TCP Congestion Control Advanced Techniques (Contd.)

CPSC 433/533, Spring 2021

Anurag Khandelwal

Router-assisted Congestion Control

- **Three tasks for CC:**
  - Isolation/fairness
  - Adjustment
  - Detecting congestion

Max-Min Fairness

- Given a set of bandwidth demands r_i and total bandwidth C, max-min bandwidth allocations are:

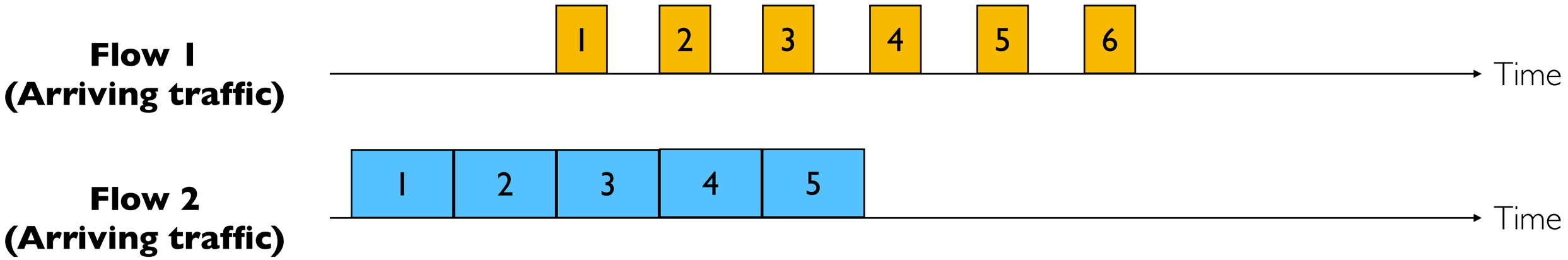
$$a_i = min(f, r_i)$$

  where f is the unique value such that $\sum_i a_i = C$

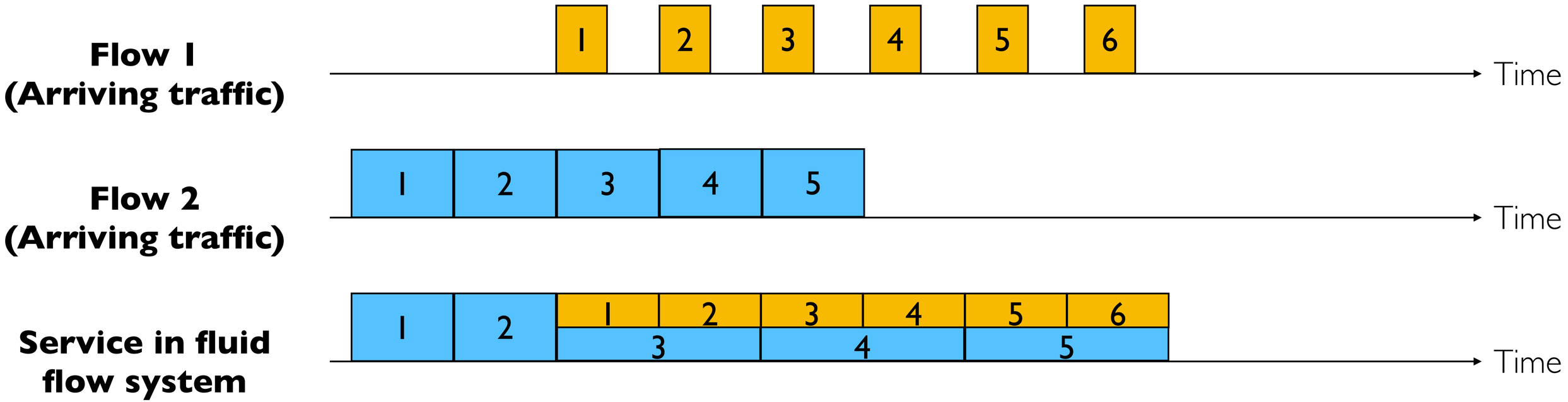
- Property:
  - If you don't get full demand, no one gets more than you

- This is what round robin service gives if all packets are the same size

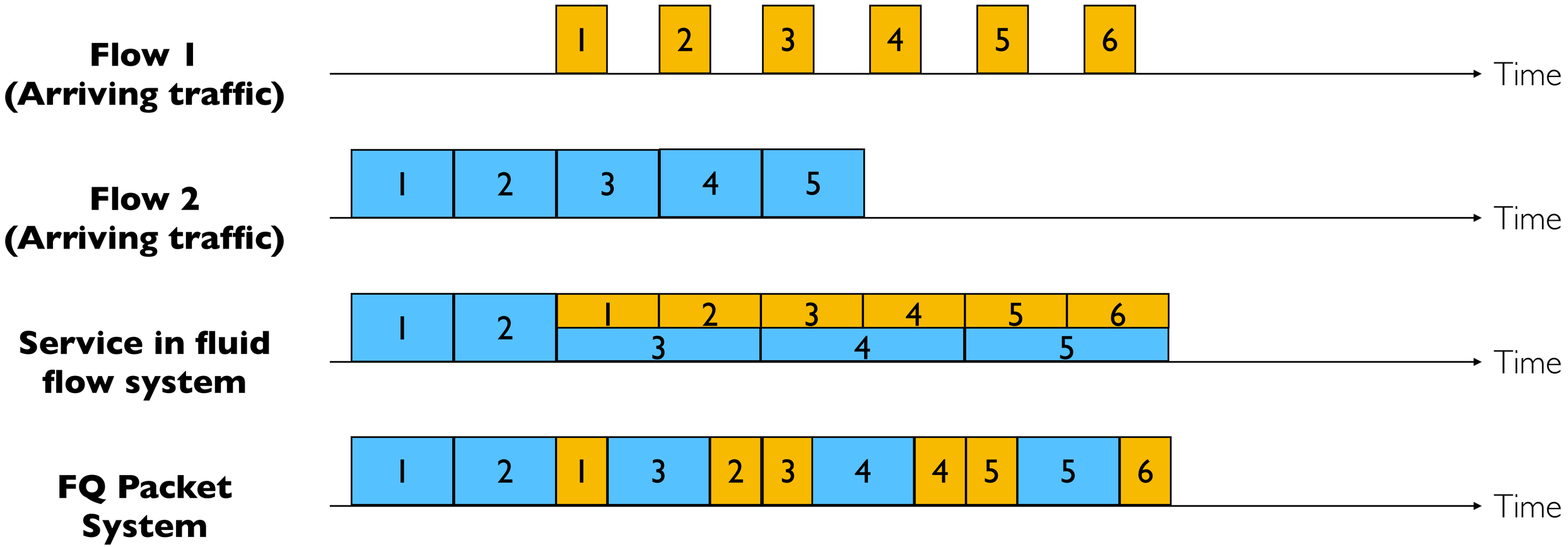
Practical Fair Queueing (FQ)



Practical Fair Queueing (FQ)



Practical Fair Queueing (FQ)



FQ vs. FIFO

FQ vs. FIFO

- **FQ Advantages:**
  - Isolation: cheating flows don't benefit
  - Bandwidth share does not depend on RTT
  - Flows can pick any rate adjustment scheme they want

FQ vs. FIFO

- **FQ Advantages:**
  - Isolation: cheating flows don't benefit
  - Bandwidth share does not depend on RTT
  - Flows can pick any rate adjustment scheme they want

- **Disadvantages**
  - More complex than FIFO: per flow queue/state, additional per-packet book-keeping

Router-assisted Congestion Control

- Three tasks for CC:
  - Isolation/fairness
  - Adjustment
  - Detecting congestion

Why not just let routers tell end-systems what rate they should use?

Why not just let routers tell end-systems what rate they should use?

- Packets carry "rate field"

Why not just let routers tell end-systems what rate they should use?

- Packets carry “rate field”
- Routers insert “fair share” f in the packet header

Why not just let routers tell end-systems what rate they should use?

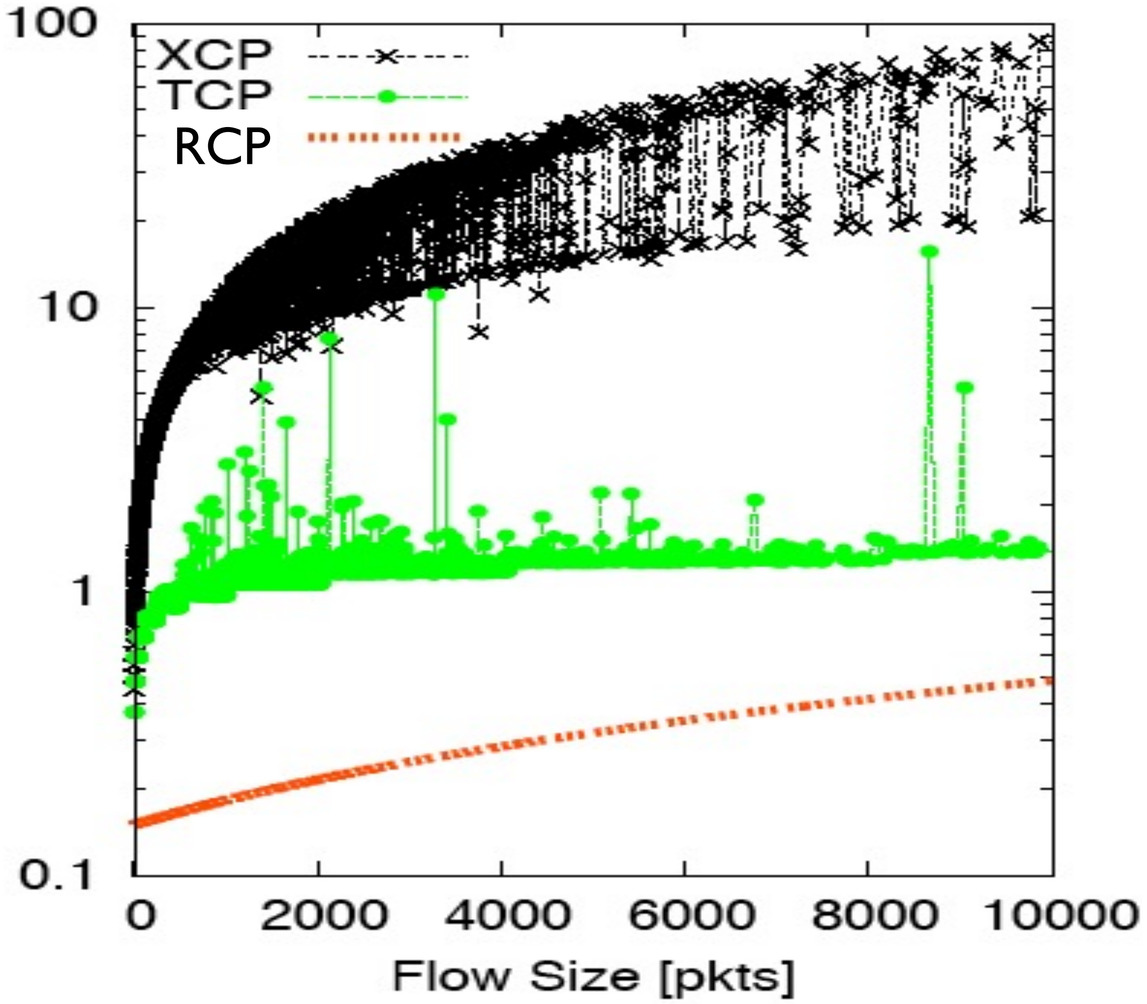
- Packets carry "rate field"
- Routers insert "fair share" f in the packet header
- End-systems set sending rate (or window size) to f
  - Hopefully (still need some policing of end-systems!)

Why not just let routers tell end-systems what rate they should use?

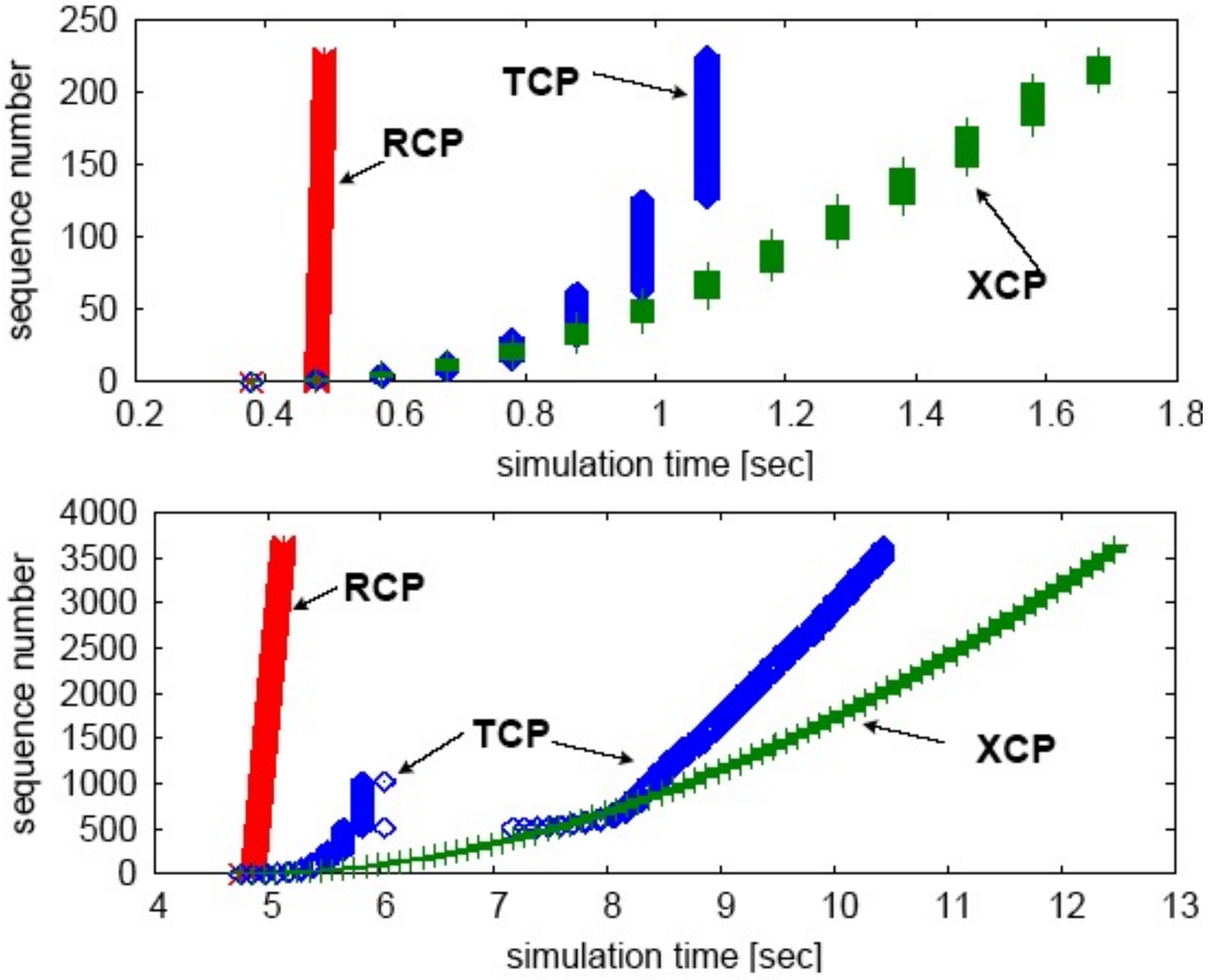
- Packets carry "rate field"
- Routers insert "fair share" f in the packet header
- End-systems set sending rate (or window size) to f
  - Hopefully (still need some policing of end-systems!)
- *Basic idea behind the "Rate Control Protocol" (RCP) from Dukkipati et. al.'07*

Flow Completion Time: TCP vs. RCP (Ignore XCP)

Flow Duration (seconds) vs. Flow Size



Why the Improvement?



Router-assisted Congestion Control

- Three tasks for CC:
  - Isolation/fairness
  - Adjustment
  - Detecting congestion

Explicit Congestion Notification (ECN)

Explicit Congestion Notification (ECN)

- **Single bit in packet header; set by congested routers**
  - If data packet has bit set, then ACK has ECN bit set

Explicit Congestion Notification (ECN)

- **Single bit in packet header; set by congested routers**
  - If data packet has bit set, then ACK has ECN bit set

- **Congestion semantics can be exactly like that of drop**
  - i.e., end-system reacts as though it saw a drop

Explicit Congestion Notification (ECN)

- **Single bit in packet header; set by congested routers**
  - If data packet has bit set, then ACK has ECN bit set

- **Congestion semantics can be exactly like that of drop**
  - i.e., end-system reacts as though it saw a drop

- **Advantages:**
  - Don't confuse corruption with congestion; recovery with rate adjustment
  - Can serve as an early indicator of congestion to avoid delays
  - Easy (easier) to incrementally deploy
    - Today: defined in RFC 3168 using **ToS/DSCP** bits in the IP header

One final proposal: Charge people for congestion!

One final proposal: Charge people for congestion!

- Use ECN as congestion markers

One final proposal: Charge people for congestion!

- Use ECN as congestion markers
- Whenever I get an ECN bit set, I have to pay $$

One final proposal: Charge people for congestion!

- Use ECN as congestion markers
- Whenever I get an ECN bit set, I have to pay $$
- Now there's no debate over what a flow is, or what fair is…

One final proposal: Charge people for congestion!

- Use ECN as congestion markers
- Whenever I get an ECN bit set, I have to pay $$
- Now there's no debate over what a flow is, or what fair is…
- Idea started by Frank Kelly at Cambridge
  - "Optimal" solution, backed by much math
  - Great idea: simple, elegant, effective
  - Unclear that it will impact practice

Recap: TCP

Recap: TCP

- TCP
  - Somewhat hacky
  - But practical/deployable
  - Good enough for Internet traffic
  - Needs of data centers might change status quo (future lecture)

Taking Stock: Transport Layer

Taking Stock: Transport Layer

- **Communication between application processes**
  - Multiplex and demultiplex from/to application processes using ***ports***

Taking Stock: Transport Layer

- **Communication between application processes**

- **Provide common end-to-end services for application layer [optional]**
  - Reliable, in-order data delivery
  - Well-placed data delivery
    - Too fast may overwhelm the receiver/network
    - Too slow is not efficient

Taking Stock: Transport Layer

- **Communication between application processes**
- **Provide common end-to-end services for application layer [optional]**
- **TCP and UDP are the common transport protocols**
  - Also SCTP, MTCP, SST, RDP, DCCP, …

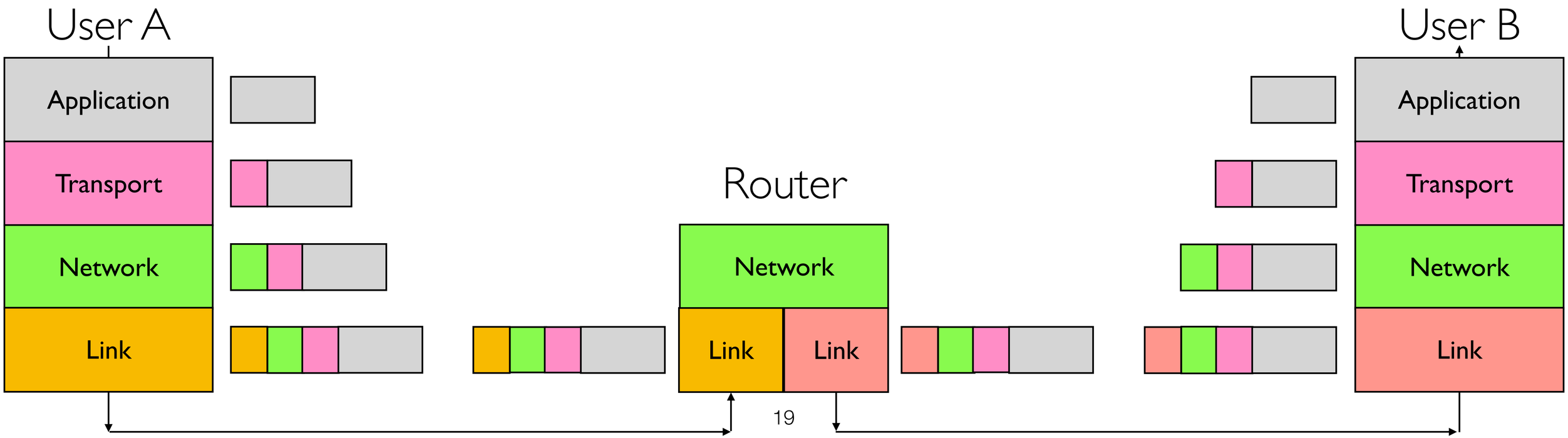
Taking Stock: Transport Layer

- **Communication between application processes**
- **Provide common end-to-end services for application layer [optional]**
- **TCP and UDP are the common transport protocols**
- **UDP is a minimalist, no-frills transport protocol**
  - Only provides multiplex/demultiplex capabilities

Taking Stock: Transport Layer

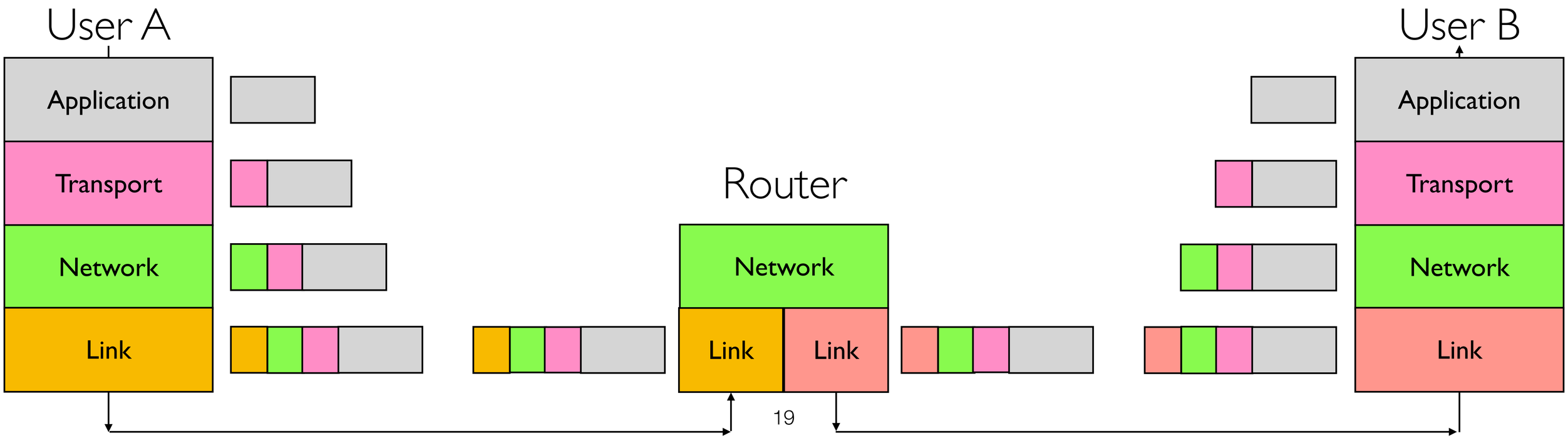
- **Communication between application processes**
- **Provide common end-to-end services for application layer [optional]**
- **TCP and UDP are the common transport protocols**
- **UDP is a minimalist, no-frills transport protocol**
- **TCP is the whole-hog protocol**
  - Offers applications a reliable, in-order, byte stream abstraction
  - With congestion control
  - But no performance guarantees (delay, bw, etc.)

Taking Stock



Taking Stock

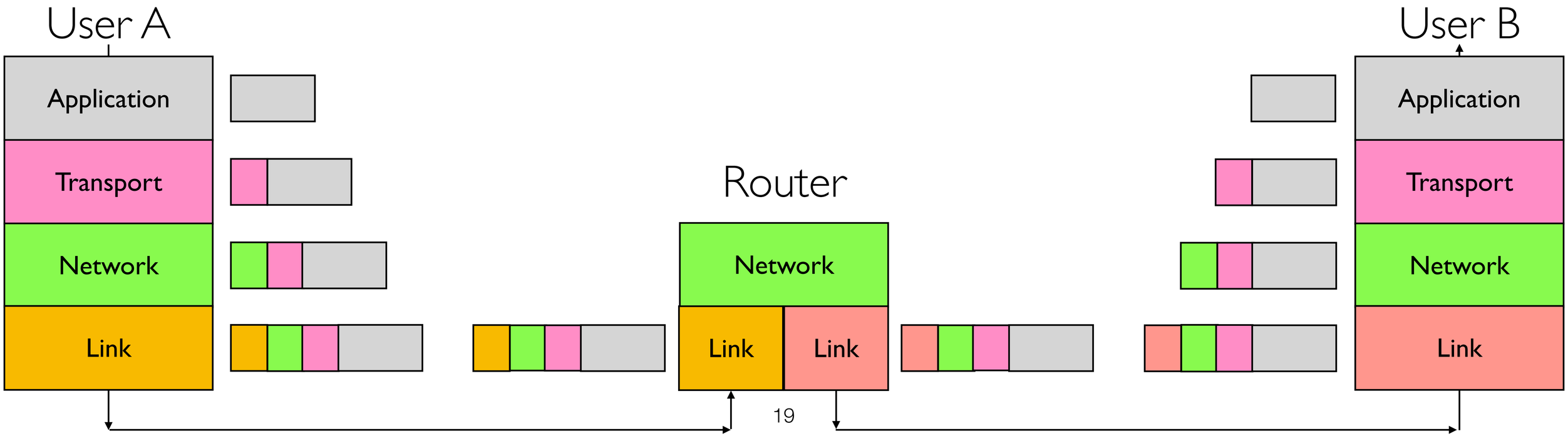
Course so far:



Taking Stock

Course so far:

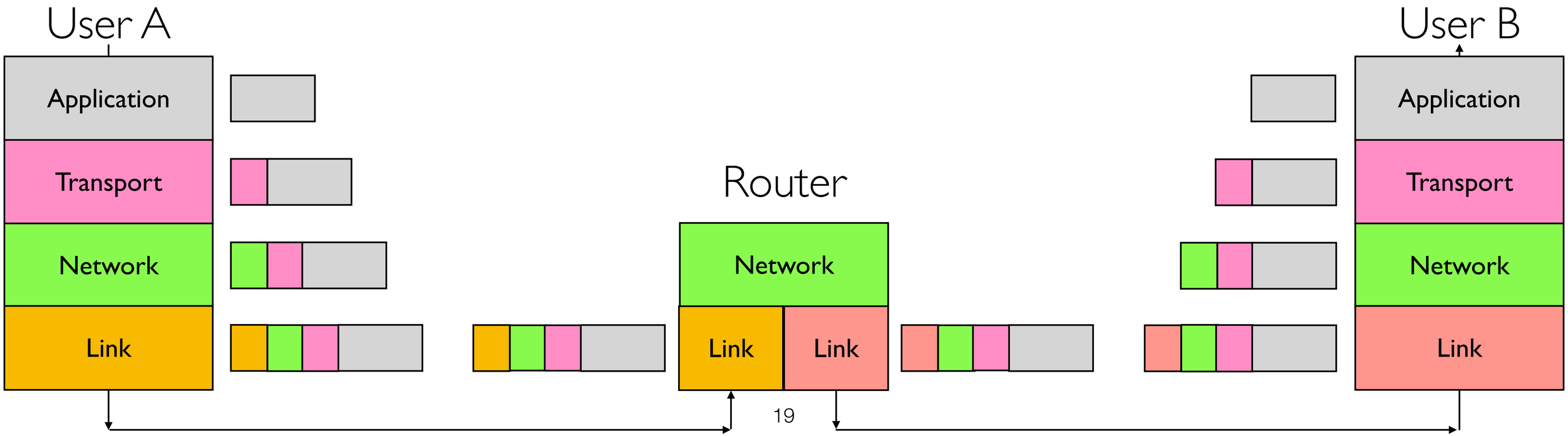
- **Concepts,** *Links, delays, switches*



Taking Stock

Course so far:

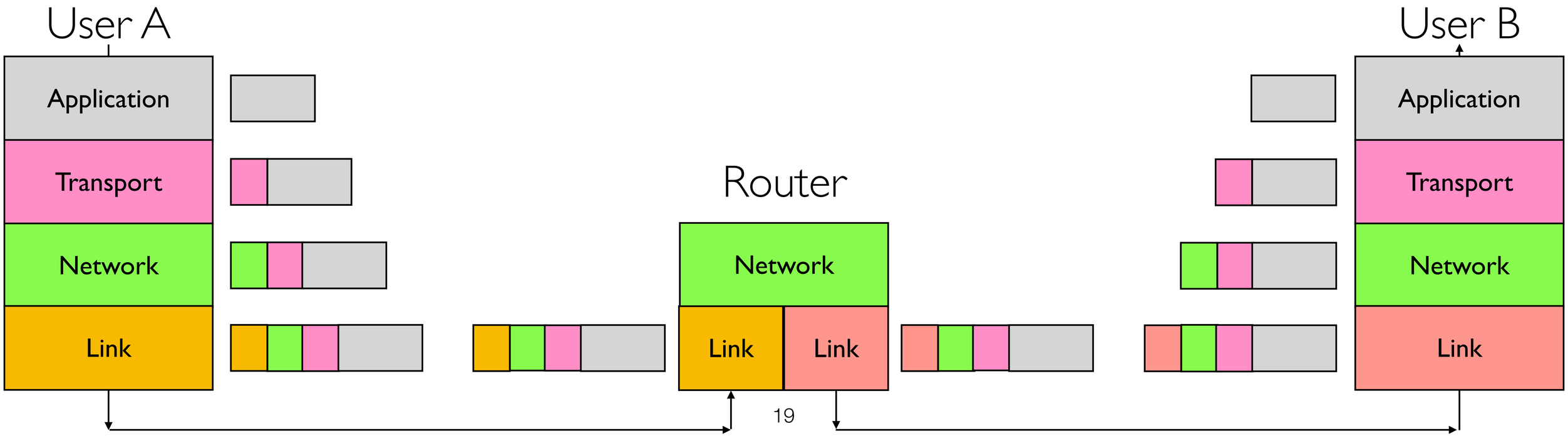
- **Concepts,** *Links, delays, switches*
- **Overall Architecture,** *Layers, protocols principles*



Taking Stock

Course so far:

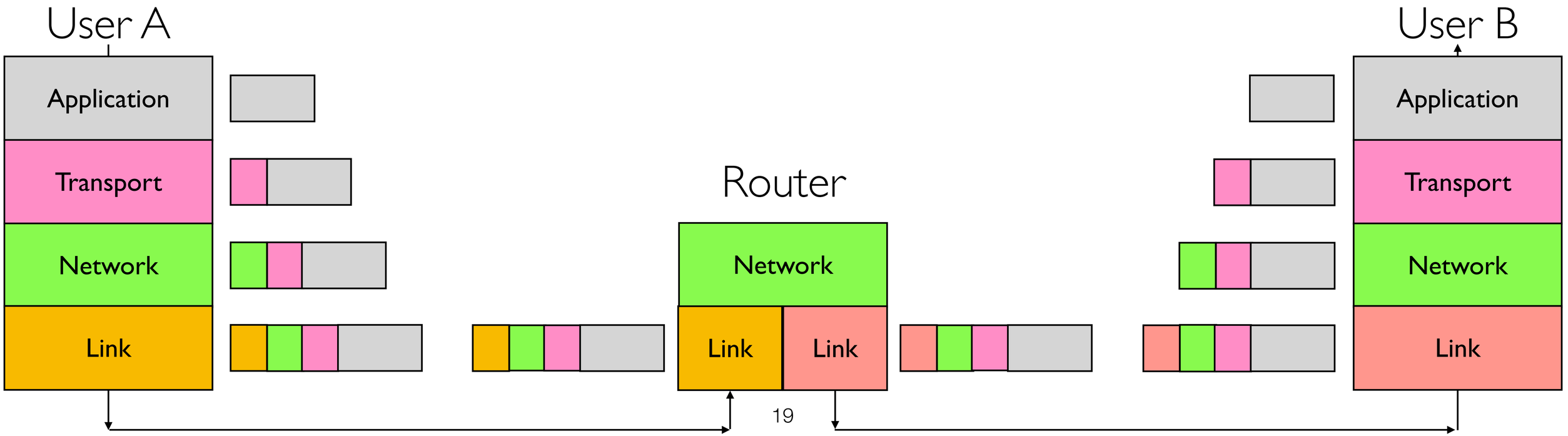
- **Concepts,** *Links, delays, switches*
- **Overall Architecture,** *Layers, protocols principles*
- **Network Layer,** *Best-effort global delivery of packets*



Taking Stock

Course so far:

- **Concepts,** *Links, delays, switches*
- **Overall Architecture,** *Layers, protocols principles*
- **Network Layer,** *Best-effort global delivery of packets*
- **Transport Layer,** *Reliable (or unreliable) delivery of data*

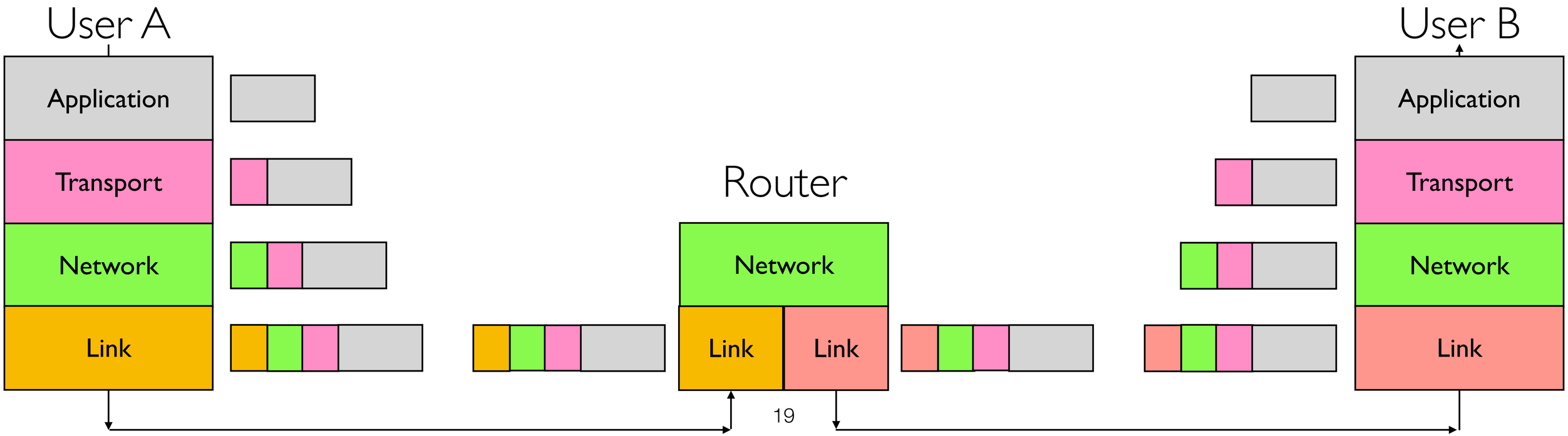


Taking Stock

Course so far:

- **Concepts,** *Links, delays, switches*
- **Overall Architecture,** *Layers, protocols principles*
- **Network Layer,** *Best-effort global delivery of packets*
- **Transport Layer,** *Reliable (or unreliable) delivery of data*

What's left?



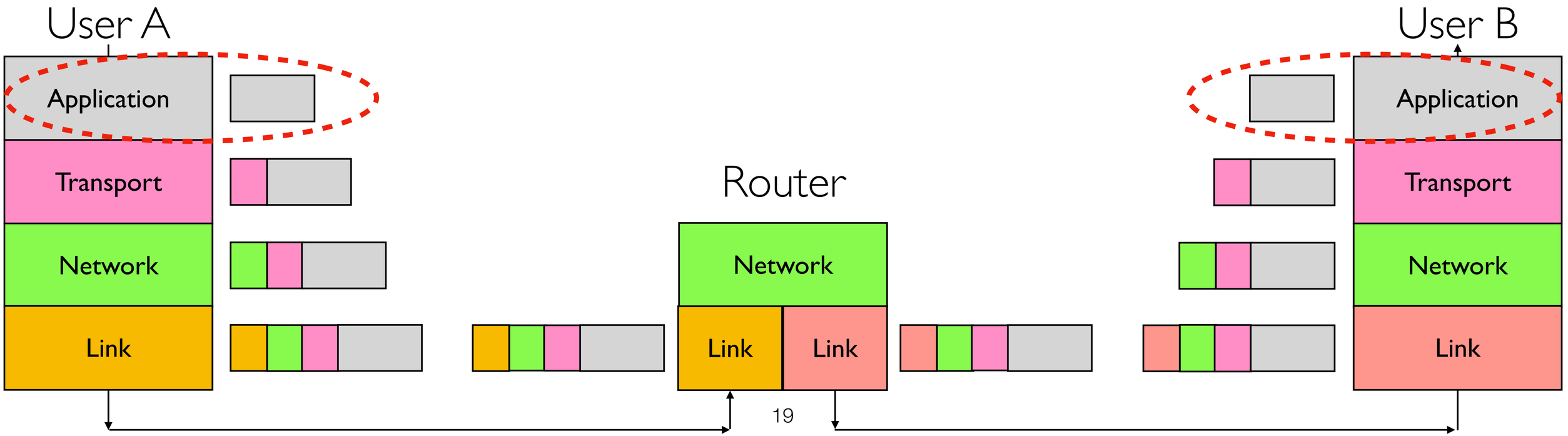
Taking Stock

Course so far:

- **Concepts,** *Links, delays, switches*
- **Overall Architecture,** *Layers, protocols principles*
- **Network Layer,** *Best-effort global delivery of packets*
- **Transport Layer,** *Reliable (or unreliable) delivery of data*

What's left?

- **Application Layer,** *DNS, HTTP (today)*



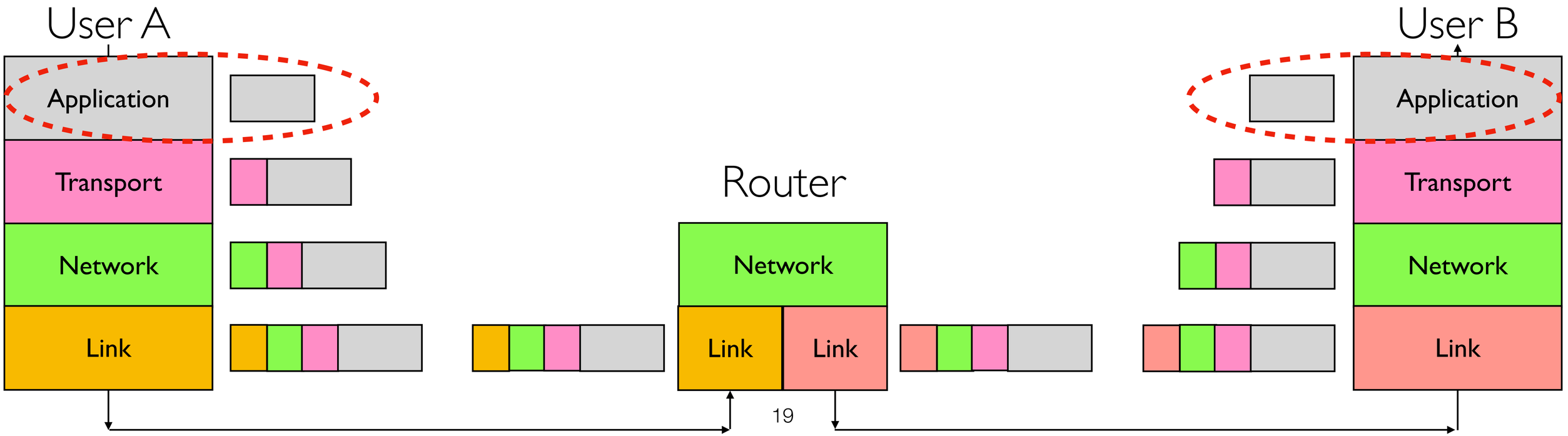
Taking Stock

Course so far:

- **Concepts,** *Links, delays, switches*
- **Overall Architecture,** *Layers, protocols principles*
- **Network Layer,** *Best-effort global delivery of packets*
- **Transport Layer,** *Reliable (or unreliable) delivery of data*

What's left?

- **Application Layer,** *DNS, HTTP (today)*
- **Lower Layers,** *Ethernet, Wireless*



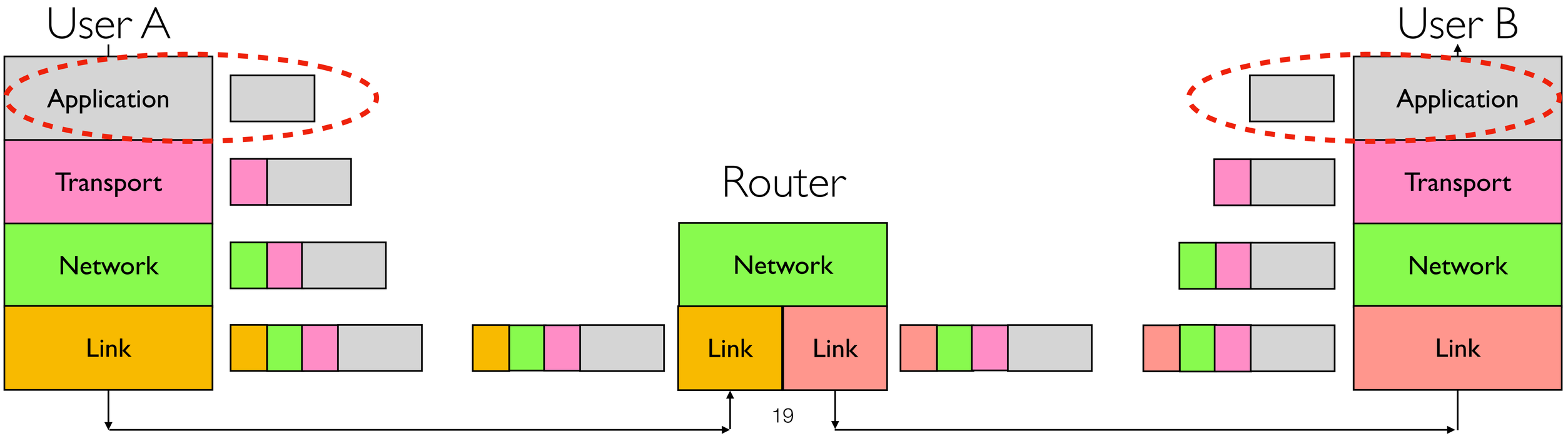
Taking Stock

Course so far:

- **Concepts,** *Links, delays, switches*
- **Overall Architecture,** *Layers, protocols principles*
- **Network Layer,** *Best-effort global delivery of packets*
- **Transport Layer,** *Reliable (or unreliable) delivery of data*

What's left?

- **Application Layer,** *DNS, HTTP (today)*
- **Lower Layers,** *Ethernet, Wireless*
- **Advanced Topics,** *Datacenters, SDN*



On to the Application Layer!

Application Layer

Application Layer

- **Domain Name System (DNS)**
  - What's behind (e.g.) zoo.cs.yale.edu?

Application Layer

- **Domain Name System (DNS)**
  - What's behind (e.g.) zoo.cs.yale.edu?

- **HTTP and the Web**
  - What happens when you click on a link?

Application Layer

- **Domain Name System (DNS)**
  - What's behind (e.g.) zoo.cs.yale.edu?
- **HTTP and the Web**
  - What happens when you click on a link?

Host Names & Addresses

Host Names & Addresses

- **Host address: e.g., *169.229.131.109***
  - A number used by protocols
  - Conforms to network structure (*the "where"*)

Host Names & Addresses

- **Host address: e.g., *169.229.131.109***
  - A number used by protocols
  - Conforms to network structure (*the "where"*)

- **Host names: e.g., *zoo.cs.yale.edu***
  - Mnemonic name usable by humans
  - Conforms to organizational structure (*the "who"*)

Host Names & Addresses

- **Host address: e.g., *169.229.131.109***
  - A number used by protocols
  - Conforms to network structure (*the "where"*)

- **Host names: e.g., *zoo.cs.yale.edu***
  - Mnemonic name usable by humans
  - Conforms to organizational structure (*the "who"*)

- **The Domain Name System (DNS) is how we map from one to the other**
  - A *directory service* for hosts on the Internet

Why Bother?

Why Bother?

- **Convenience**
  - Easier to remember **www.google.com** than 172.217.8.174

Why Bother?

- **Convenience**
  - Easier to remember **www.google.com** than 172.217.8.174

- **Provides a level of indirection!**
  - Decoupled names from addresses
  - Many uses beyond just naming a specific host

DNS: Early Days

DNS: Early Days

- **Mappings stored in a hosts.txt file (in /etc/hosts)**
  - Maintained by the Stanford Research Institute (SRI)
  - New versions periodically copied from SRI (via FTP)

DNS: Early Days

- **Mappings stored in a hosts.txt file (in /etc/hosts)**
  - Maintained by the Stanford Research Institute (SRI)
  - New versions periodically copied from SRI (via FTP)

- **As the Internet grew, this system broke down (obviously)**
  - SRI couldn't handle the load
  - Conflicts in selecting names
  - Hosts had inaccurate copies of hosts.txt

DNS: Early Days

- **Mappings stored in a hosts.txt file (in /etc/hosts)**
  - Maintained by the Stanford Research Institute (SRI)
  - New versions periodically copied from SRI (via FTP)

- **As the Internet grew, this system broke down (obviously)**
  - SRI couldn't handle the load
  - Conflicts in selecting names
  - Hosts had inaccurate copies of hosts.txt

- **The Domain Name Service (DNS) was invented to fix this**

Goals?

Goals?

- **Scalable**
  - Many *names*
  - Many *updates*
  - Many *users* creating names
  - Many *users* looking up names

Goals?

- **Scalable**
  - Many *names*
  - Many *updates*
  - Many *users* creating names
  - Many *users* looking up names
- **Highly available**

Goals?

- **Scalable**
  - Many *names*
  - Many *updates*
  - Many *users* creating names
  - Many *users* looking up names
- **Highly available**
- **Correct**
  - No naming conflicts (uniqueness)
  - Consistency → observe the latest update

Goals?

- **Scalable**
  - Many *names*
  - Many *updates*
  - Many *users* creating names
  - Many *users* looking up names
- **Highly available**
- **Correct**
  - No naming conflicts (uniqueness)
  - Consistency → observe the latest update
- **Lookups are fast**

How Do We Scale?

How Do We Scale?

- Partition the namespace

How Do We Scale?

- **Partition the namespace**

- **Distribute the administration of each partition**
  - Autonomy to update my partition's machines' names
  - Don't have to track other partition's updates

How Do We Scale?

- **Partition the namespace**

- **Distribute the administration of each partition**
  - Autonomy to update my partition's machines' names
  - Don't have to track other partition's updates

- **Distribute name resolution for each partition**
  - If I need to figure out the address for a name, I ask the partition that it belongs to

How Do We Scale?

- **Partition the namespace**

- **Distribute the administration of each partition**
  - Autonomy to update my partition's machines' names
  - Don't have to track other partition's updates

- **Distribute name resolution for each partition**
  - If I need to figure out the address for a name, I ask the partition that it belongs to

- ***How should we partition things?***

Key Idea: Hierarchical Distribution

Key Idea: Hierarchical Distribution

Three intertwined hierarchies

Key Idea: Hierarchical Distribution

Three intertwined hierarchies

- Hierarchical naming
  - As opposed to flat namespace

Key Idea: Hierarchical Distribution

Three intertwined hierarchies

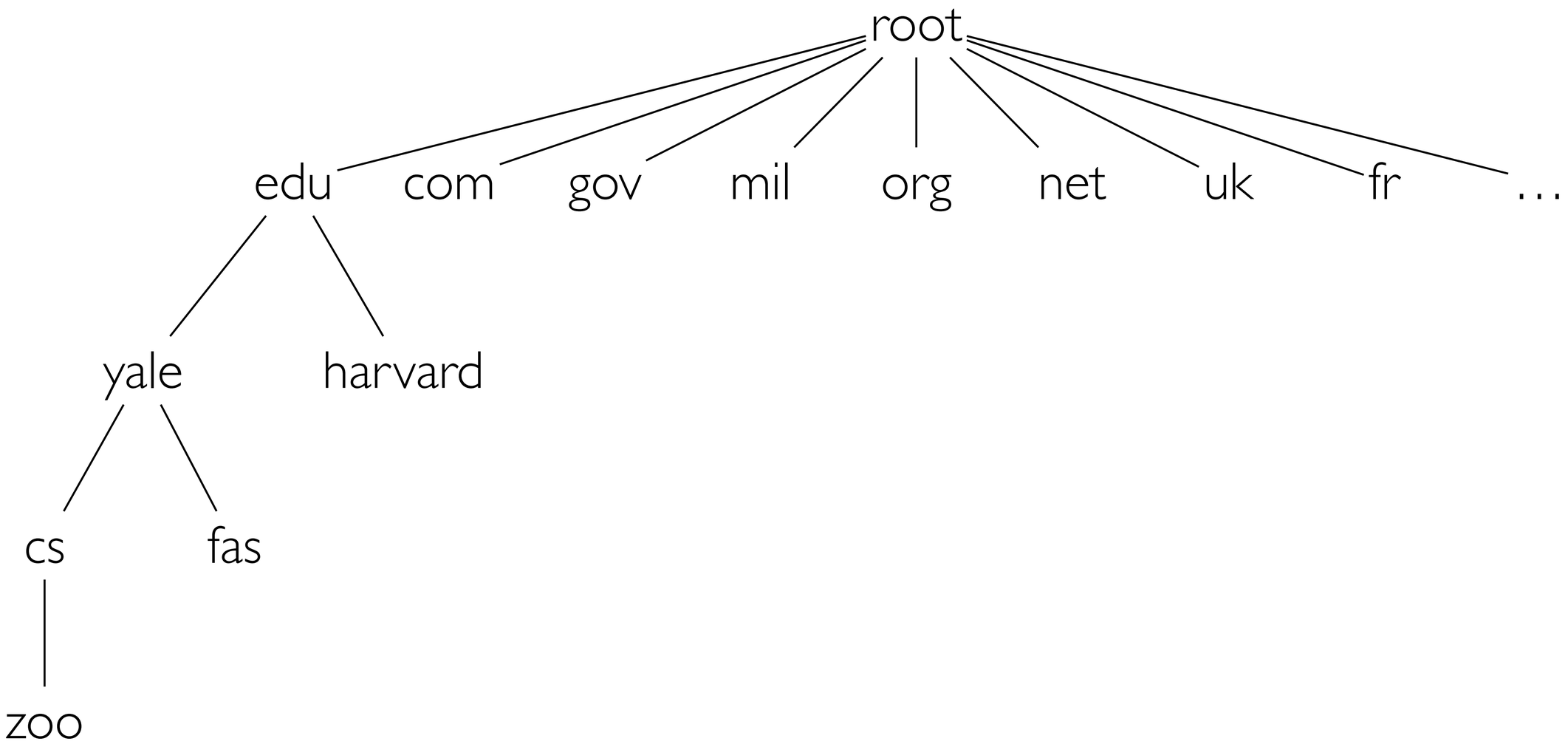
- Hierarchical naming
  - As opposed to flat namespace
- Hierarchical administration
  - As opposed to centralized administration

Key Idea: Hierarchical Distribution

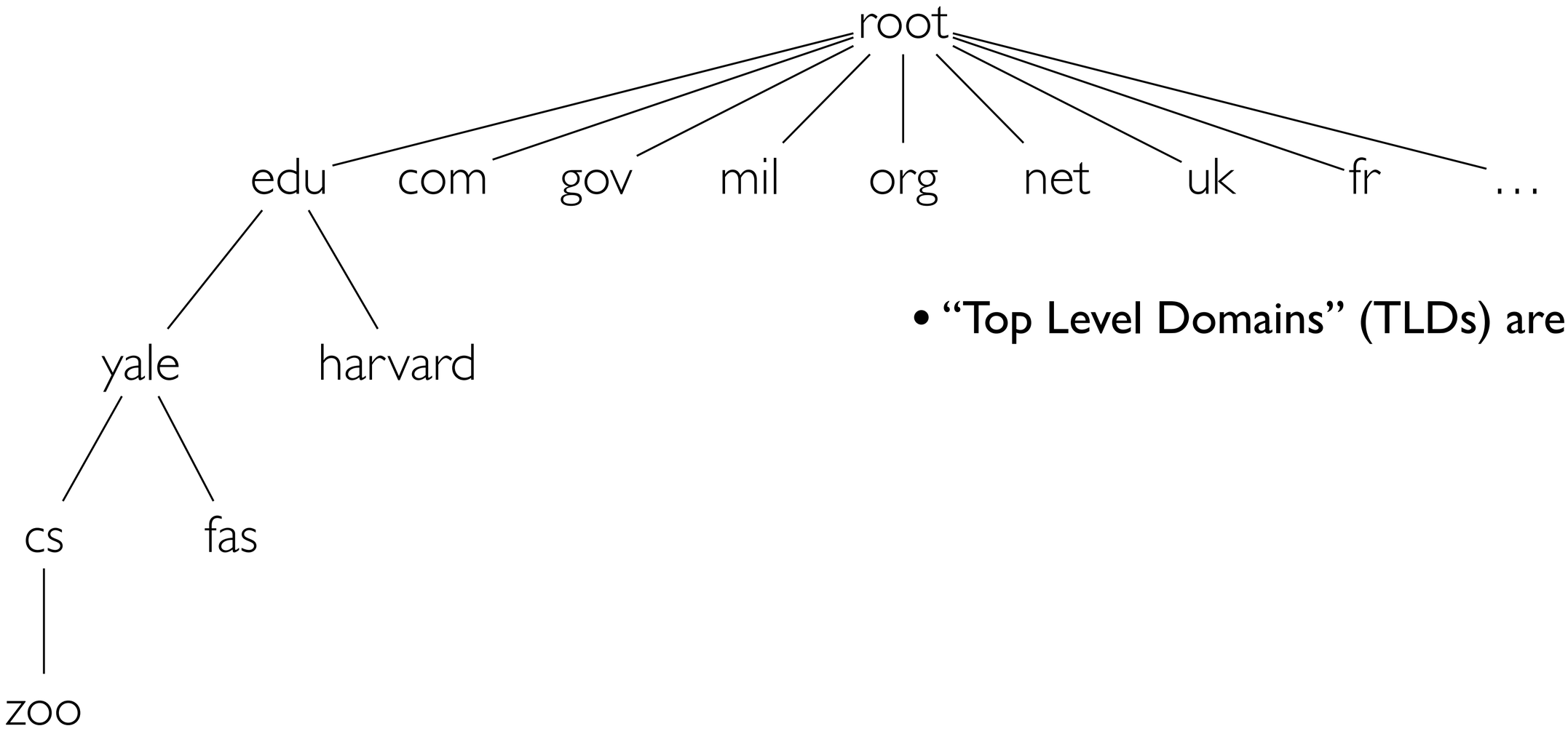
Three intertwined hierarchies

- **Hierarchical naming**
  - As opposed to flat namespace
- **Hierarchical administration**
  - As opposed to centralized administration
- **Hierarchical storage**
  - As opposed to centralized storage

Hierarchical Naming

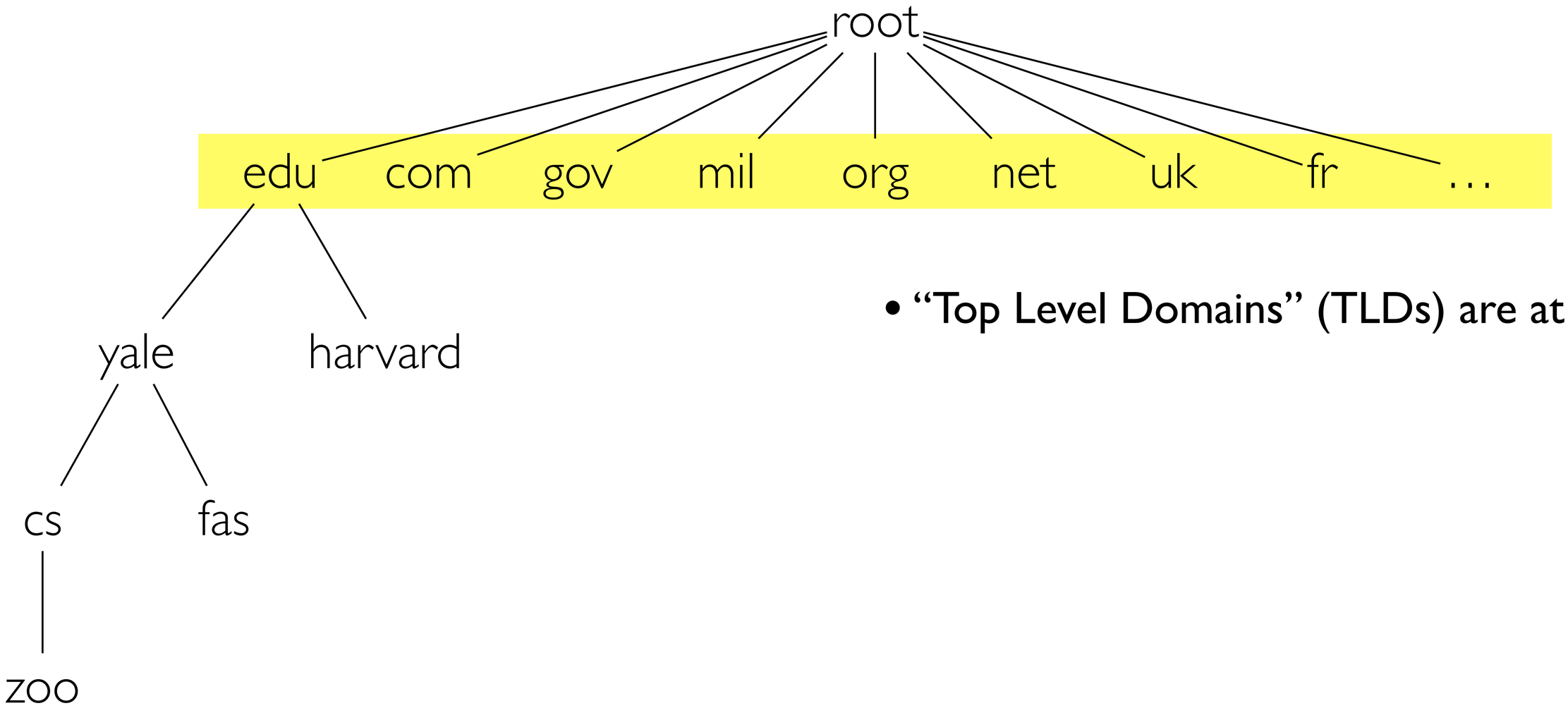


Hierarchical Naming



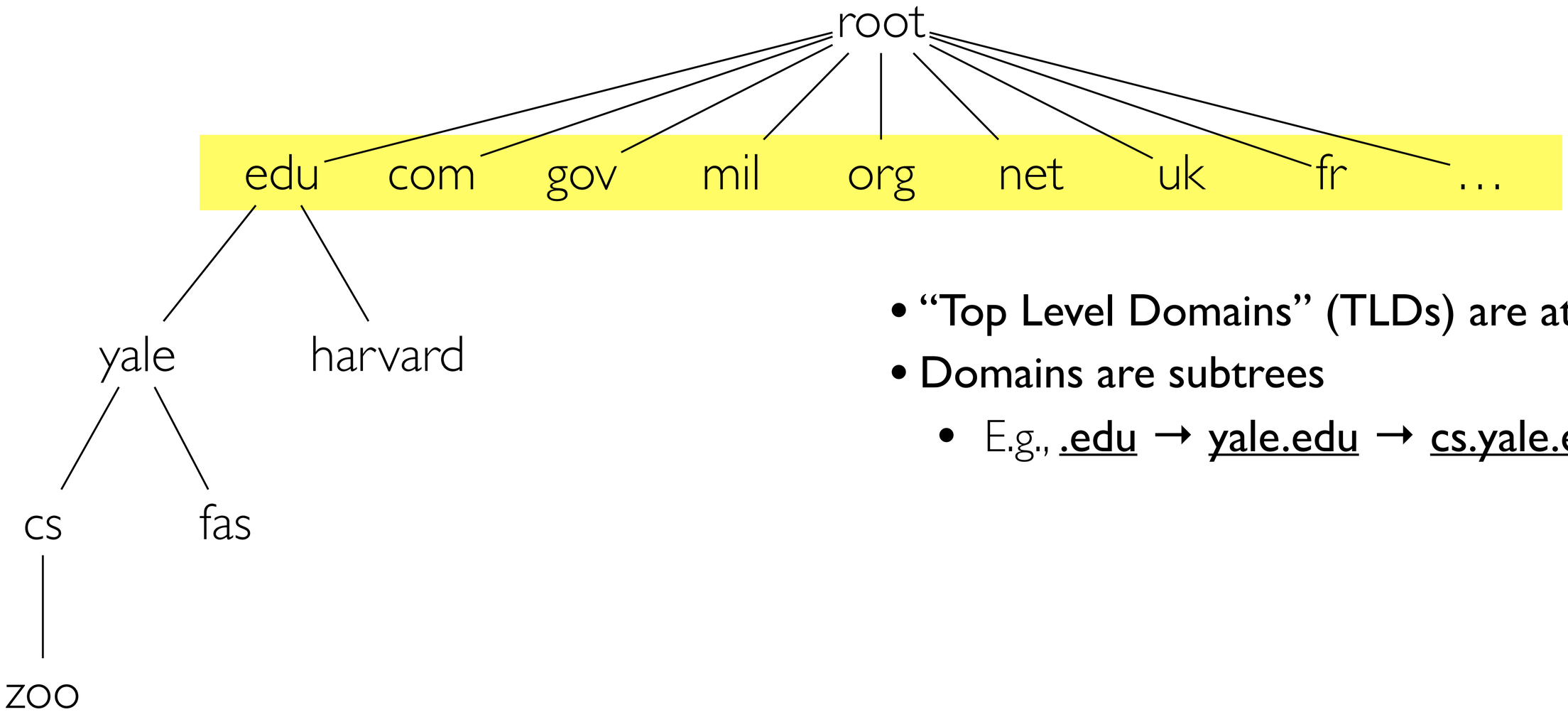
- "Top Level Domains" (TLDs) are at the top

Hierarchical Naming



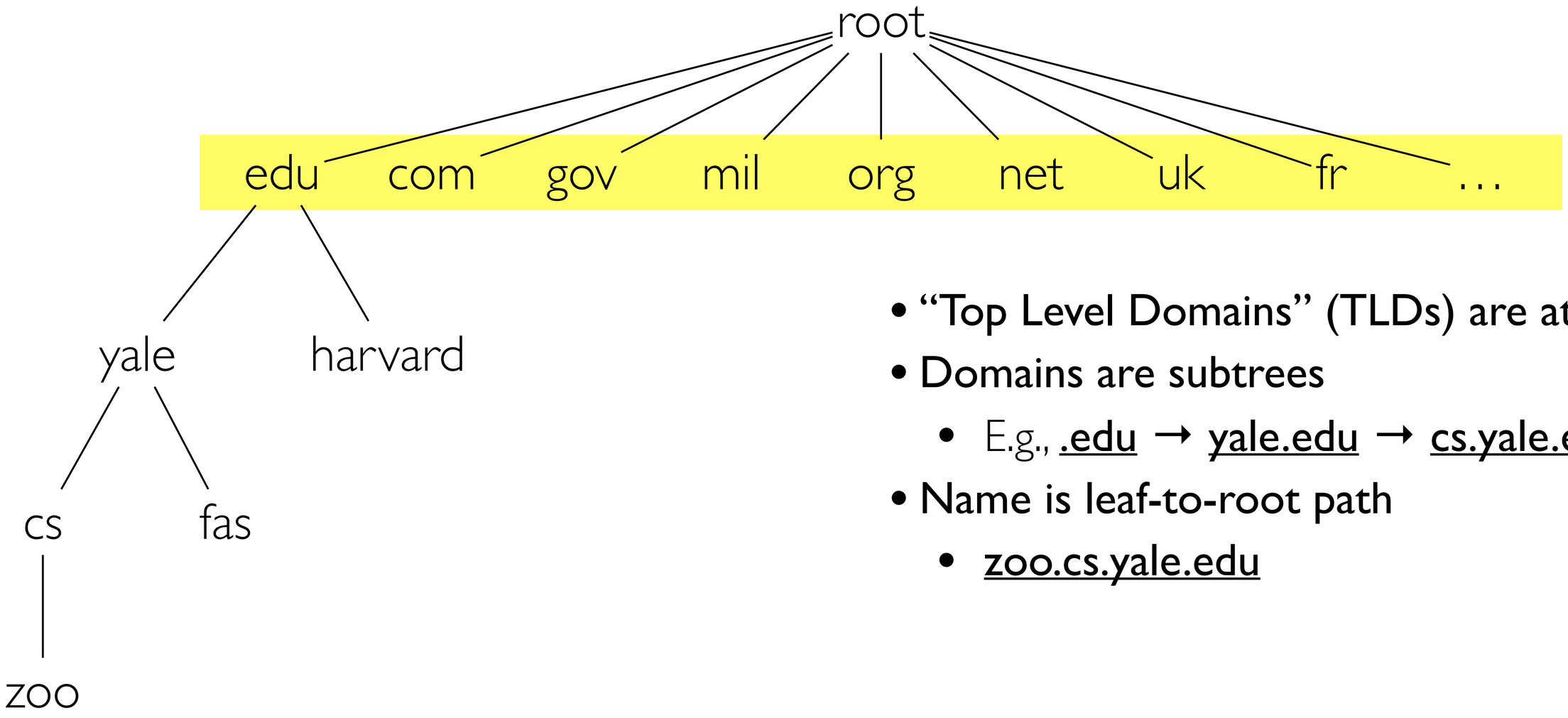
- "Top Level Domains" (TLDs) are at the top

Hierarchical Naming



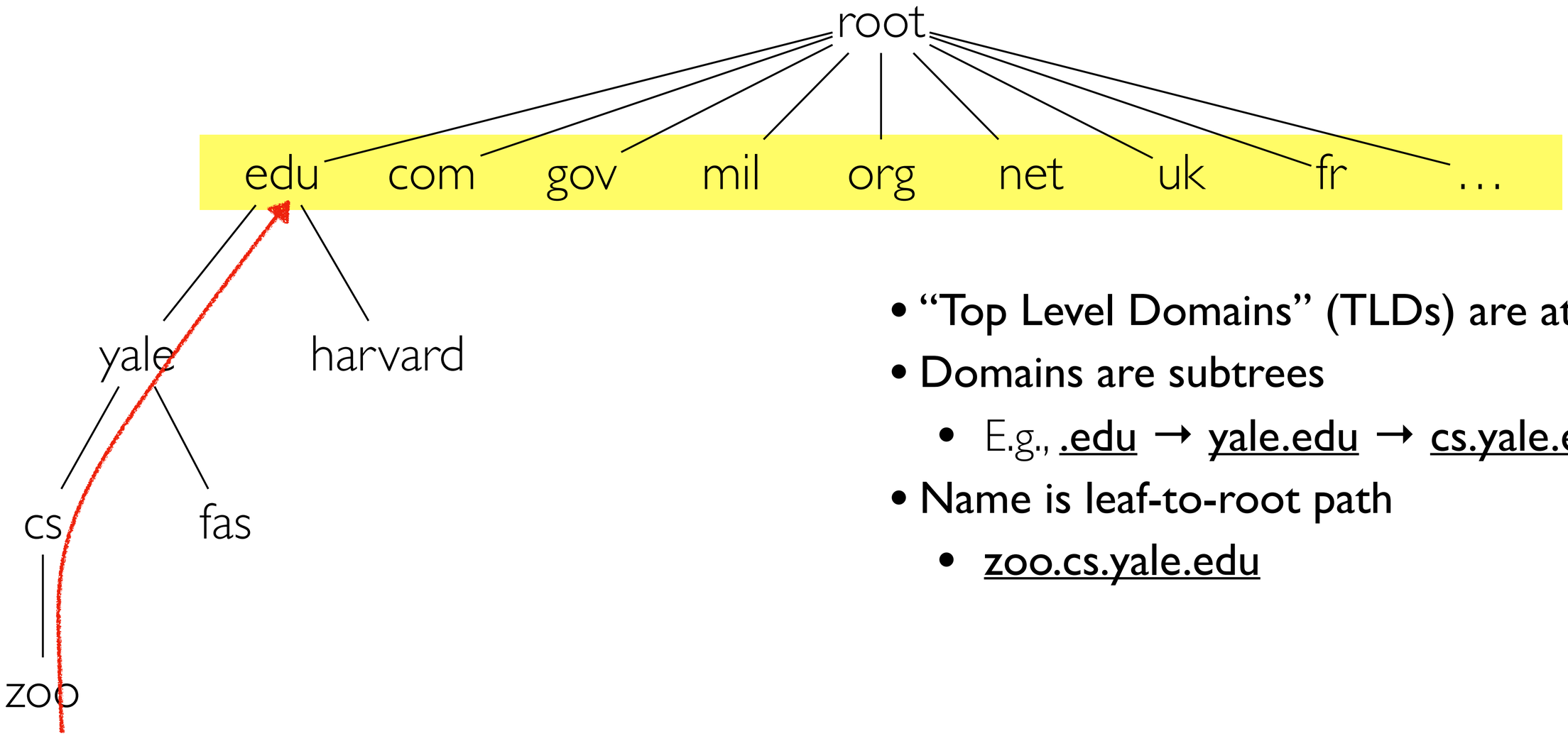
- "Top Level Domains" (TLDs) are at the top
- Domains are subtrees
  - E.g., .edu → yale.edu → cs.yale.edu →…

Hierarchical Naming



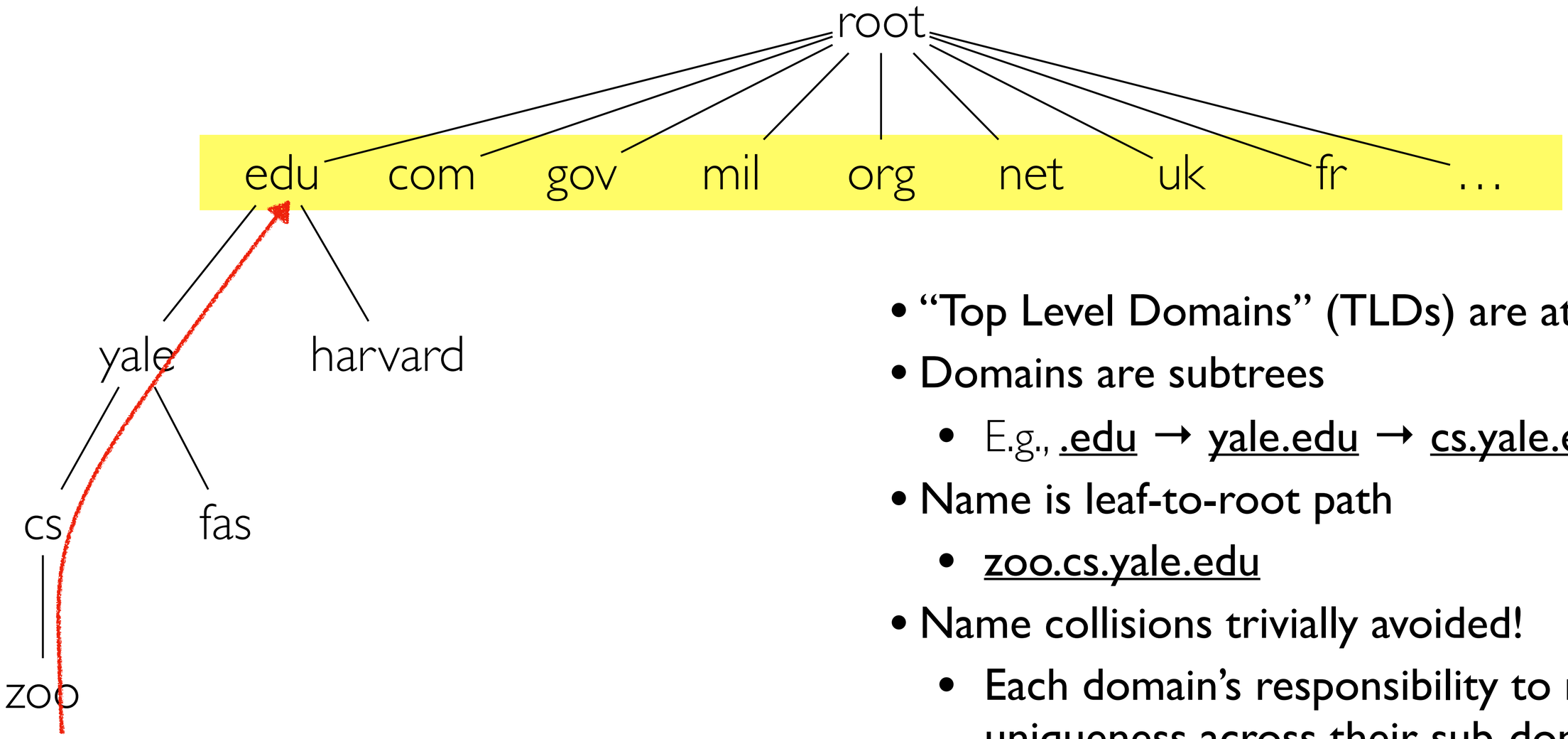
- "Top Level Domains" (TLDs) are at the top
- Domains are subtrees
  - E.g., .edu → yale.edu → cs.yale.edu →…
- Name is leaf-to-root path
  - zoo.cs.yale.edu

Hierarchical Naming



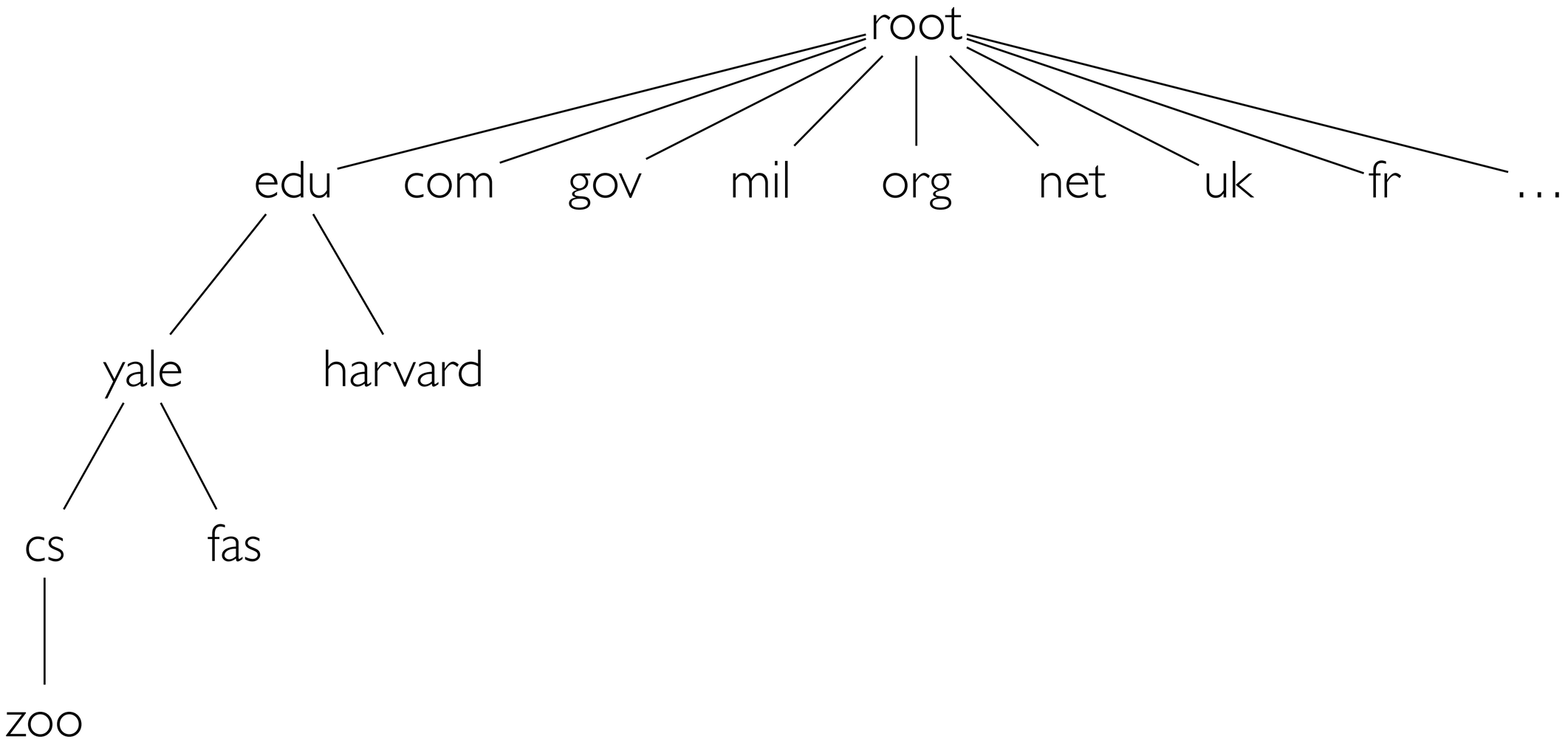
- "Top Level Domains" (TLDs) are at the top
- Domains are subtrees
  - E.g., .edu → yale.edu → cs.yale.edu →…
- Name is leaf-to-root path
  - zoo.cs.yale.edu

Hierarchical Naming

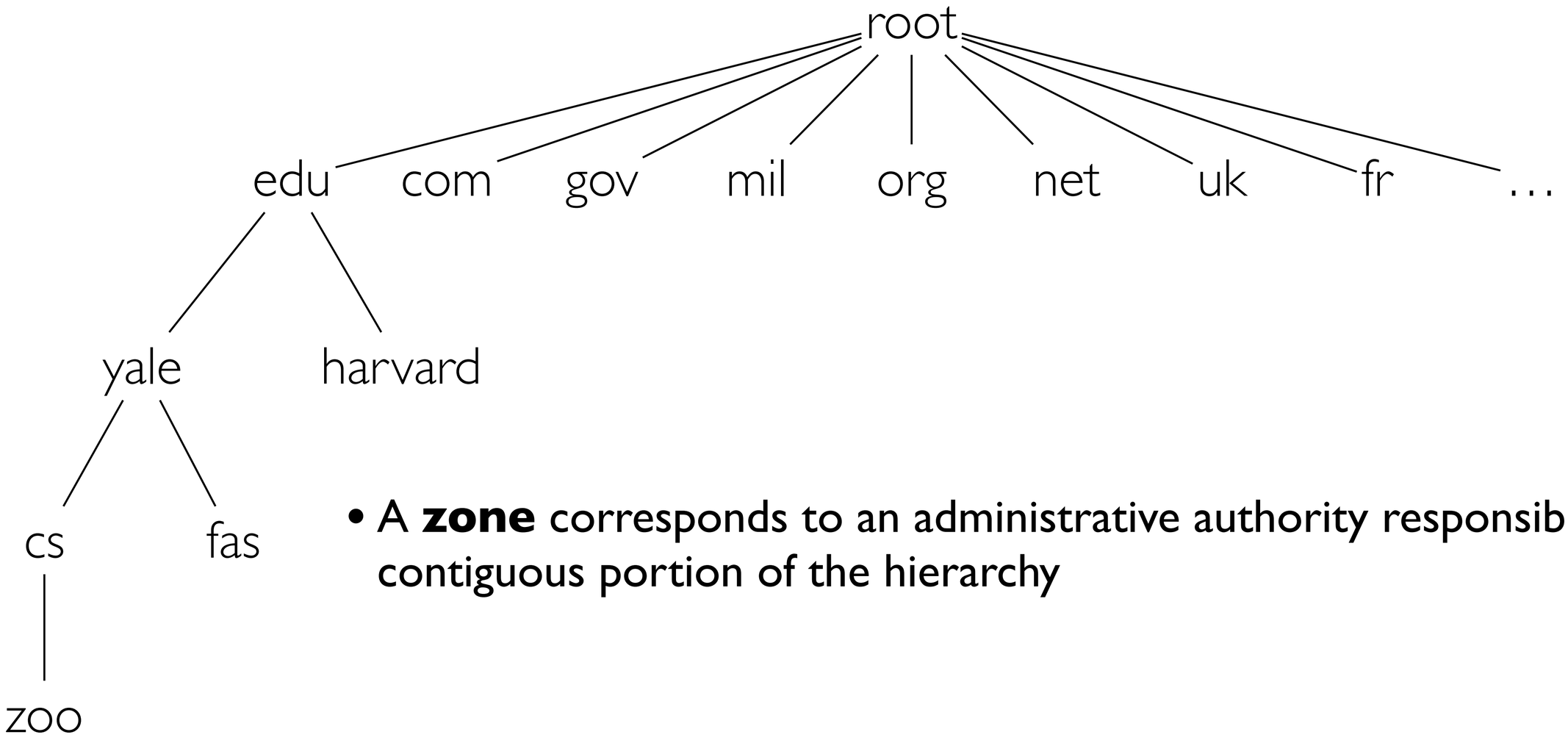


- "Top Level Domains" (TLDs) are at the top
- Domains are subtrees
  - E.g., .edu → yale.edu → cs.yale.edu → …
- Name is leaf-to-root path
  - zoo.cs.yale.edu
- Name collisions trivially avoided!
  - Each domain's responsibility to maintain uniqueness across their sub-domains

Hierarchical Administration

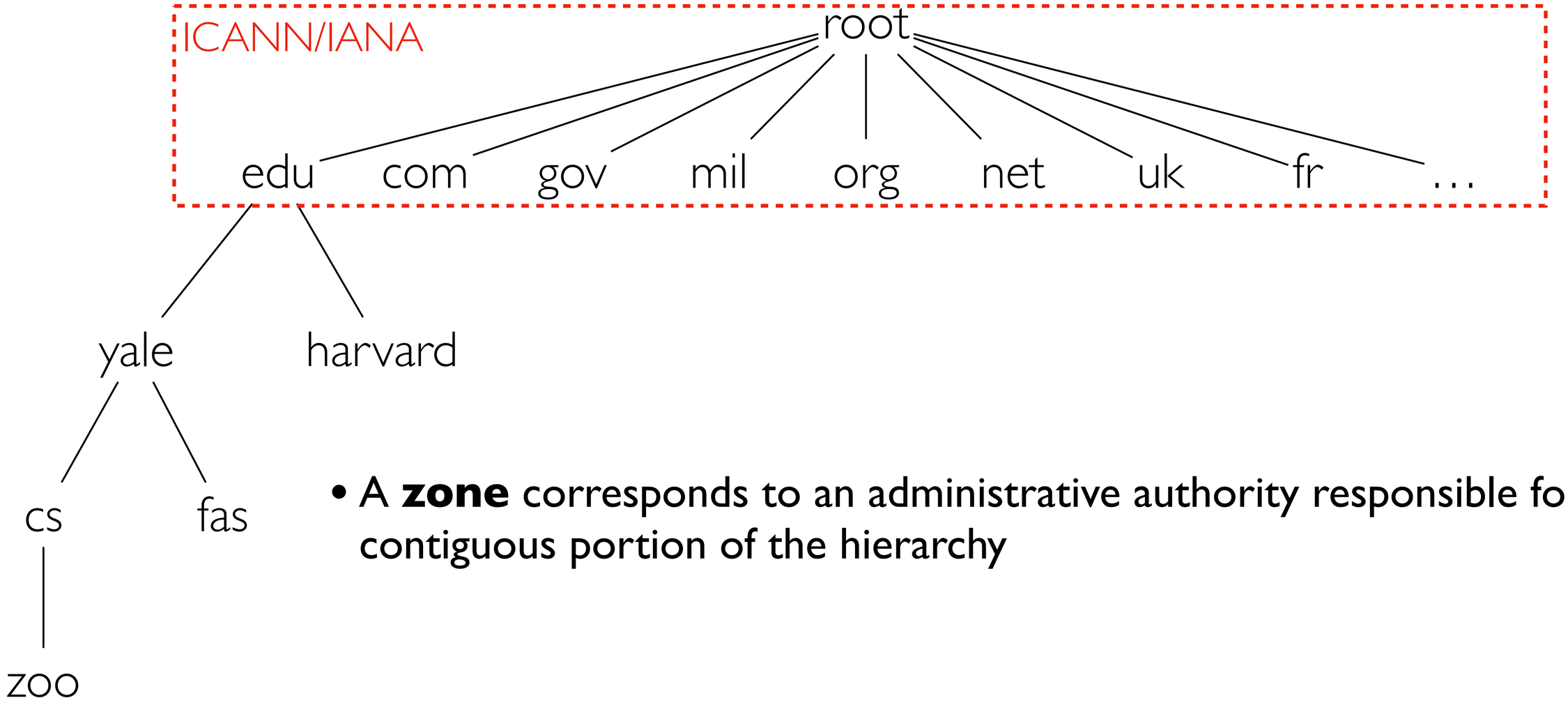


Hierarchical Administration



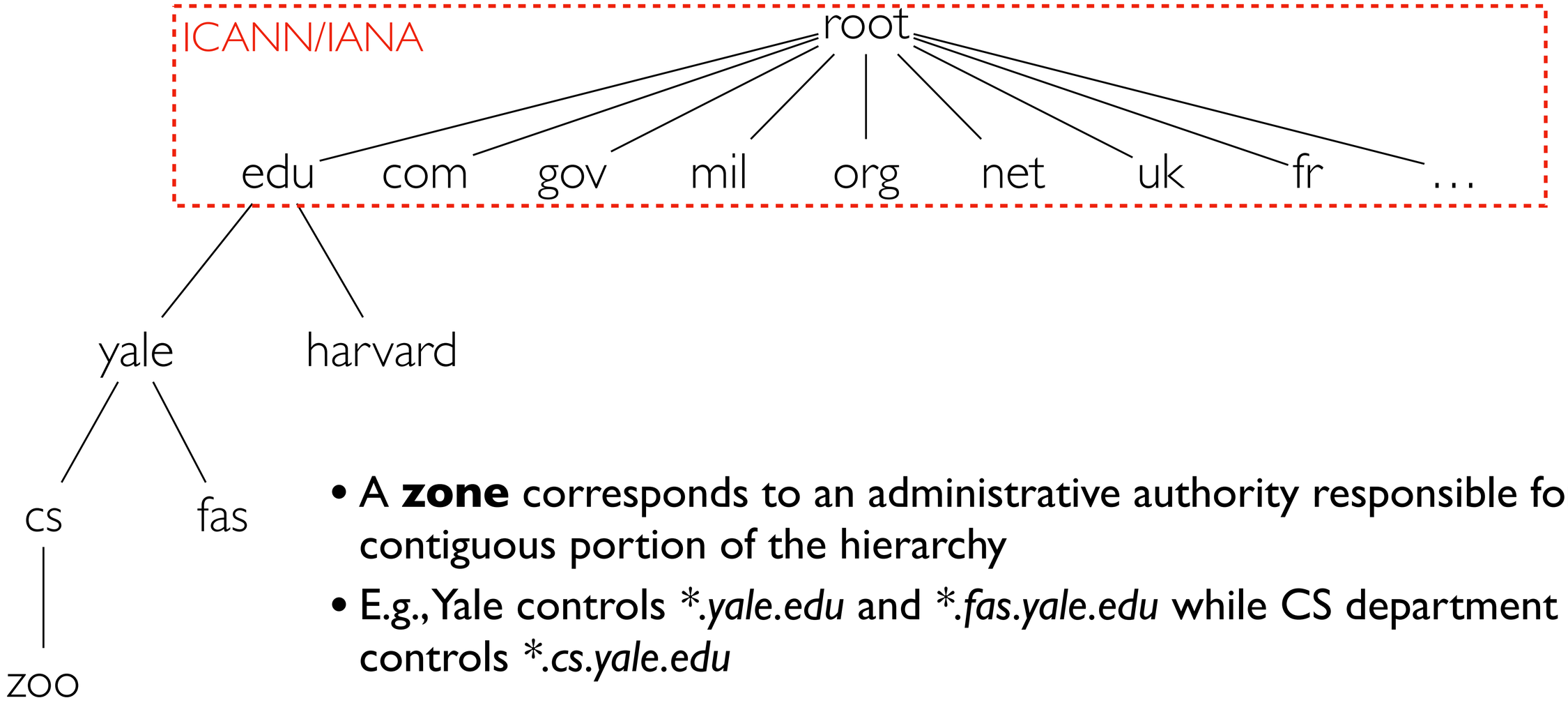
- A **zone** corresponds to an administrative authority responsible for a contiguous portion of the hierarchy

Hierarchical Administration



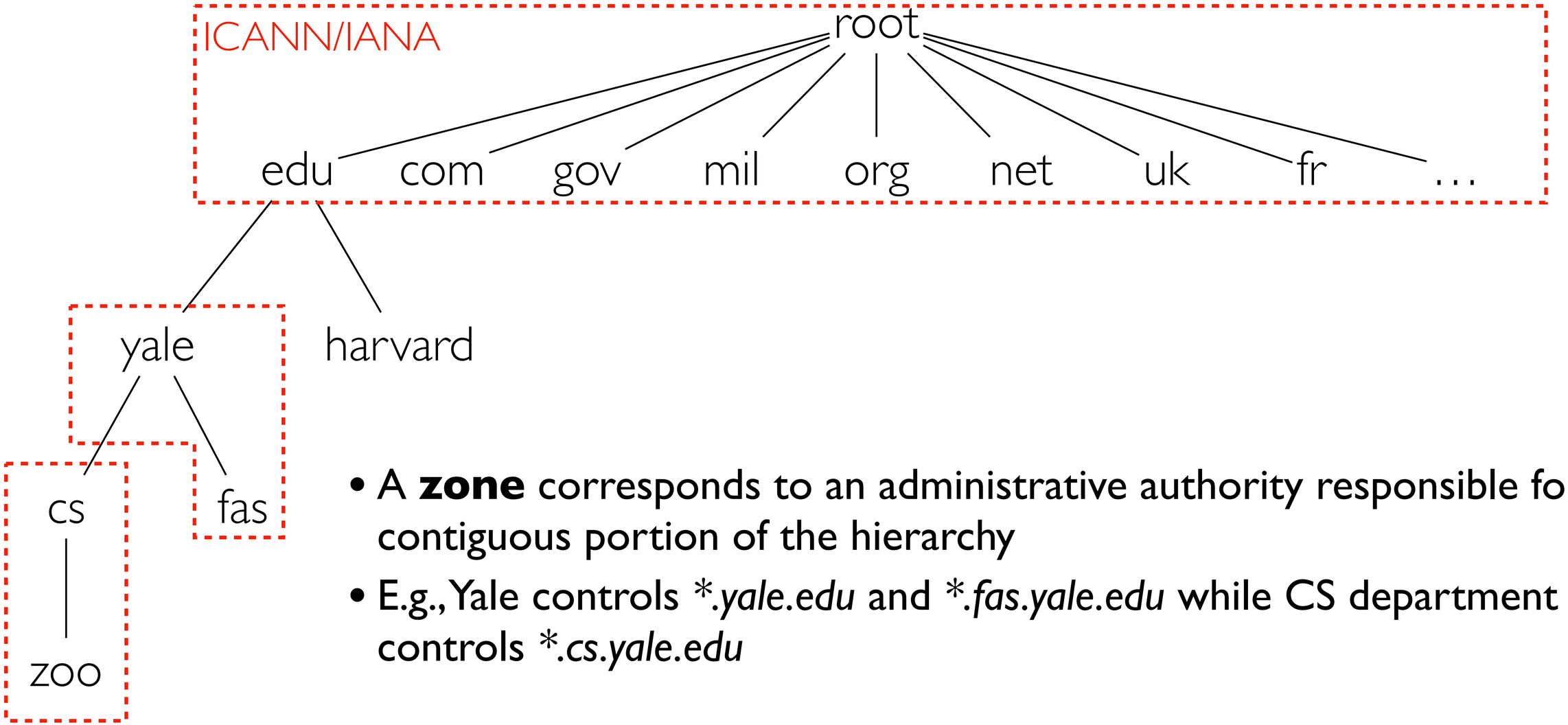
- A **zone** corresponds to an administrative authority responsible for a contiguous portion of the hierarchy

Hierarchical Administration



- A **zone** corresponds to an administrative authority responsible for a contiguous portion of the hierarchy
- E.g., Yale controls **.yale.edu* and **.fas.yale.edu* while CS department controls **.cs.yale.edu*

Hierarchical Administration



- A **zone** corresponds to an administrative authority responsible for a contiguous portion of the hierarchy
- E.g., Yale controls **.yale.edu* and **.fas.yale.edu* while CS department controls **.cs.yale.edu*

Storage Hierarchy

Storage Hierarchy

- **Top of hierarchy: Root servers**
  - Location hardwired into relevant servers
  - Store addresses **all** next level DNS servers…

Storage Hierarchy

- **Top of hierarchy: Root servers**
  - Location hardwired into relevant servers
  - Store addresses **all** next level DNS servers…
- **Next Level: Top-Level Domain (TLD) servers**
  - .com, .edu, etc.; managed professionally
  - Store addresses of next level DNS servers **under them…**

Storage Hierarchy

- **Top of hierarchy: Root servers**
  - Location hardwired into relevant servers
  - Store addresses **all** next level DNS servers…
- **Next Level: Top-Level Domain (TLD) servers**
  - .com, .edu, etc.; managed professionally
  - Store addresses of next level DNS servers **under them…**
- **Bottom Level: Authoritative DNS servers**
  - Store the **actual name-to-address mappings**
  - Maintained by the corresponding administrative authority

Storage Hierarchy

Storage Hierarchy

- Every server knows the address of the root name server

Storage Hierarchy

- Every server knows the address of the root name server
- Root servers know the address of all TLD servers

Storage Hierarchy

- Every server knows the address of the root name server
- Root servers know the address of all TLD servers
- …

Storage Hierarchy

- Every server knows the address of the root name server
- Root servers know the address of all TLD servers
- …
- An authoritative DNS server stores name-to-address mappings (“resource records”) for all DNS names in the domain that it has authority for

Storage Hierarchy

- Every server knows the address of the root name server
- Root servers know the address of all TLD servers
- …
- An authoritative DNS server stores name-to-address mappings (“resource records”) for all DNS names in the domain that it has authority for

- Each server stores a subset of the total DNS database

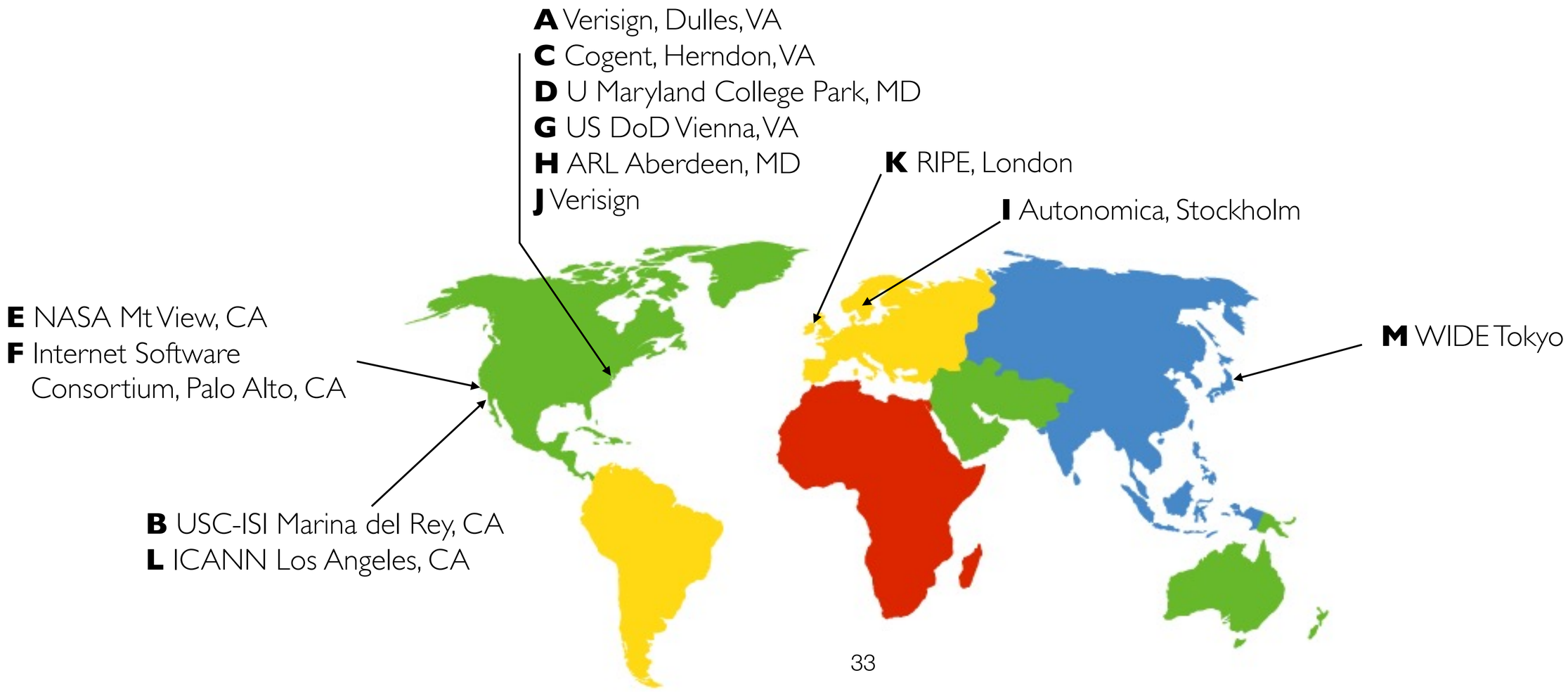
Storage Hierarchy

- Every server knows the address of the root name server
- Root servers know the address of all TLD servers
- …
- An authoritative DNS server stores name-to-address mappings (“resource records”) for all DNS names in the domain that it has authority for

- Each server stores a subset of the total DNS database
- Each server can discover the server(s) responsible for any portion of the hierarchy

DNS Root Servers

- 13 root servers (labeled A-M; see http://www.root-servers)



DNS Root Servers

- 13 root servers (labeled A-M; see http://www.root-servers)
- Replicated via any-casting

A Verisign, Dulles, VA
C Cogent, Herndon, VA
D U Maryland College Park, MD
G US DoD Vienna, VA
H ARL Aberdeen, MD
J Verisign (21 locations)

K RIPE, London (plus 16 other locations)

I Autonomica, Stockholm (plus 29 other locations)

E NASA Mt View, CA
F Internet Software Consortium, Palo Alto, CA (and 37 other locations)

M WIDE Tokyo (plus Seoul, Paris, San Fransisco)

B USC-ISI Marina del Rey, CA
L ICANN Los Angeles, CA

Anycast in a nutshell

Anycast in a nutshell

- Routing finds shortest paths to destination

Anycast in a nutshell

- Routing finds shortest paths to destination
- What happens if multiple machines advertise the same address?

Anycast in a nutshell

- Routing finds shortest paths to destination
- What happens if multiple machines advertise the same address?
- The network will deliver the packet to the closest machine with that address

Anycast in a nutshell

- Routing finds shortest paths to destination
- What happens if multiple machines advertise the same address?
- The network will deliver the packet to the closest machine with that address
- This is called "anycast"
  - Very robust
  - Requires no modification to routing algorithms

DNS Records

DNS Records

- DNS Servers store **resource records (RRs)**
  - RR is (name, value, type, TTL)

DNS Records

- DNS Servers store **resource records (RRs)**
  - RR is (name, value, type, TTL)

- Type = A: (→*Address*)
  - Name = hostname
  - Value = IP address

DNS Records

- DNS Servers store **resource records (RRs)**
  - RR is (name, value, type, TTL)

- Type = A: (→*Address*)
  - Name = hostname
  - Value = IP address
- Type = NS: (→*Name Server*)
  - Name = domain
  - Value = name of DNS server for domain

DNS Records

- DNS Servers store **resource records (RRs)**
  - RR is (name, value, type, TTL)

- Type = A: (→*Address*)
  - Name = hostname
  - Value = IP address
- Type = NS: (→*Name Server*)
  - Name = domain
  - Value = name of DNS server for domain
- Type = MX: (→*Mail eXchanger*)
  - Name = domain in email address
  - Value = name(s) of mail server(s)

Inserting Resource Records into DNS

Inserting Resource Records into DNS

- Example: you just created company “FooBar”

Inserting Resource Records into DNS

- Example: you just created company “FooBar”
- You get a block of IP addresses from your ISP
  - Say 212.44.9.128/25

Inserting Resource Records into DNS

- **Example: you just created company "FooBar"**
- **You get a block of IP addresses from your ISP**
  - Say 212.44.9.128/25
- **Register foobar.com at registrar (e.g., GoDaddy)**
  - Provide registrar with names and IP addresses of your authoritative name server(s)
  - Registrar inserts RR pairs into the .com TLD server
    - (foobar.com, dns1.foobar.com, NS)
    - (dns1.foobar.com, 212.44.9.129, A)

Inserting Resource Records into DNS

- **Example: you just created company "FooBar"**
- **You get a block of IP addresses from your ISP**
  - Say 212.44.9.128/25
- **Register foobar.com at registrar (e.g., GoDaddy)**
  - Provide registrar with names and IP addresses of your authoritative name server(s)
  - Registrar inserts RR pairs into the .com TLD server
    - (foobar.com, dns1.foobar.com, NS)
    - (dns1.foobar.com, 212.44.9.129, A)
- **Store resource records in your server dns1.foobar.com**
  - e.g., type A records: (foobar.com, 212.44.9.130, A), (social.foobar.com, 212.44.9.131, A), etc.
  - e.g., type MX records for foobar.com

Using DNS (Client/Application view)

Using DNS (Client/Application view)

- **Two components**
  - Local DNS server
  - Resolver software on hosts

Using DNS (Client/Application view)

- **Two components**
  - Local DNS server
  - Resolver software on hosts

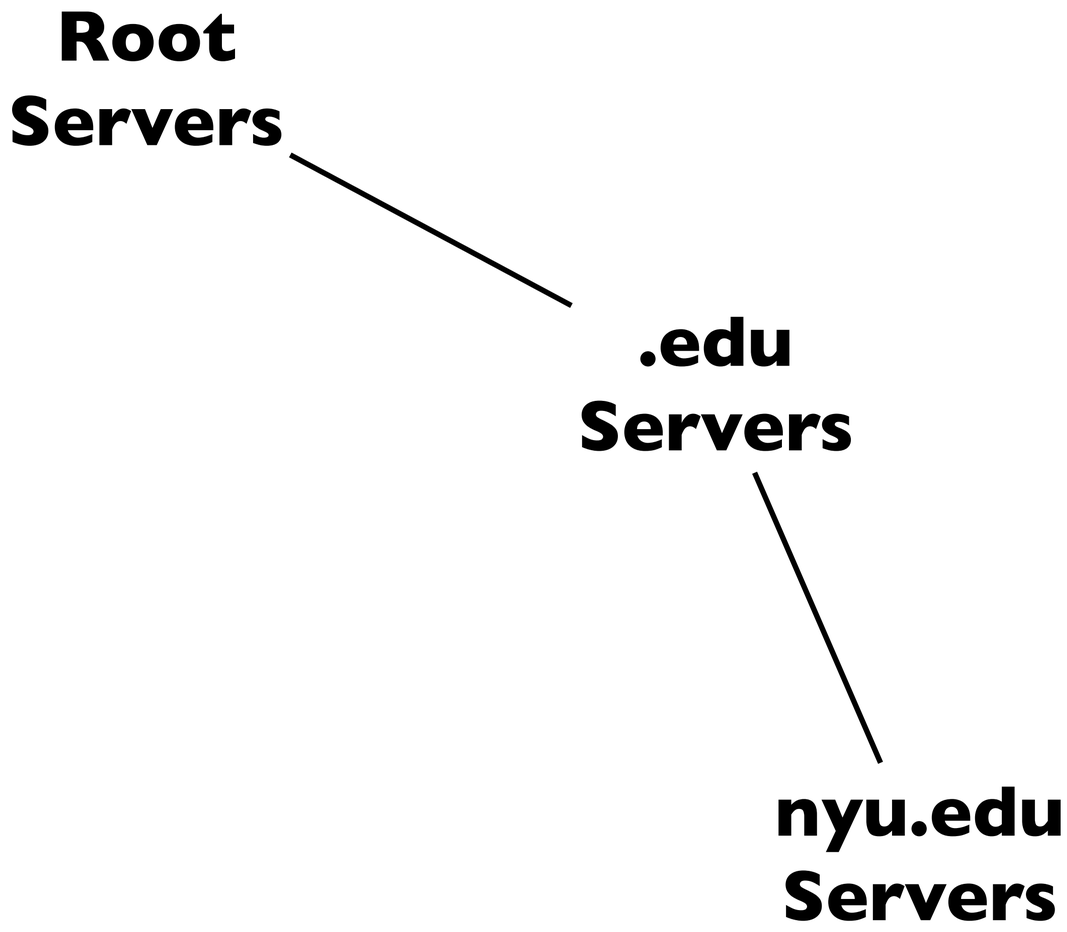
- **Local DNS server ("default name server")**
  - Clients configured with the default server's address or learn it via a host configuration protocol (e.g., DHCP — future lecture)

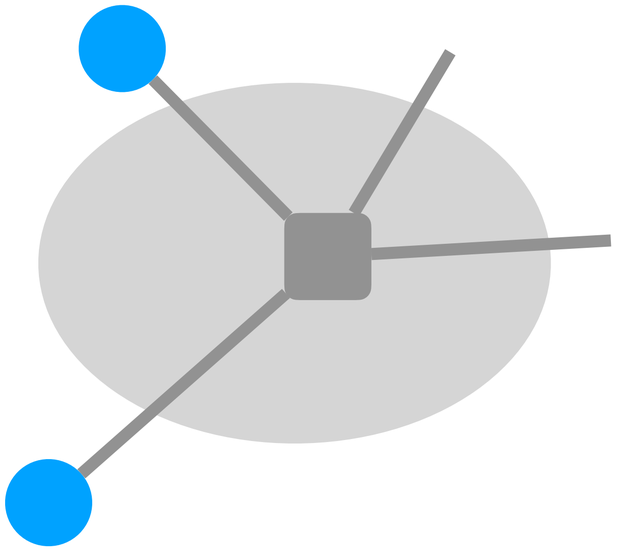
Using DNS (Client/Application view)

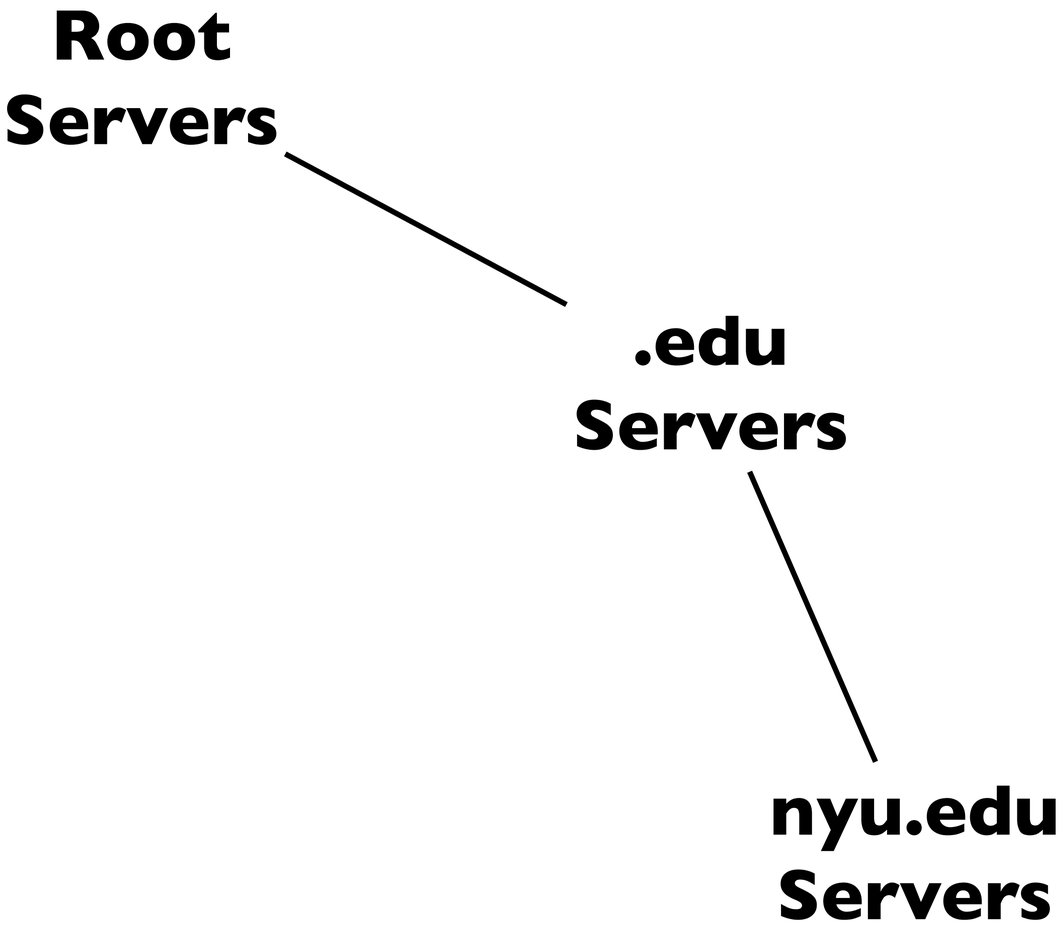
- **Two components**
  - Local DNS server
  - Resolver software on hosts

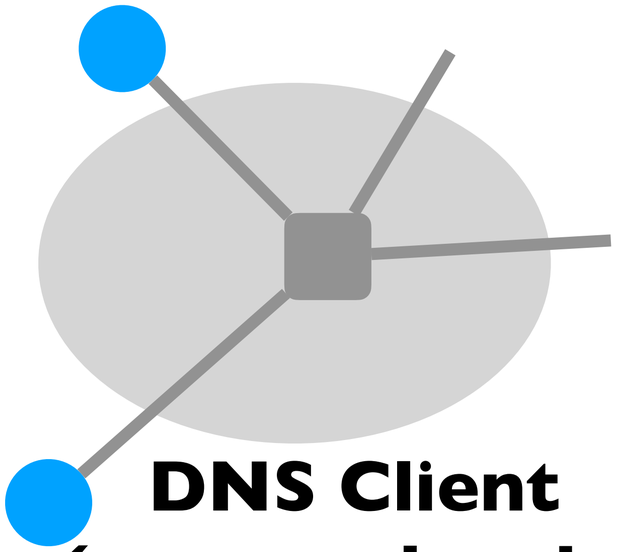
- **Local DNS server ("default name server")**
  - Clients configured with the default server's address or learn it via a host configuration protocol (e.g., DHCP — future lecture)

- **Client application:**
  - Obtain DNS name (e.g., from URL)
  - Triggers DNS request to its local DNS server


Root
Servers
.edu
Servers
nyu.edu
Servers




Root
Servers
.edu
Servers
nyu.edu
Servers



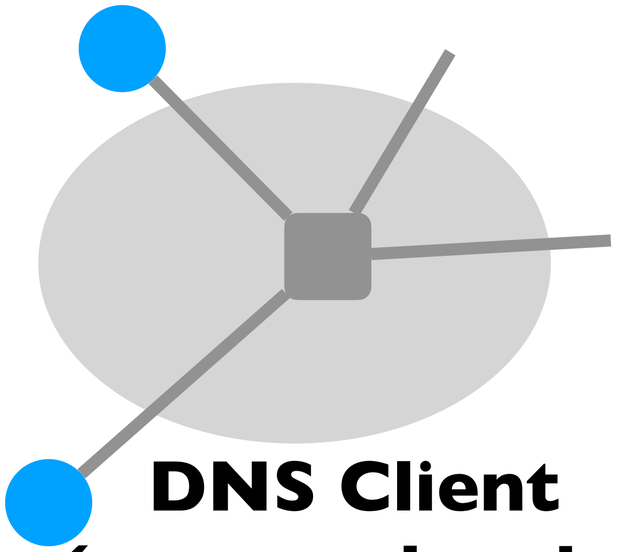
DNS Client
(me.cs.yale.edu)

Root Servers

.edu Servers

nyu.edu Servers

Local DNS server
(mydns.yale.edu)



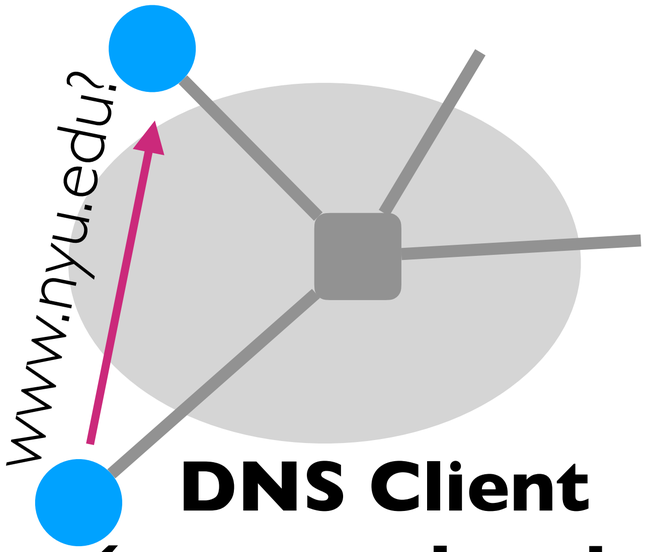
DNS Client
(me.cs.yale.edu)

Root
Servers

.edu
Servers

nyu.edu
Servers

Local DNS server (mydns.yale.edu)

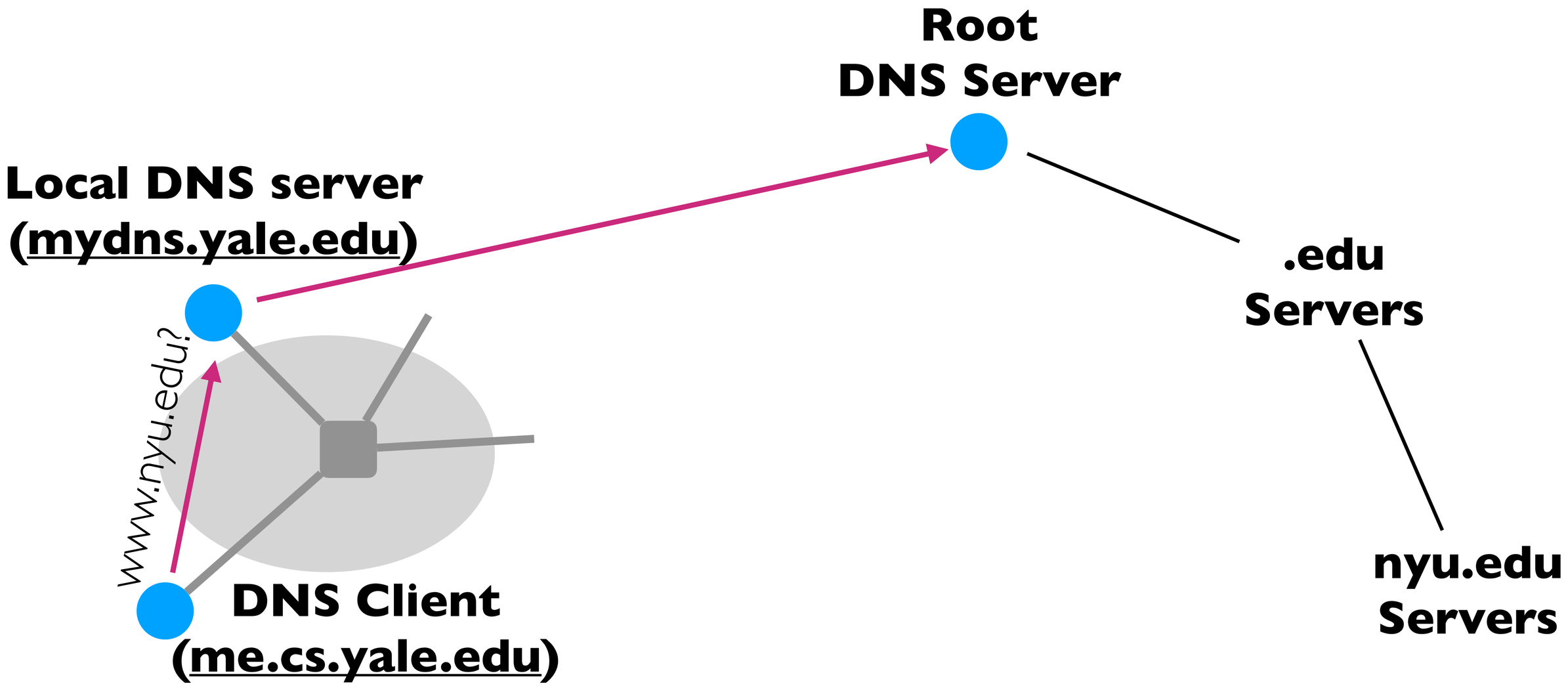


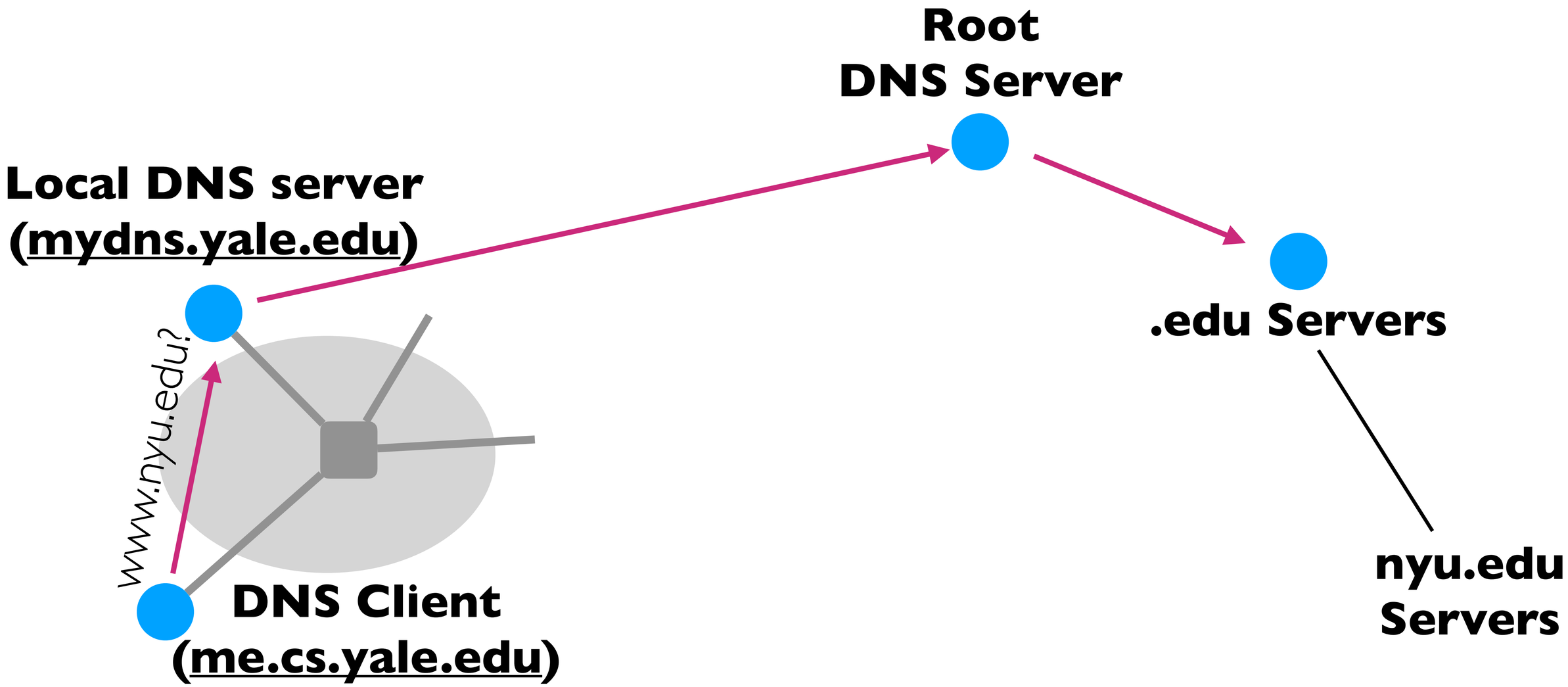
DNS Client (me.cs.yale.edu)

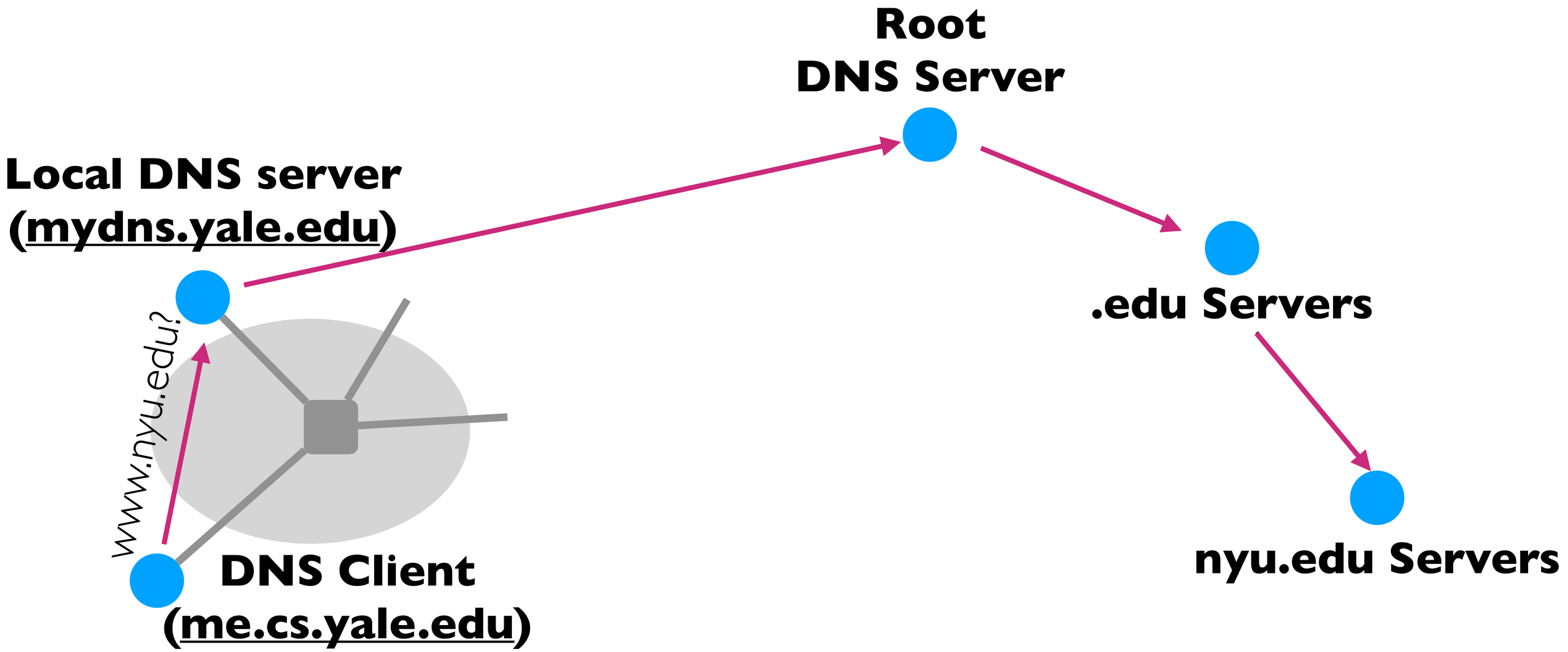
Root Servers

.edu Servers

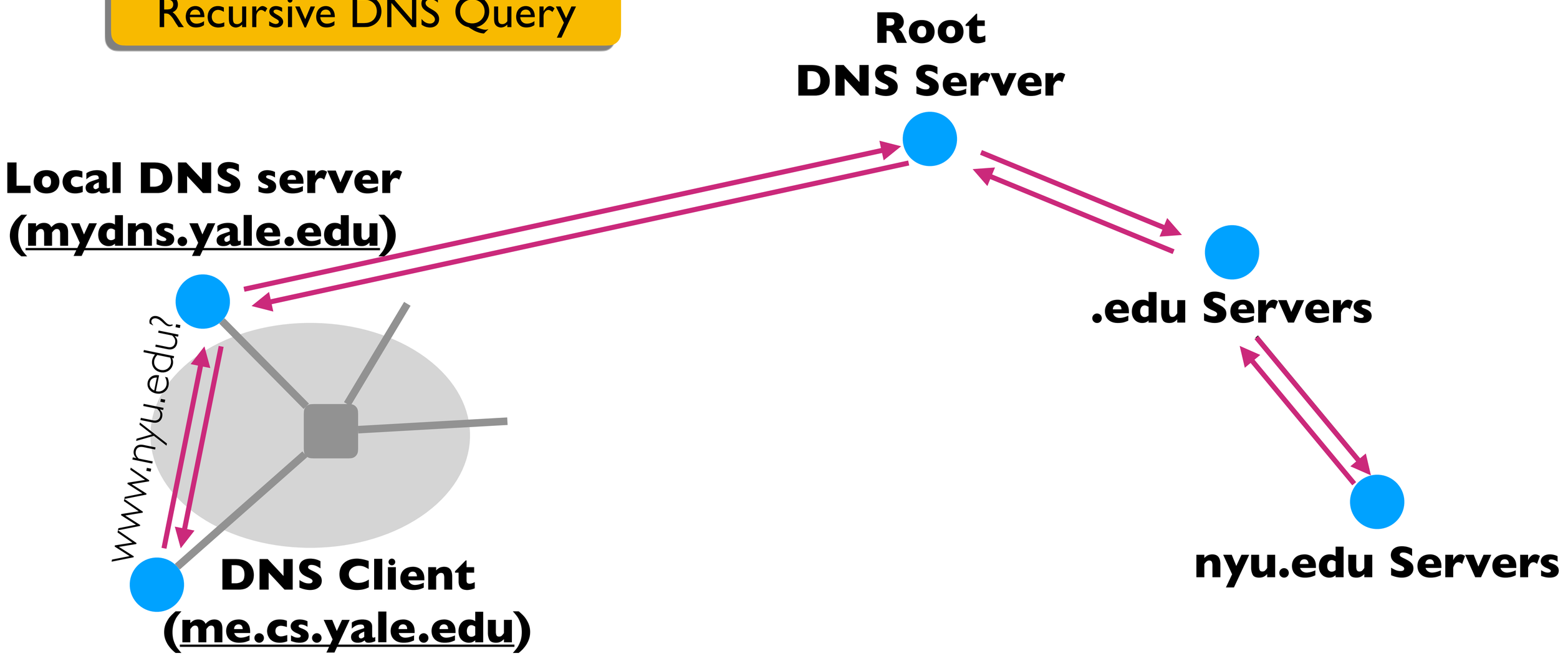
nyu.edu Servers


Root
DNS Server
Local DNS server
(mydns.yale.edu)
.edu
Servers
www.nyu.edu?
DNS Client
(me.cs.yale.edu)
nyu.edu
Servers


Root
DNS Server
Local DNS server
(mydns.yale.edu)
.edu Servers
www.nyu.edu?
DNS Client
(me.cs.yale.edu)
nyu.edu
Servers

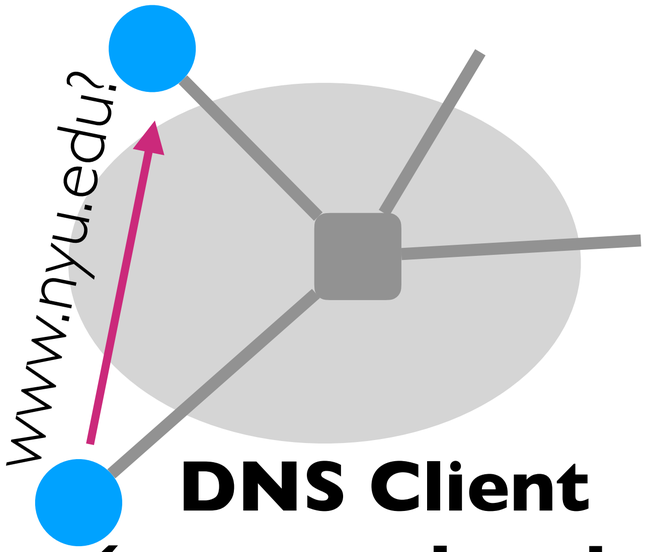

Root
DNS Server
Local DNS server
(mydns.yale.edu)
.edu Servers
www.nyu.edu?
DNS Client
(me.cs.yale.edu)
nyu.edu Servers

Recursive DNS Query



Root
DNS Server

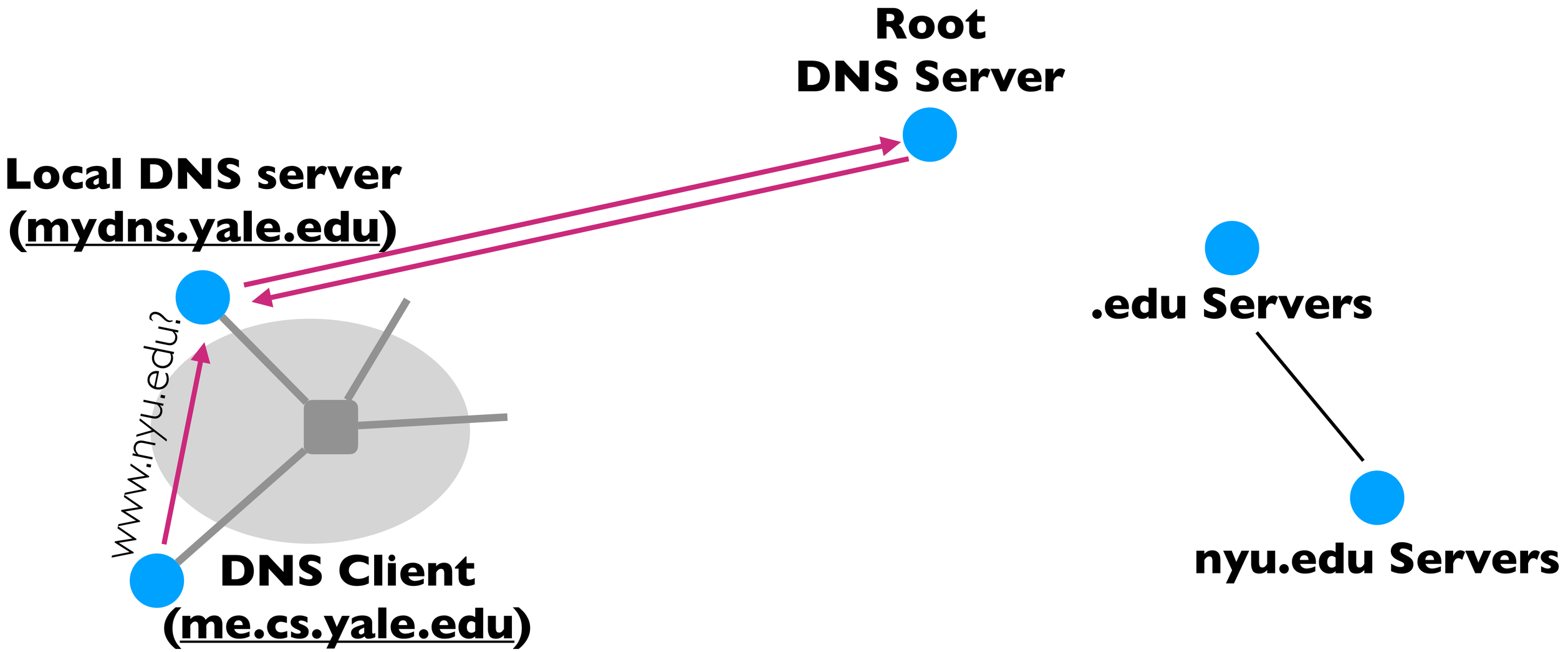
Local DNS server
(mydns.yale.edu)

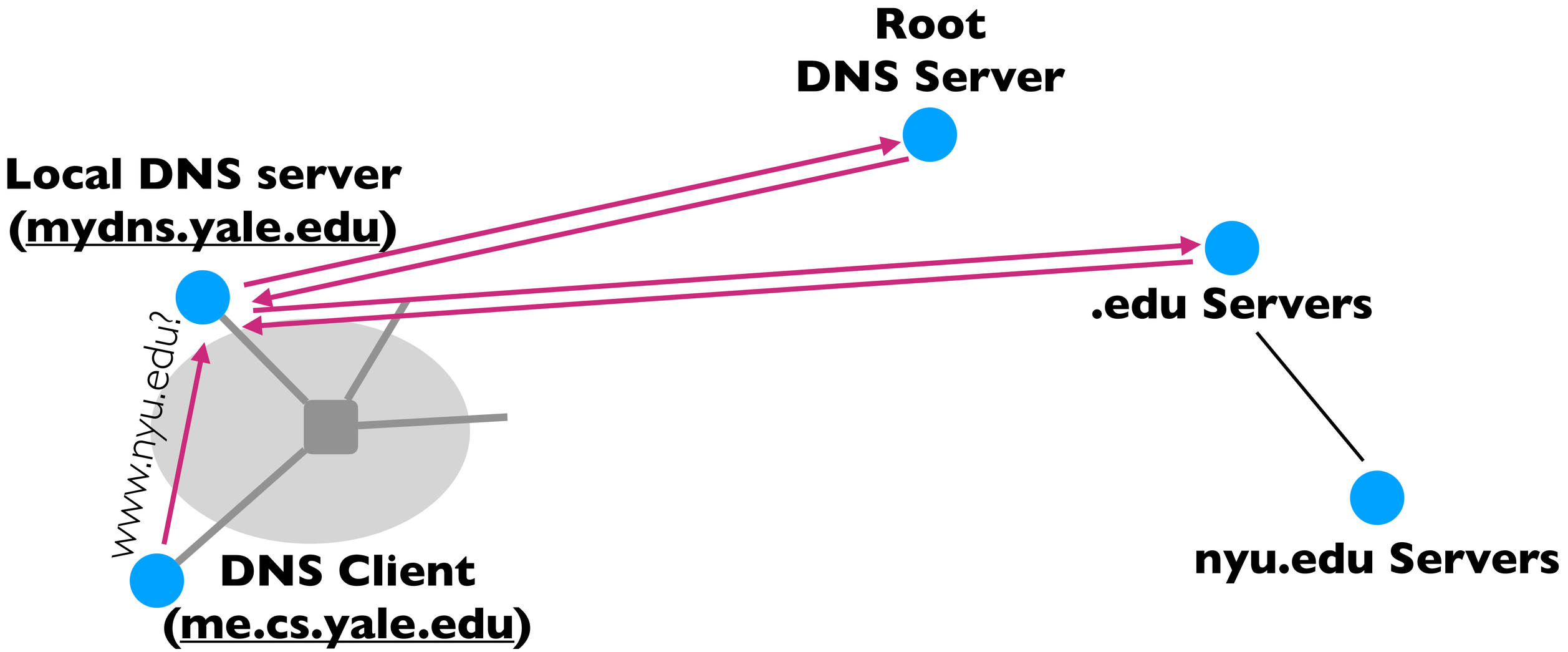


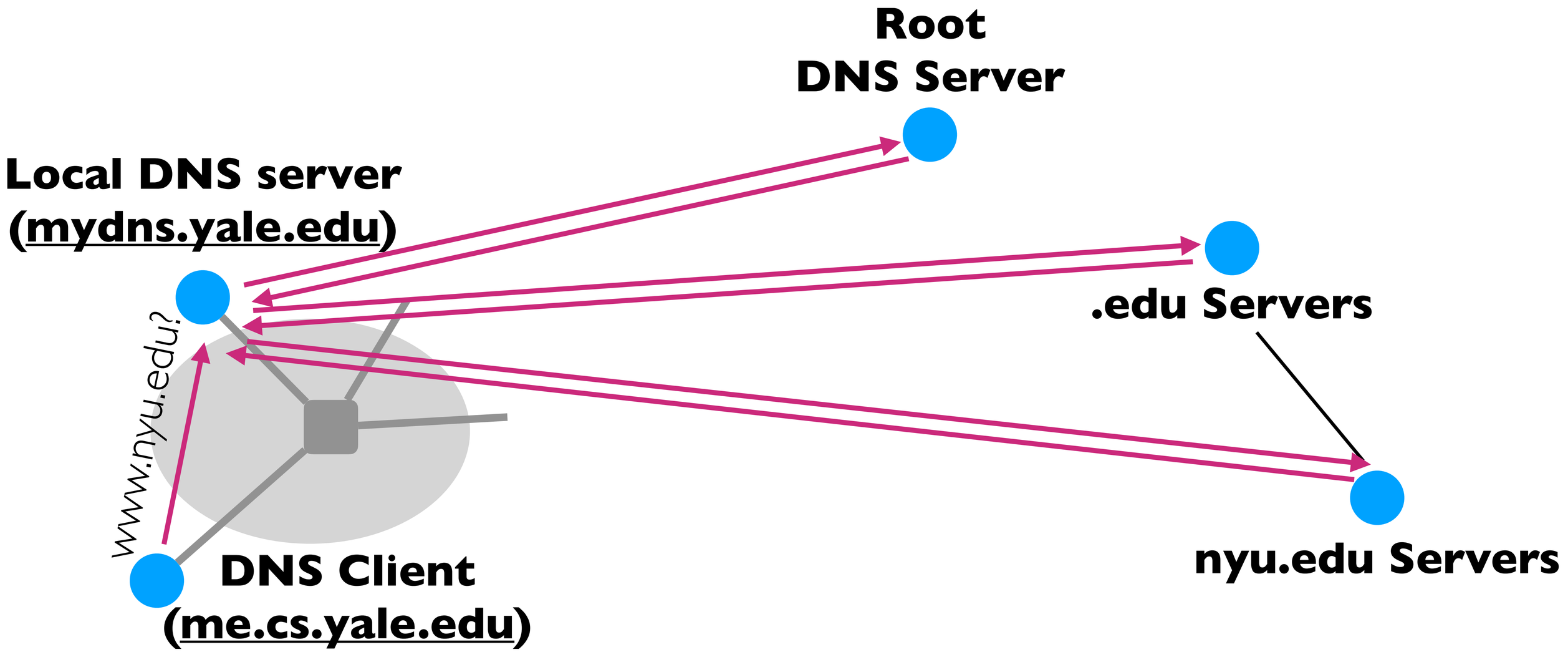
DNS Client
(me.cs.yale.edu)

.edu Servers

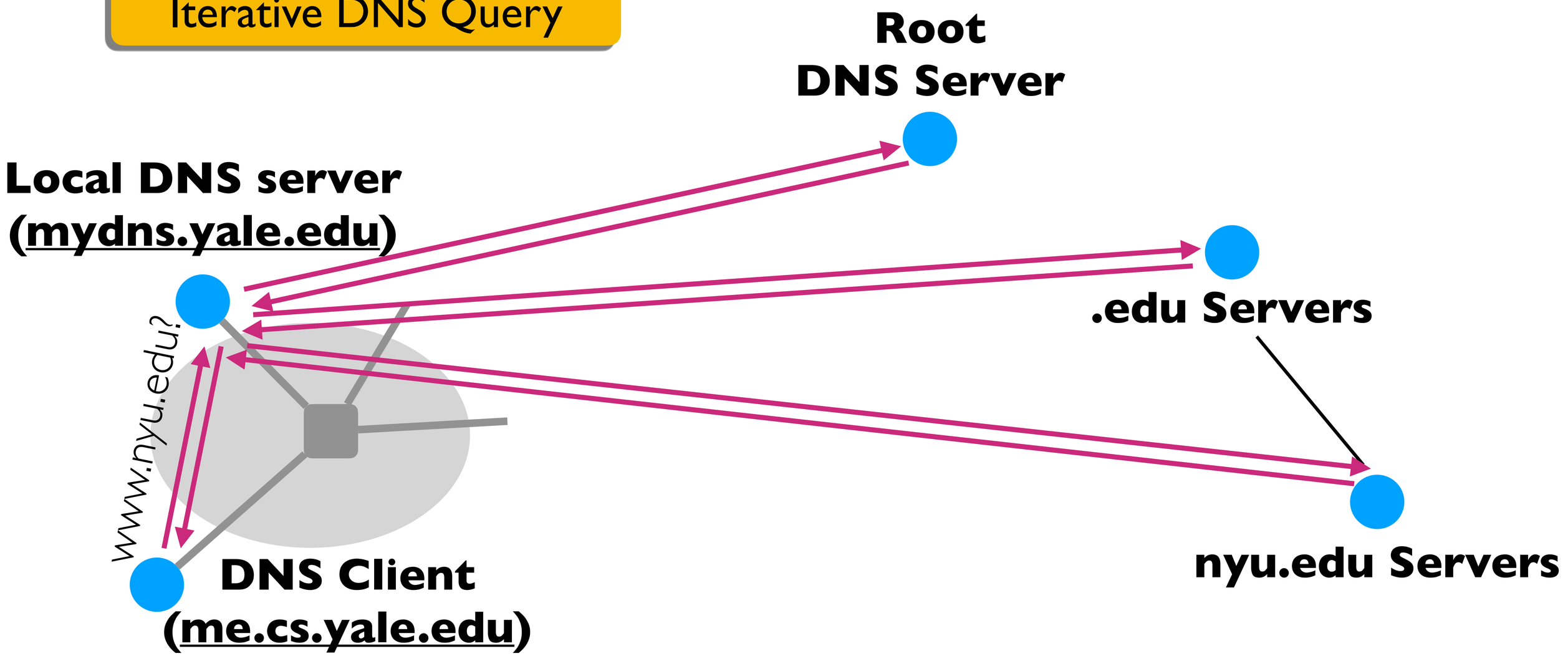
nyu.edu Servers


Root
DNS Server
Local DNS server
(mydns.yale.edu)
www.nyu.edu?
DNS Client
(me.cs.yale.edu)
.edu Servers
nyu.edu Servers


Root
DNS Server
Local DNS server
(mydns.yale.edu)
.edu Servers
www.nyu.edu?
DNS Client
(me.cs.yale.edu)
nyu.edu Servers


Root
DNS Server
Local DNS server
(mydns.yale.edu)
.edu Servers
www.nyu.edu?
DNS Client
(me.cs.yale.edu)
nyu.edu Servers

Iterative DNS Query



DNS Protocol

DNS Protocol

- Query and Reply messages; both with the same message format
  - *See text for details*

DNS Protocol

- **Query and Reply messages; both with the same message format**
  - *See text for details*

- **Client-Server Interaction on UDP Port 53**
  - Spec. supports TCP too, but not always implemented

Goals: How are we doing?

- Scalable
  - Many names
  - Many updates
  - Many users creating names
  - Many users looking up names
- Highly available
- Correct
  - No naming conflicts (uniqueness)
  - Consistency
- Lookups are fast

Per-domain Availability

Per-domain Availability

- DNS servers are **replicated**
  - Primary and secondary name servers required
  - Name service available if at least one replica is up
  - Queries can be load-balanced between replicas

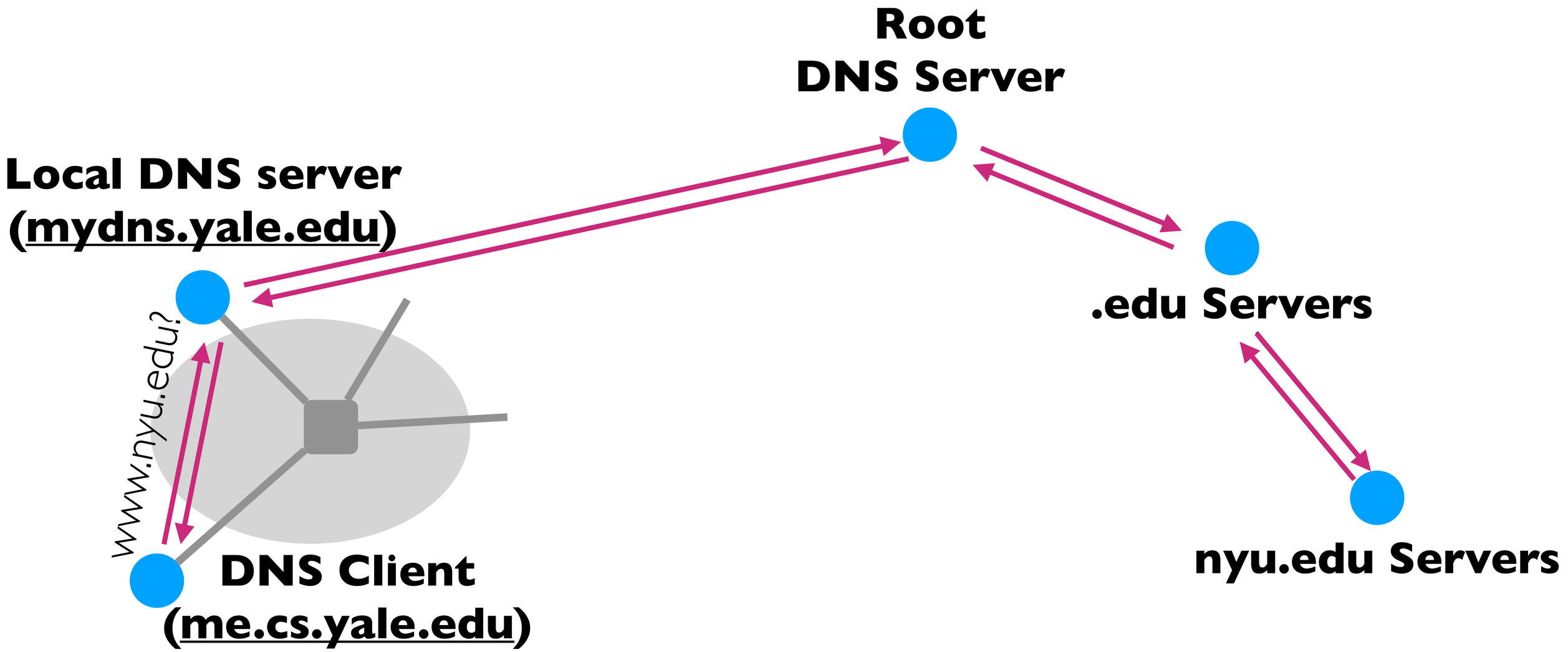
Per-domain Availability

- DNS servers are **replicated**
  - Primary and secondary name servers required
  - Name service available if at least one replica is up
  - Queries can be load-balanced between replicas

- **Try alternate servers on timeout**
  - *Exponential backoff* when retrying same server

Goals: How are we doing?

- Scalable
  - Many names
  - Many updates
  - Many users creating names
  - Many users looking up names
- Highly available
- Correct
  - No naming conflicts (uniqueness)
  - Consistency
- Lookups are fast


Root
DNS Server
Local DNS server
(mydns.yale.edu)
.edu Servers
www.nyu.edu?
DNS Client
(me.cs.yale.edu)
nyu.edu Servers

Caching

Caching

- Caching of DNS responses at all levels

Caching

- Caching of DNS responses at all levels
- Reduces load at all levels

Caching

- Caching of DNS responses at all levels

- Reduces load at all levels
- Reduces delay experienced by DNS client

DNS Caching

DNS Caching

- **How DNS caching works**
  - DNS servers cache responses to queries
  - Responses include a "time-to-live" (TTL) field
  - Server deletes cached entry after TTL expires

DNS Caching

- **How DNS caching works**
  - DNS servers cache responses to queries
  - Responses include a "time-to-live" (TTL) field
  - Server deletes cached entry after TTL expires

- **Why caching is effective**
  - The top-level servers very rarely change
  - Popular sites visited often → local DNS server often has the information cached

Negative Caching

Negative Caching

- **Remember things that don't work**
  - Misspellings like *www.cnn.comm* and *www.cnnn.com*
  - These can take a long time to fail the first time
  - Good to remember that they don't work
  - … so the failure takes less time the next time around

Negative Caching

- **Remember things that don't work**
  - Misspellings like *www.cnn.comm* and *www.cnnn.com*
  - These can take a long time to fail the first time
  - Good to remember that they don't work
  - … so the failure takes less time the next time around
- **Negative caching is optional**

Time to put your malicious hats on…

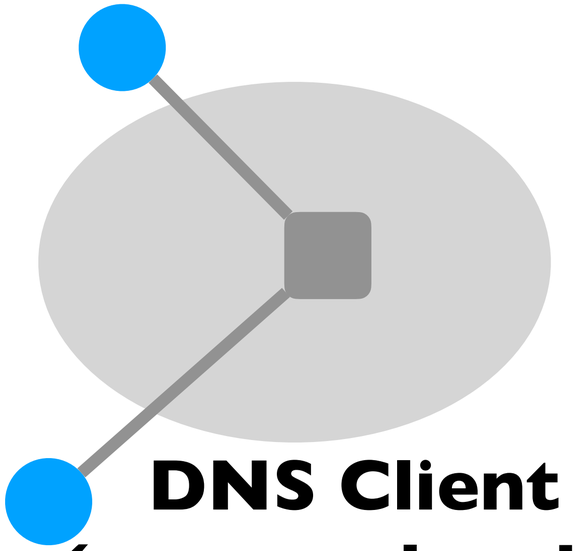
How can one attach DNS?

Root
DNS Server

.edu Servers

nyu.edu Servers

Local DNS server (mydns.yale.edu)



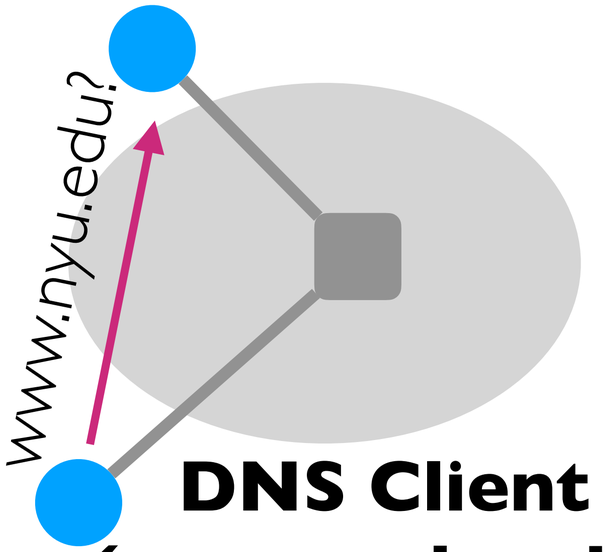
DNS Client (me.cs.yale.edu)

Root
DNS Server

.edu Servers

nyu.edu Servers

Local DNS server (mydns.yale.edu)



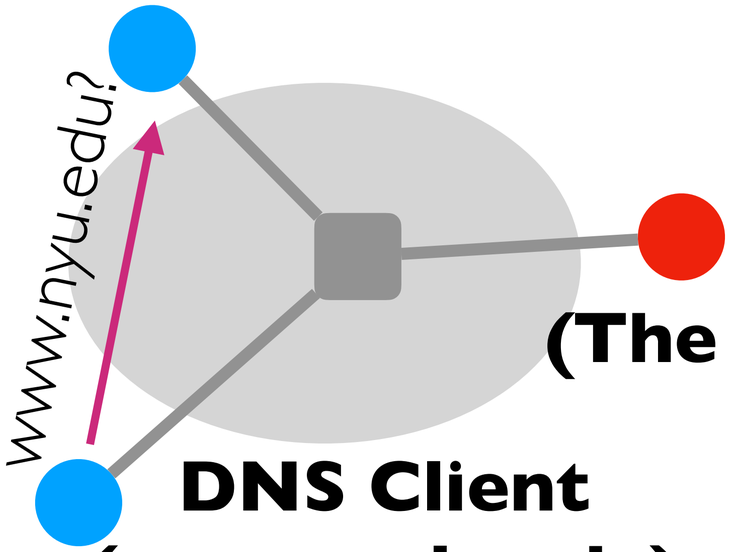
DNS Client (me.cs.yale.edu)

Root
DNS Server

.edu Servers

nyu.edu Servers

Local DNS server
(mydns.yale.edu)



Persa
(The impersonator)

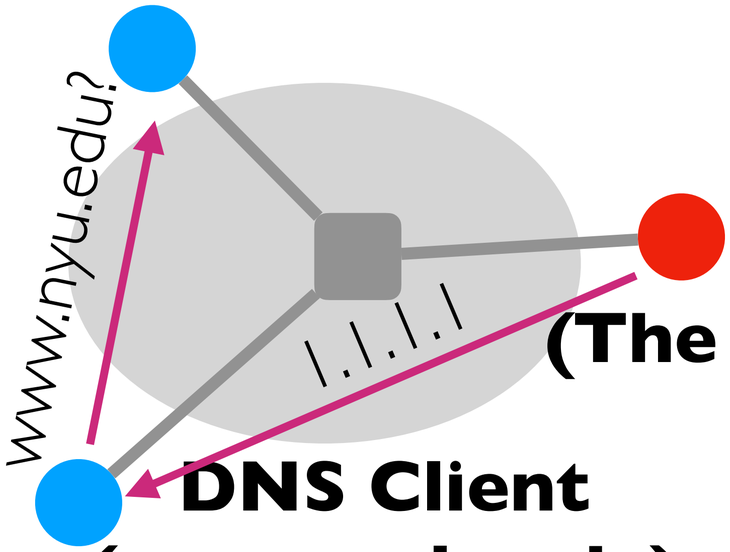
DNS Client
(me.cs.yale.edu)

Root
DNS Server

.edu Servers

nyu.edu Servers

Local DNS server
(mydns.yale.edu)



Persa
(The impersonator)

DNS Client
(me.cs.yale.edu)

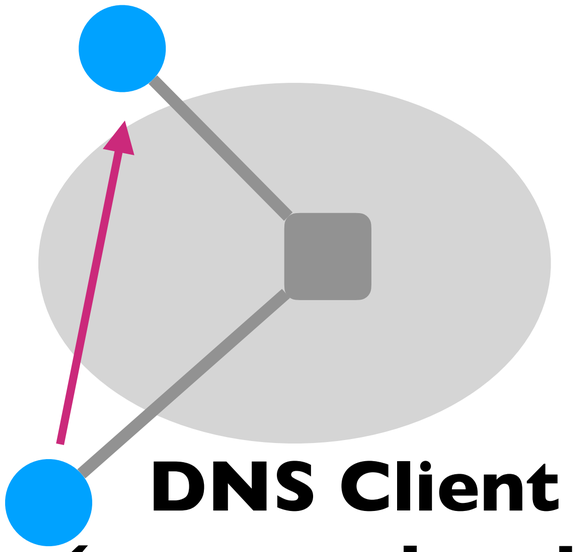
How Can One Attack DNS?

How Can One Attack DNS?

- **Impersonate the local DNS server**
  - Give the wrong IP address to the DNS client

Root
DNS Server

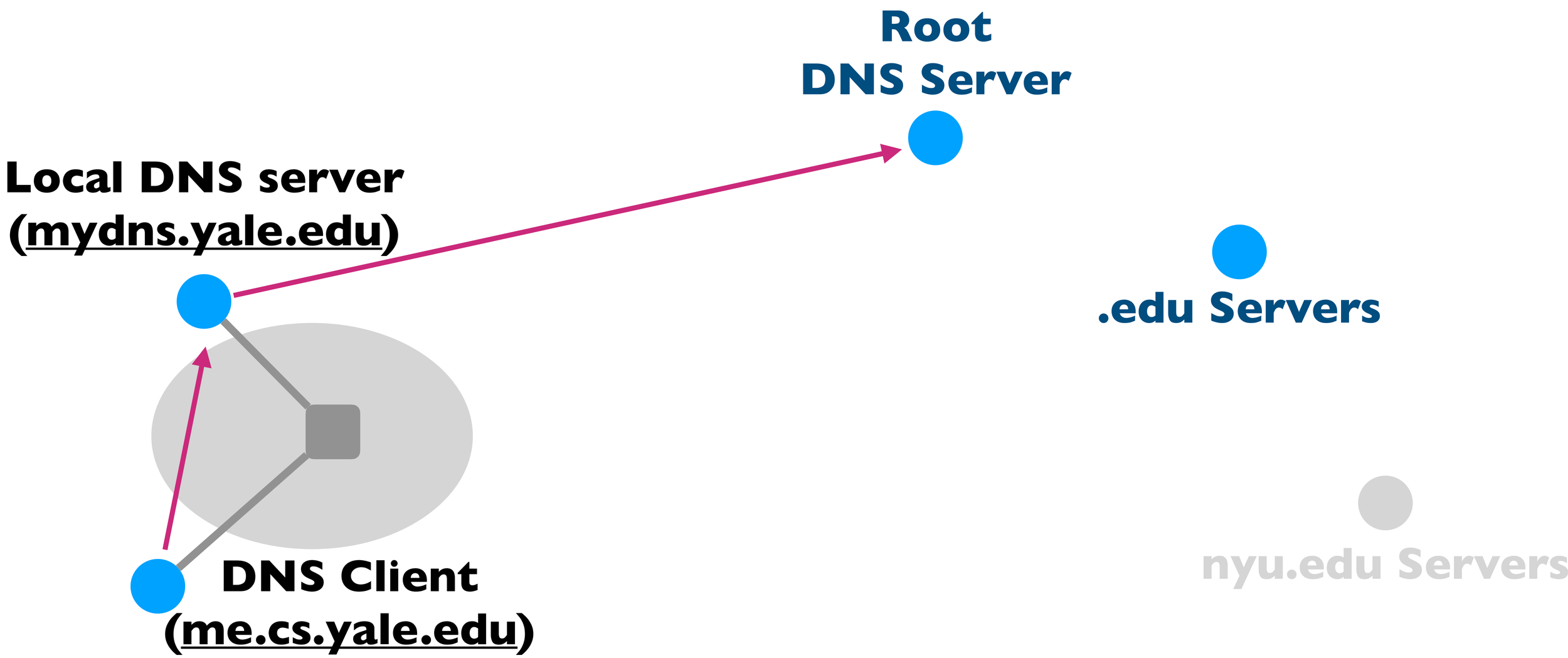
Local DNS server
(mydns.yale.edu)

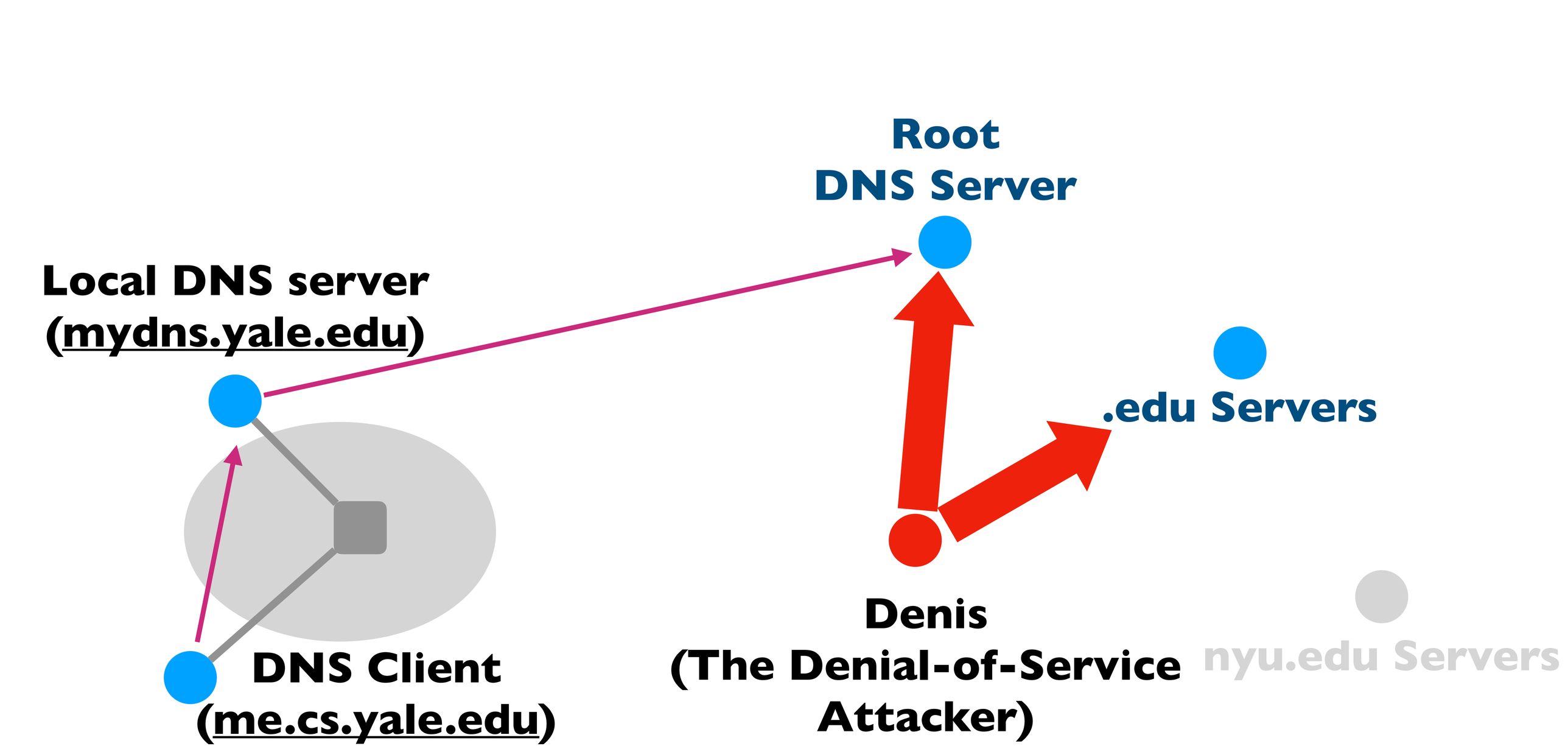


.edu Servers

nyu.edu Servers

DNS Client
(me.cs.yale.edu)


Root
DNS Server
Local DNS server
(mydns.yale.edu)
.edu Servers
nyu.edu Servers
DNS Client
(me.cs.yale.edu)


Root
DNS Server
Local DNS server
(mydns.yale.edu)
.edu Servers
Denis
(The Denial-of-Service
Attacker)
nyu.edu Servers
DNS Client
(me.cs.yale.edu)

How Can One Attack DNS?

- **Impersonate the local DNS server**
  - Give the wrong IP address to the DNS client

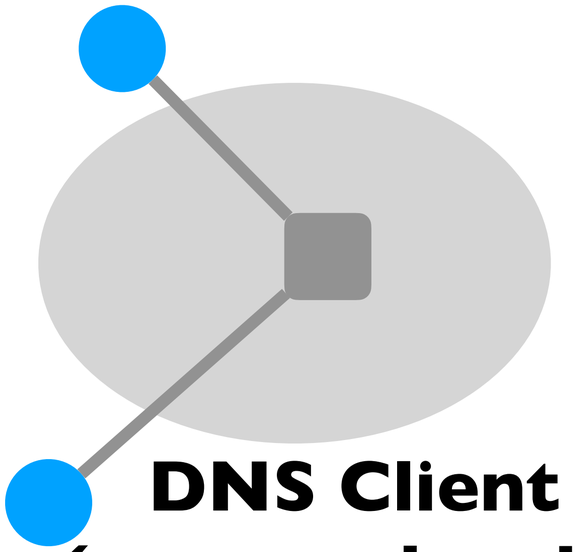
- **Denial-of-service the root or TLD servers**
  - Make them unavailable to the rest of the world

Root
DNS Server

.edu Servers

nyu.edu Servers

Local DNS server (mydns.yale.edu)



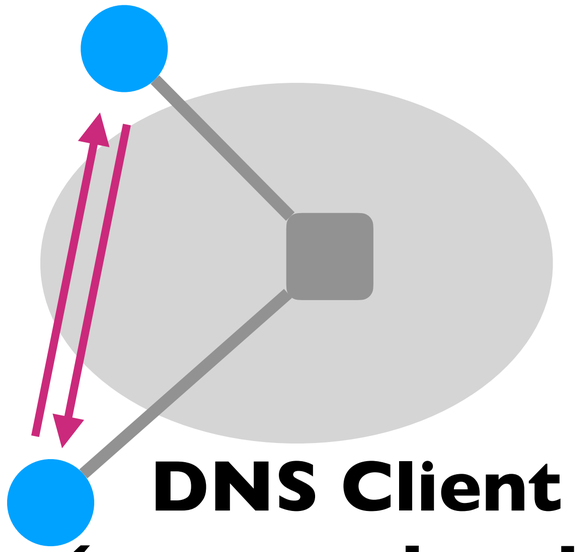
DNS Client (me.cs.yale.edu)

Root
DNS Server

.edu Servers

nyu.edu Servers

Local DNS server
(mydns.yale.edu)



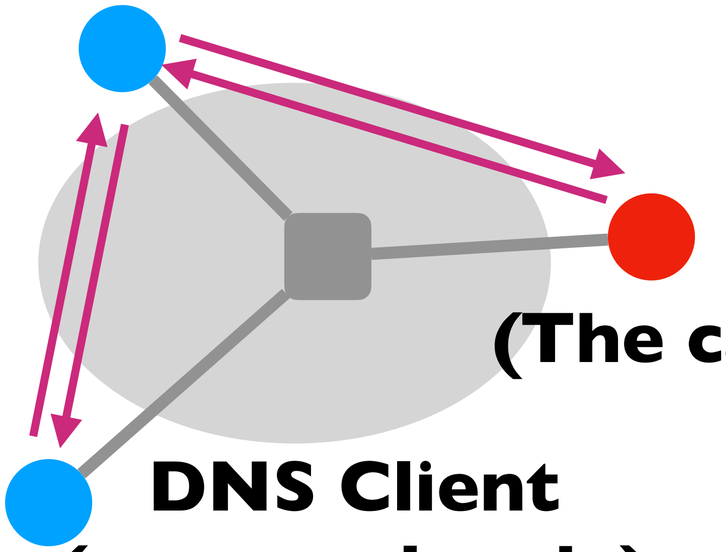
DNS Client
(me.cs.yale.edu)

Root
DNS Server

.edu Servers

nyu.edu Servers

Local DNS server
(mydns.yale.edu)



Cacha
(The cache poisoner)

DNS Client
(me.cs.yale.edu)

How Can One Attack DNS?

- **Impersonate the local DNS server**
  - Give the wrong IP address to the DNS client
- **Denial-of-service the root or TLD servers**
  - Make them unavailable to the rest of the world
- **Poison the cache of a DNS server**
  - Increase the delay experienced by DNS clients

Important Properties of DNS

Important Properties of DNS

Administrative delegation and hierarchy results in:

Important Properties of DNS

Administrative delegation and hierarchy results in:

- Easy unique naming

Important Properties of DNS

Administrative delegation and hierarchy results in:

- Easy unique naming
- “Fate sharing” for network failures

Important Properties of DNS

Administrative delegation and hierarchy results in:

- Easy unique naming
- “Fate sharing” for network failures
- Reasonable trust model

Important Properties of DNS

Administrative delegation and hierarchy results in:

- Easy unique naming
- “Fate sharing” for network failures
- Reasonable trust model
- Caching lends scalability, performance

DNS Provides Indirection

DNS Provides Indirection

- Addresses can **change** underneath
  - Move *www.cnn.com* to a new IP address
  - Humans/applications are unaffected

DNS Provides Indirection

- Addresses can **change** underneath
  - Move *www.cnn.com* to a new IP address
  - Humans/applications are unaffected

- Name could map to **multiple** IP addresses
  - Enables load-balancing

DNS Provides Indirection

- Addresses can **change** underneath
  - Move *www.cnn.com* to a new IP address
  - Humans/applications are unaffected
- Name could map to **multiple** IP addresses
  - Enables load-balancing
- Multiple names for the same addresses
  - E.g., many services (mail, www, ftp) on same machine

DNS Provides Indirection

- Addresses can **change** underneath
  - Move *www.cnn.com* to a new IP address
  - Humans/applications are unaffected
- Name could map to **multiple** IP addresses
  - Enables load-balancing
- Multiple names for the same addresses
  - E.g., many services (mail, www, ftp) on same machine
- Allowing "host" names to evolve into "service" names