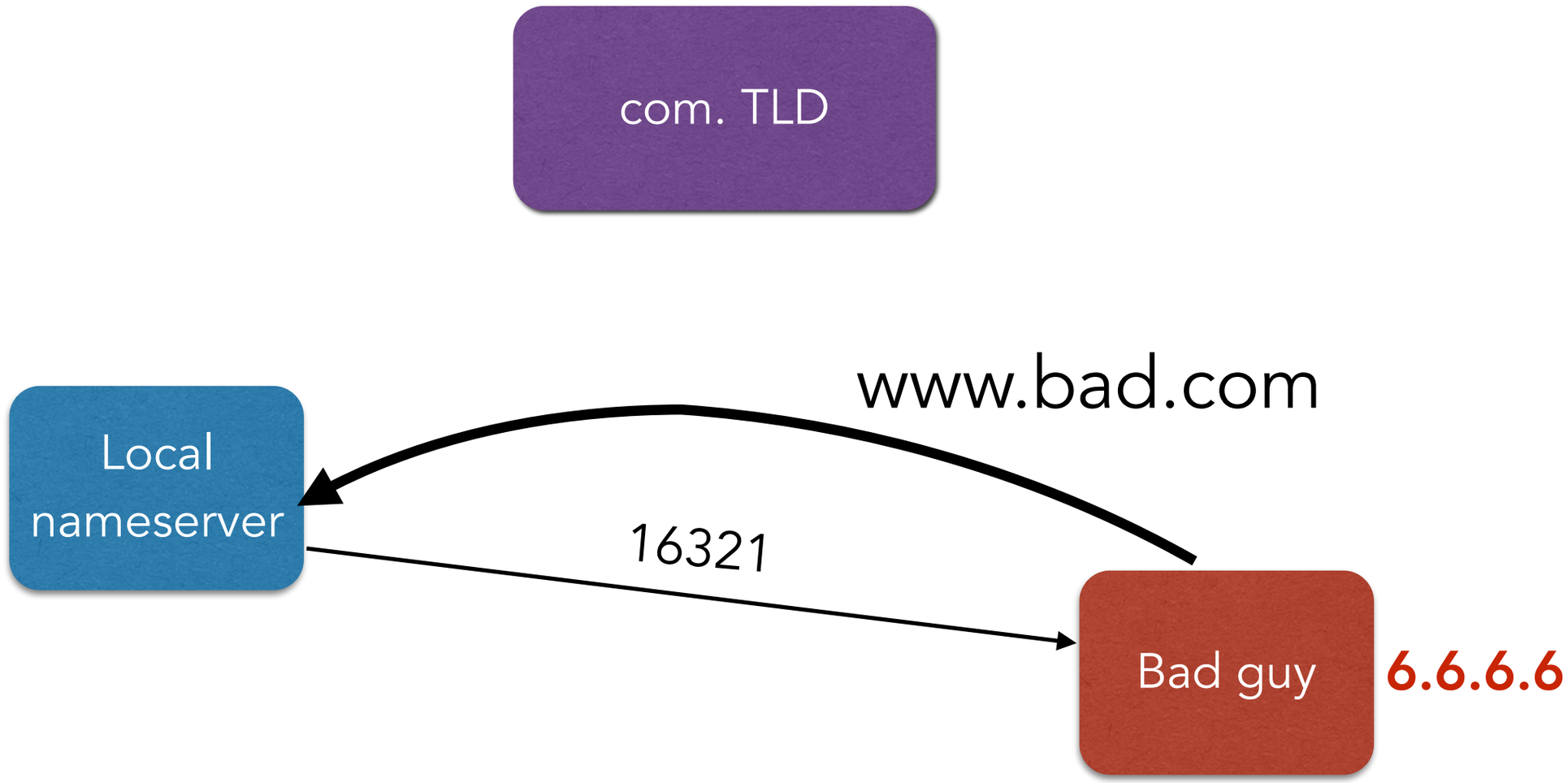
CACHE POISONING

Can we do more harm than a single record?



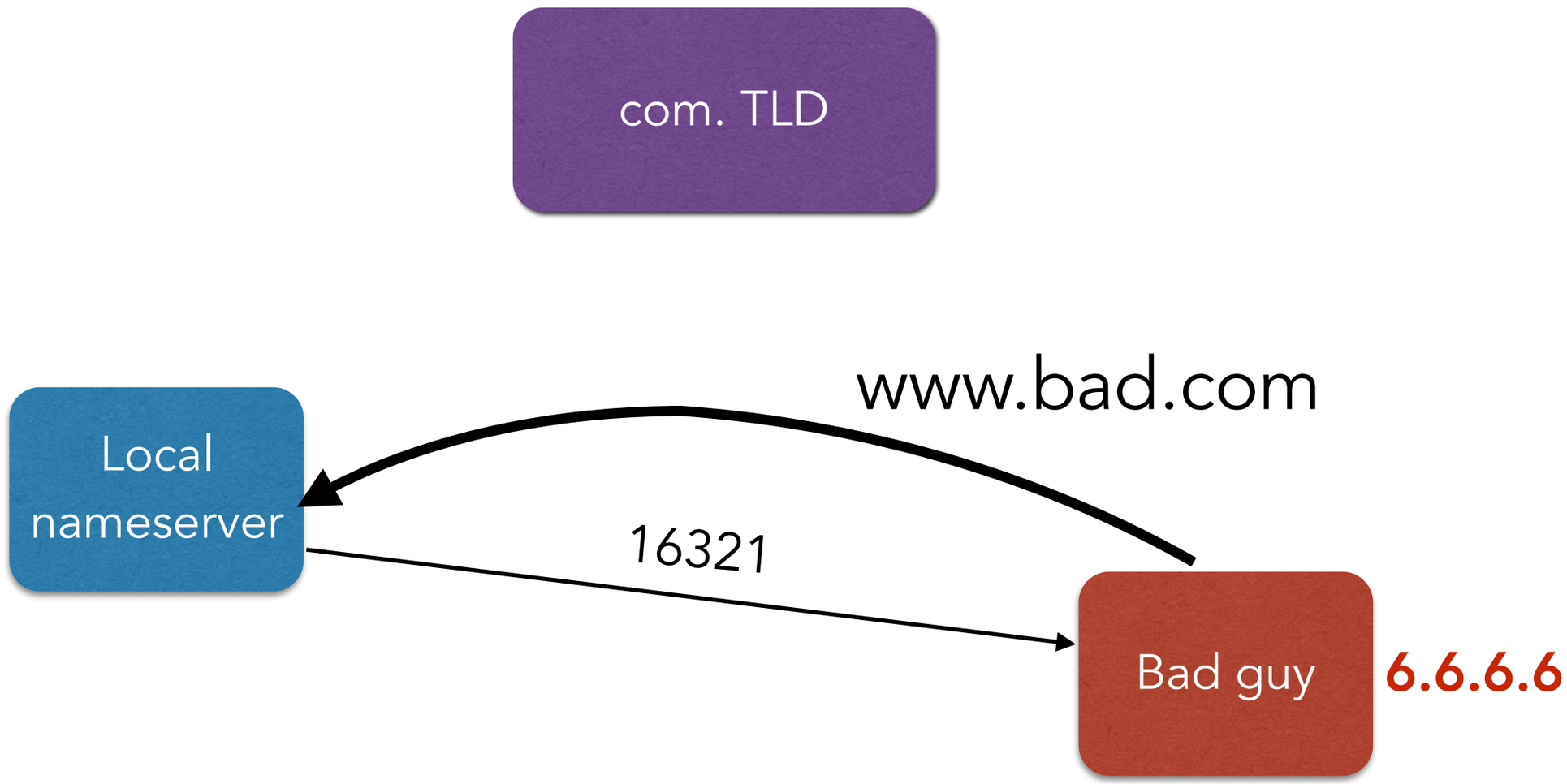
CACHE POISONING

Can we do more harm than a single record?



CACHE POISONING

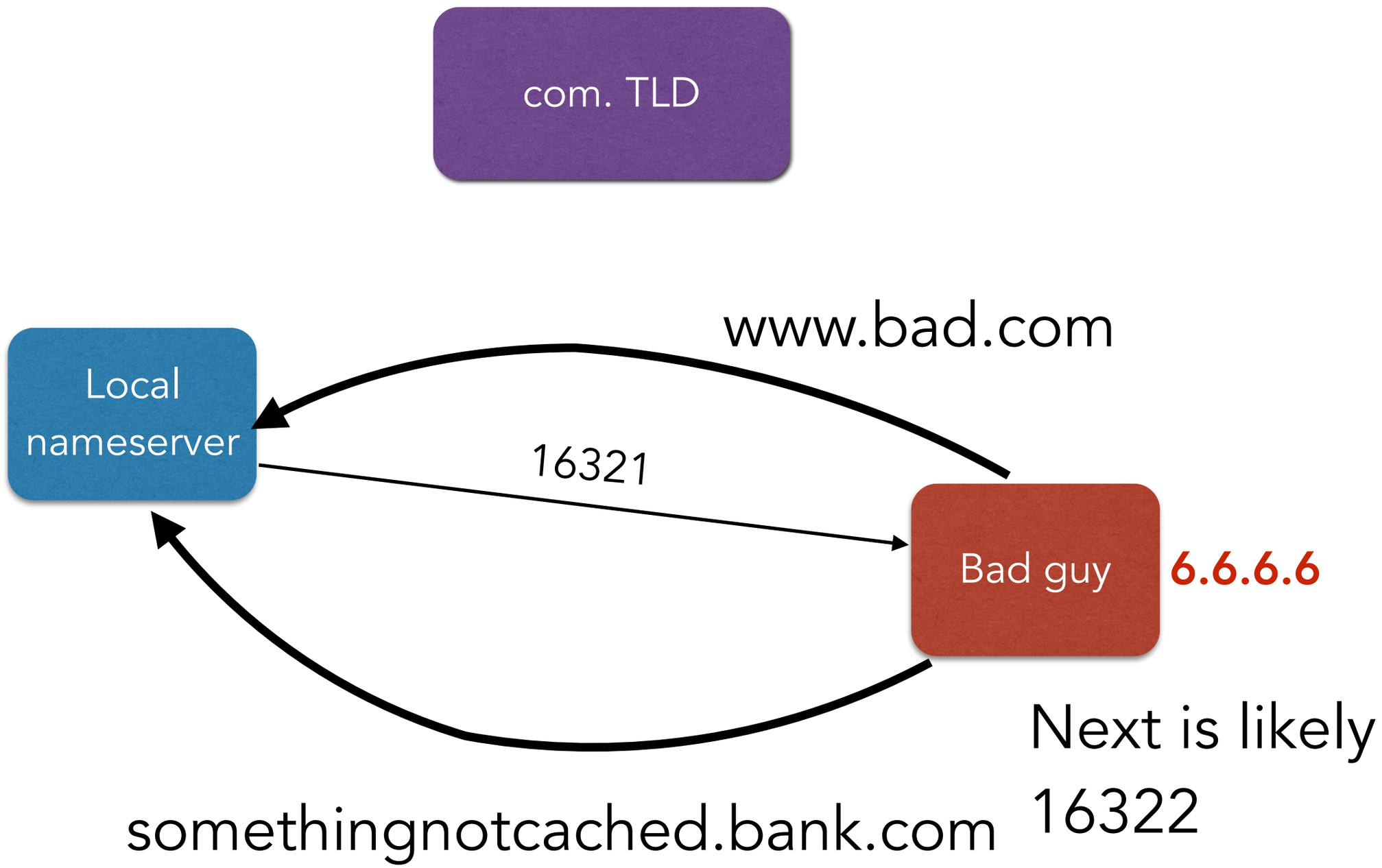
Can we do more harm than a single record?



Next is likely 16322

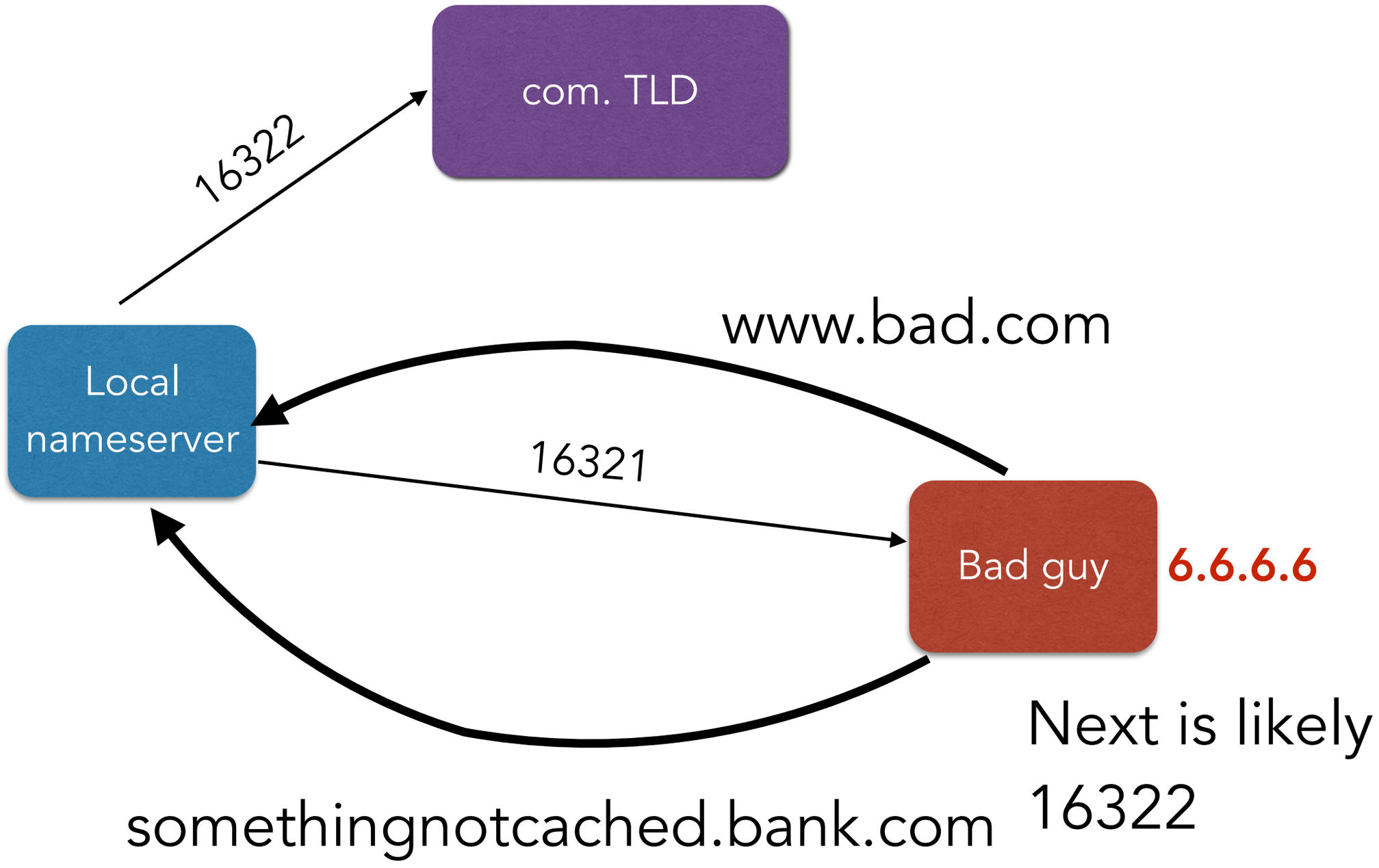
CACHE POISONING

Can we do more harm than a single record?



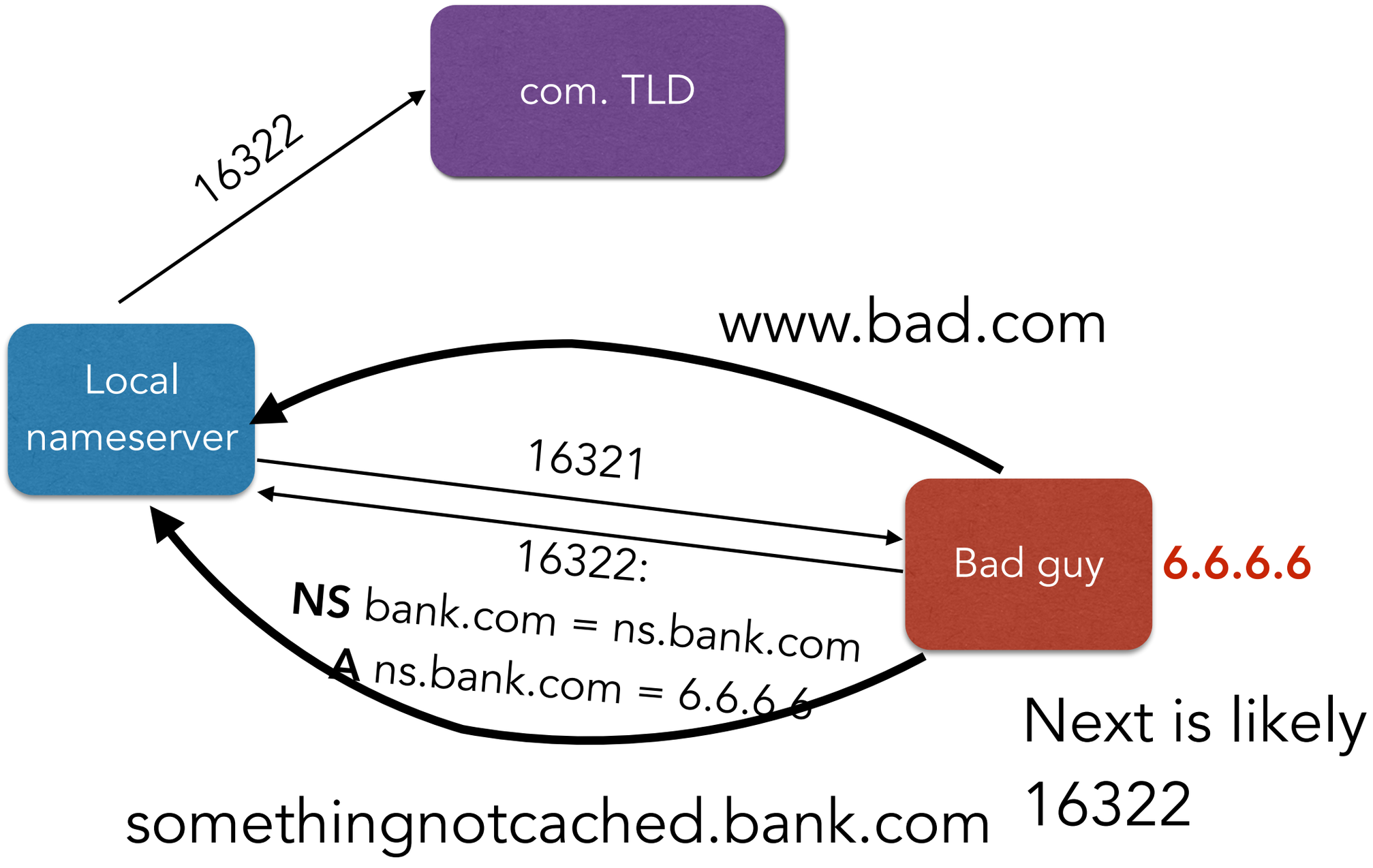
CACHE POISONING

Can we do more harm than a single record?



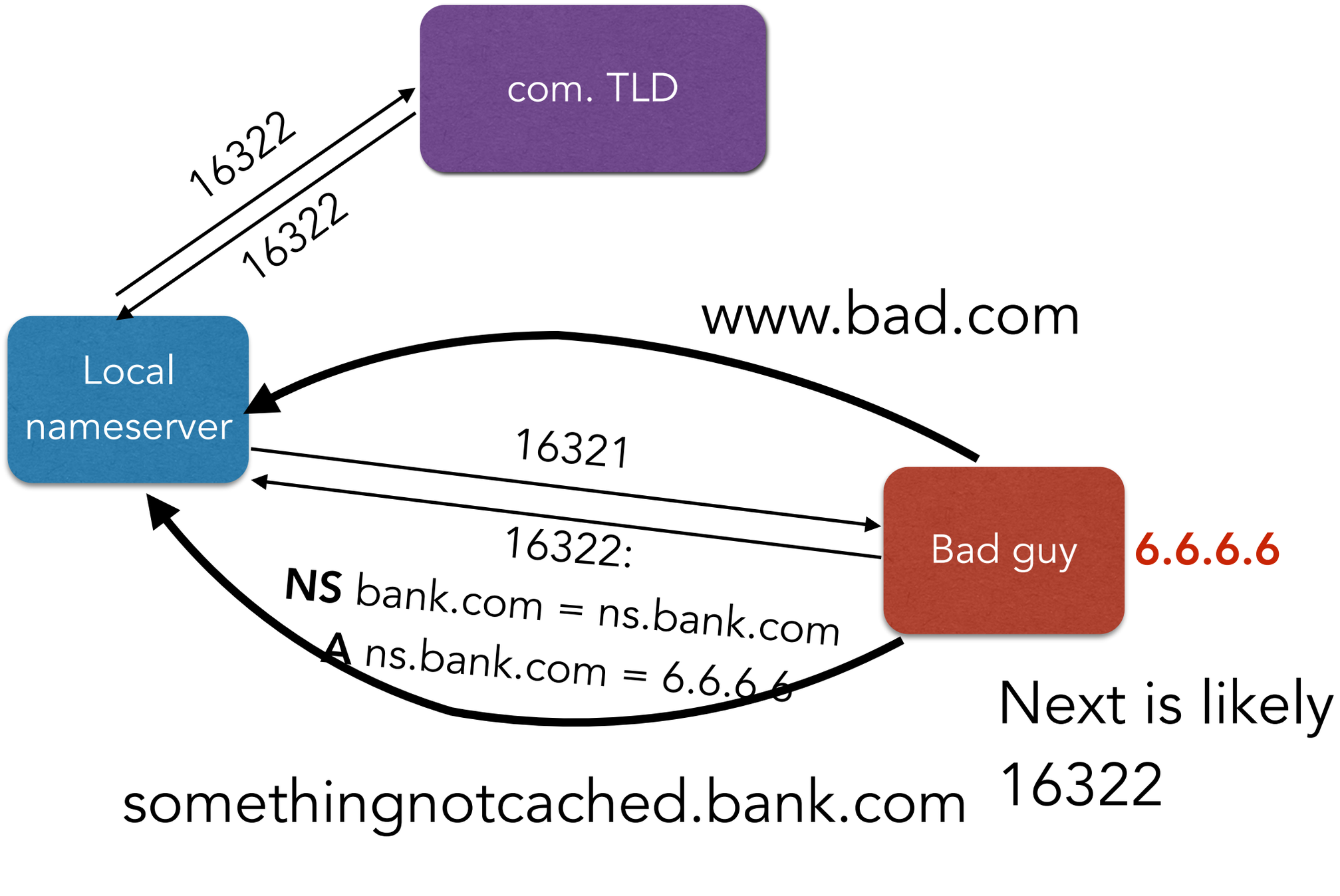
CACHE POISONING

Can we do more harm than a single record?



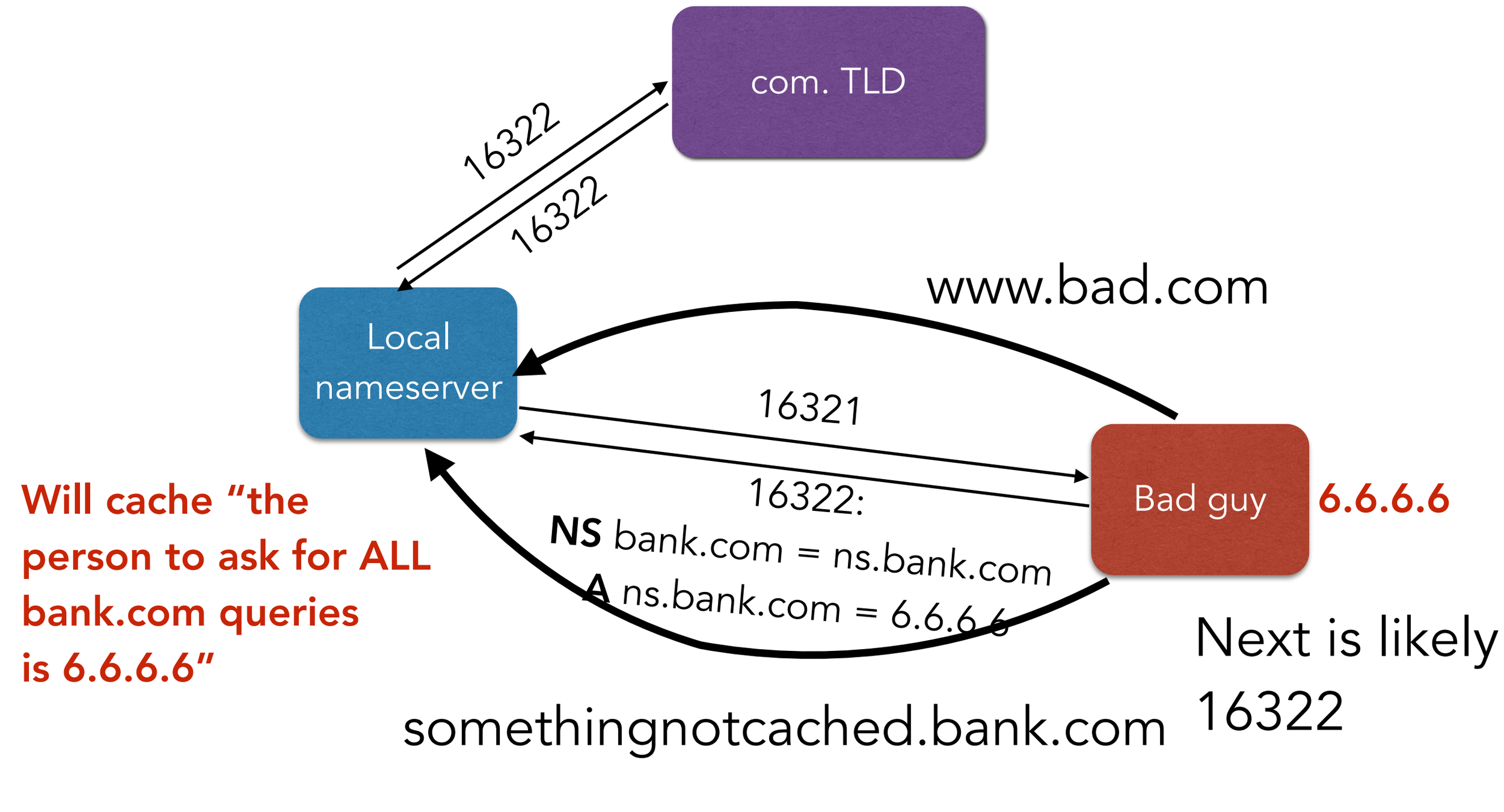
CACHE POISONING

Can we do more harm than a single record?



CACHE POISONING

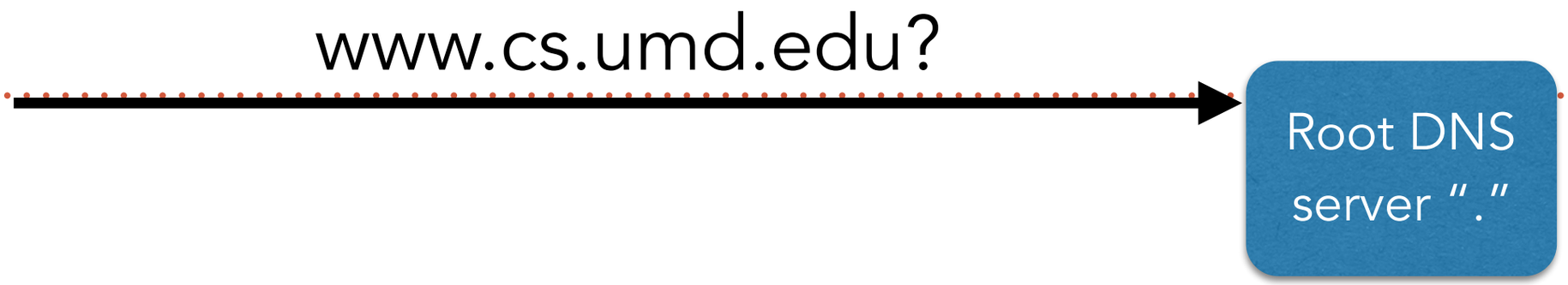
Can we do more harm than a single record?



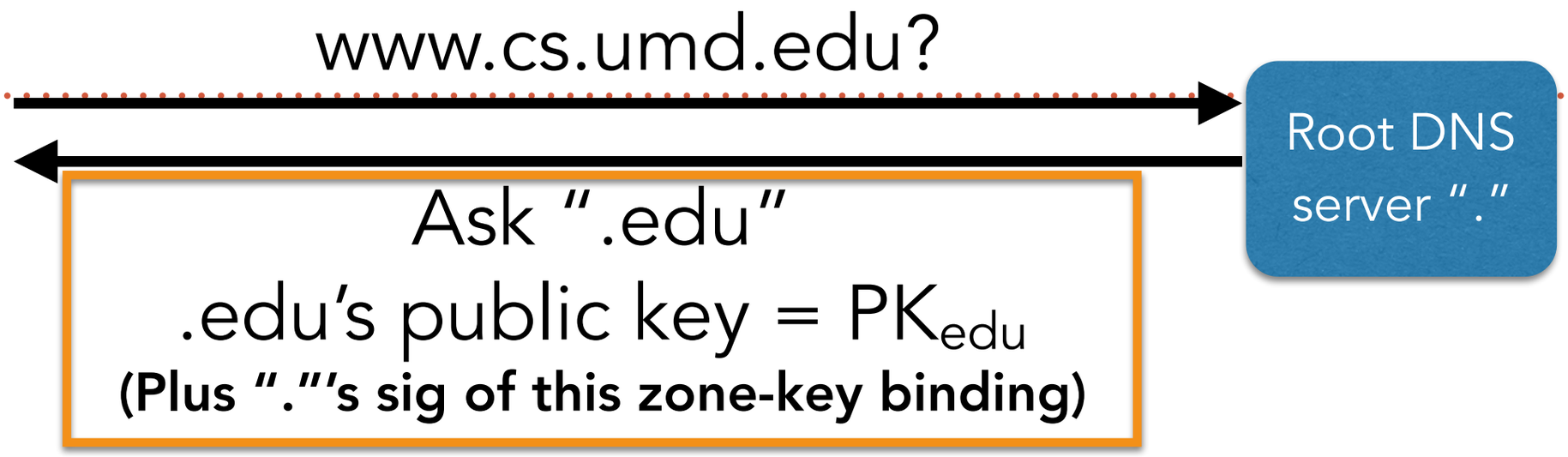
SOLUTIONS?

- Randomizing query ID?
  - Not sufficient alone: only 16 bits of entropy
- Randomize source port, as well
  - There’s no reason for it stay constant
  - Gets us another 16 bits of entropy
- DNSSEC?

DNSSEC



DNSSEC


www.cs.umd.edu?
Root DNS server "."
Ask ".edu"
.edu's public key = PK_{edu}
(Plus "."'s sig of this zone-key binding)

DNSSEC

www.cs.umd.edu?

Root DNS server "."

Ask ".edu"

.edu's public key = PK_{edu}

(Plus "."'s sig of this zone-key binding)

www.cs.umd.edu?

TLD DNS server

DNSSEC

www.cs.umd.edu?

Root DNS server "."

Ask ".edu"

.edu's public key = PK_{edu}

(Plus "."'s sig of this zone-key binding)

www.cs.umd.edu?

TLD DNS server

Ask "umd.edu"

umd.edu's public key = PK_{umd}

(Plus "edu"'s sig of this zone-key binding)

DNSSEC

www.cs.umd.edu?

Root DNS server "."

Ask ".edu"

.edu's public key = PK_{edu}

(Plus "."'s sig of this zone-key binding)

www.cs.umd.edu?

TLD DNS server

Ask "umd.edu"

umd.edu's public key = PK_{umd}

(Plus "edu"'s sig of this zone-key binding)

www.cs.umd.edu?

Authoritative DNS server

DNSSEC

www.cs.umd.edu?

Root DNS server "."

Ask ".edu"
.edu's public key = PK_{edu}
(Plus "."'s sig of this zone-key binding)

www.cs.umd.edu?

TLD DNS server

Ask "umd.edu"
umd.edu's public key = PK_{umd}
(Plus "edu"'s sig of this zone-key binding)

www.cs.umd.edu?

Authoritative DNS server

IN A www.cs.umd.edu 128.8.127.3
(Plus "umd.edu"'s signature of the *answer*

DNSSEC

www.cs.umd.edu?

Root DNS server "."

Ask ".edu"

.edu's public key = PK_{edu}

(Plus "."'s sig of this zone-key binding)

www.cs.umd.edu?

TLD DNS server

Ask "umd.edu"

umd.edu's public key = PK_{umd}

(Plus "edu"'s sig of this zone-key binding)

www.cs.umd.edu?

Authoritative DNS server

IN A www.cs.umd.edu 128.8.127.3

(Plus "umd.edu"'s signature of the *answer*

Only the authoritative answer is signed

PROPERTIES OF DNSSEC

- If everyone has deployed it, and if you know the root's keys, then prevents spoofed responses
  - Very similar to PKIs in this sense

- But unlike PKIs, we still want authenticity despite the fact that not everyone has deployed DNSSEC
  - What if someone replies back without DNSSEC?
  - Ignore = secure but you can't connect to a lot of hosts
  - Accept = can connect but insecure

- Back to our notion of incremental deployment
  - DNSSEC is not all that useful incrementally