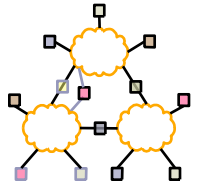
CE693: Adv. Computer Networking

L-16 P2P and DNS

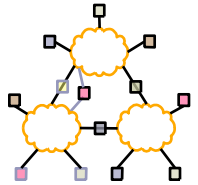
Acknowledgments: Lecture slides are from the graduate level Computer Networks course thought by Srinivasan Seshan at CMU. When slides are obtained from other sources, a a reference will be noted on the bottom of that slide. A full list of references is provided on the last slide.

Overview



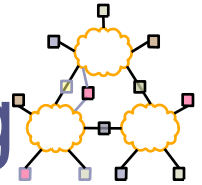
- P2P
- DNS

Peer-to-Peer Networks: BitTorrent



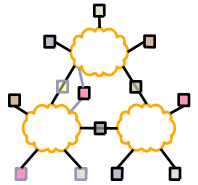
- BitTorrent history and motivation
  - 2002: B. Cohen debuted BitTorrent
  - Key motivation: popular content
    - Popularity exhibits temporal locality (Flash Crowds)
    - E.g., Slashdot/Digg effect, CNN Web site on 9/11, release of a new movie or game
  - Focused on efficient fetching, not searching
    - Distribute same file to many peers
    - Single publisher, many downloaders
  - Preventing free-loading

BitTorrent: Simultaneous Downloading



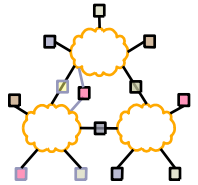
- Divide large file into many pieces
  - Replicate different pieces on different peers
  - A peer with a complete piece can trade with other peers
  - Peer can (hopefully) assemble the entire file
- Allows simultaneous downloading
  - Retrieving different parts of the file from different peers at the same time
  - And uploading parts of the file to peers
  - Important for very large files

BitTorrent: Tracker



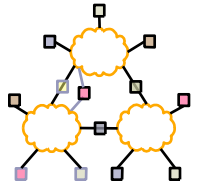
- Infrastructure node
  - Keeps track of peers participating in the torrent
- Peers register with the tracker
  - Peer registers when it arrives
  - Peer periodically informs tracker it is still there
- Tracker selects peers for downloading
  - Returns a random set of peers
  - Including their IP addresses
  - So the new peer knows who to contact for data
- Can have “trackerless” system using DHT

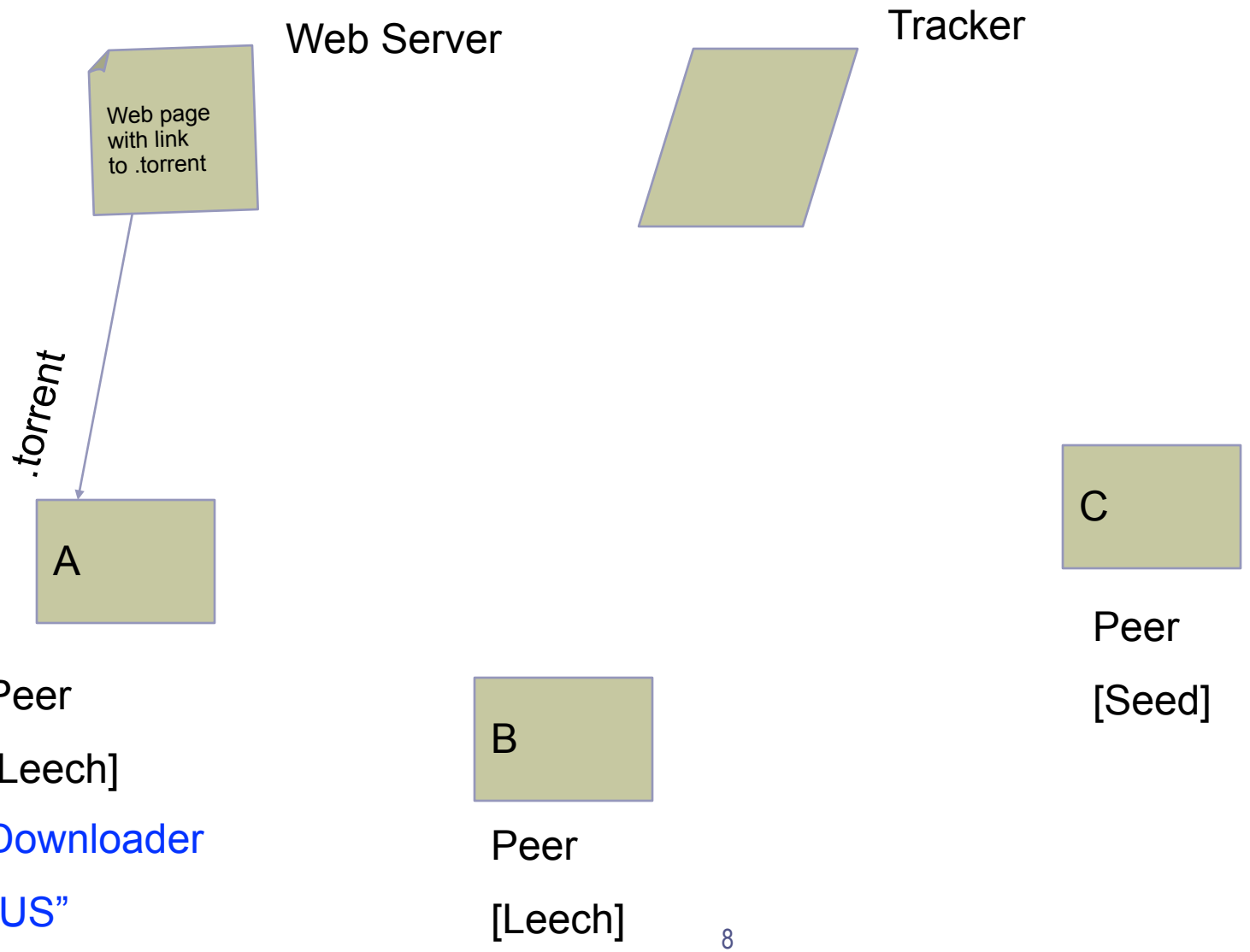
BitTorrent: Chunks



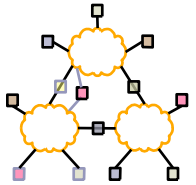
- Large file divided into smaller pieces
  - Fixed-sized chunks
  - Typical chunk size of 256 Kbytes
- Allows simultaneous transfers
  - Downloading chunks from different neighbors
  - Uploading chunks to other neighbors
- Learning what chunks your neighbors have
  - Periodically asking them for a list
- File done when all chunks are downloaded

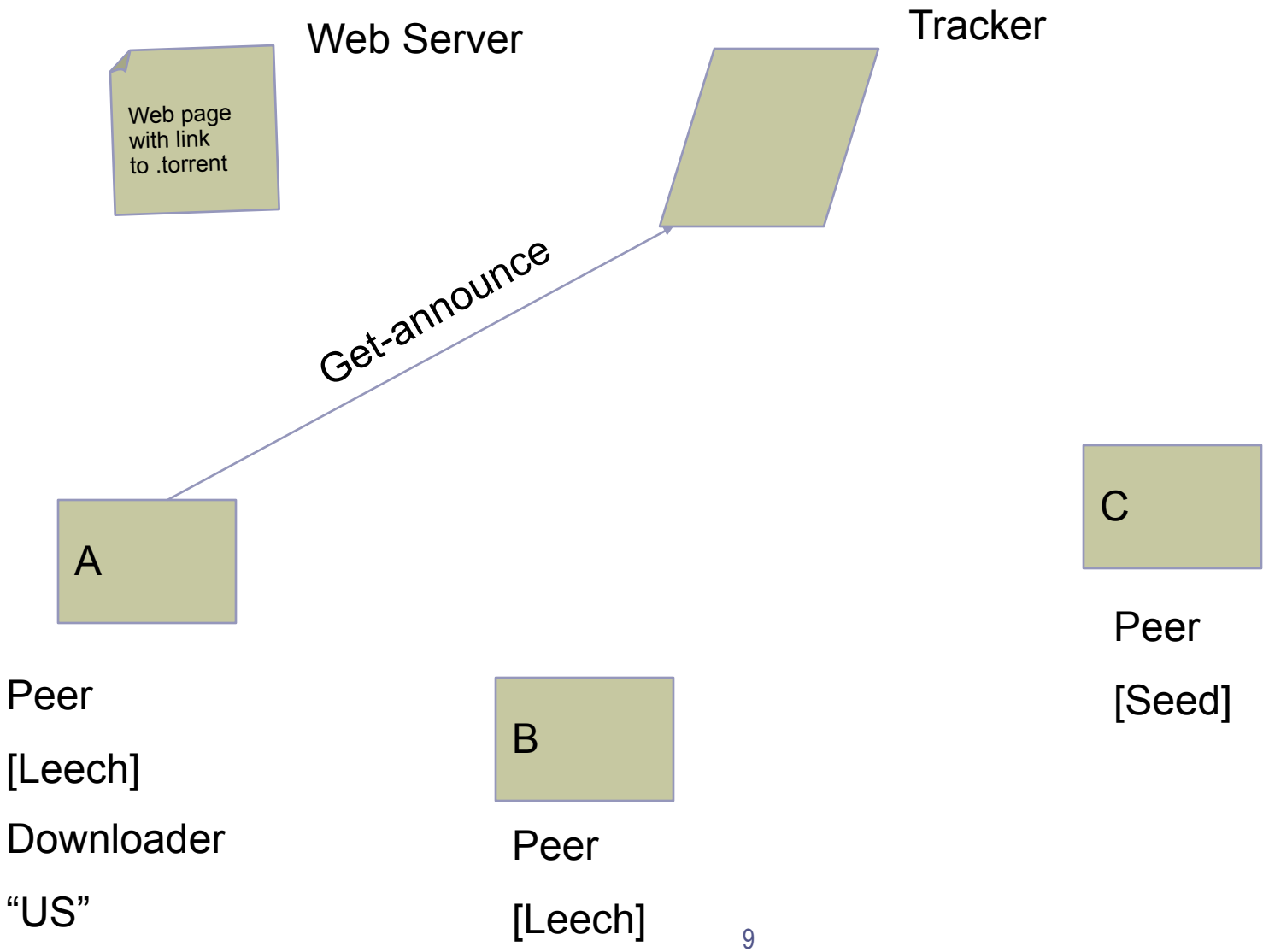
BitTorrent: Overall Architecture



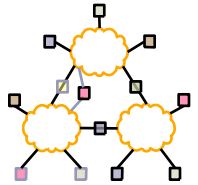


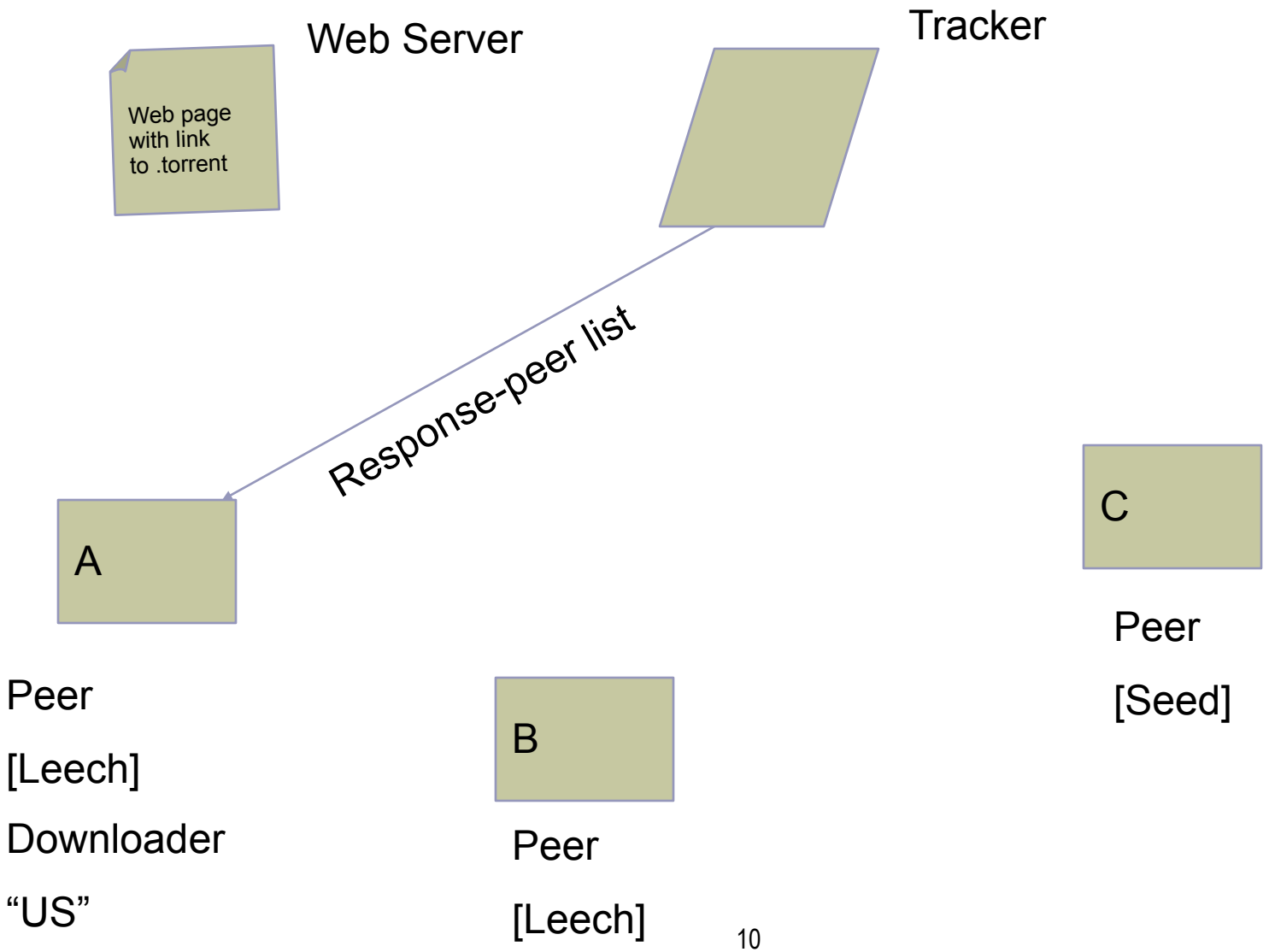
BitTorrent: Overall Architecture



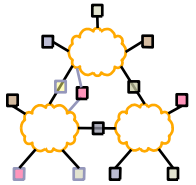


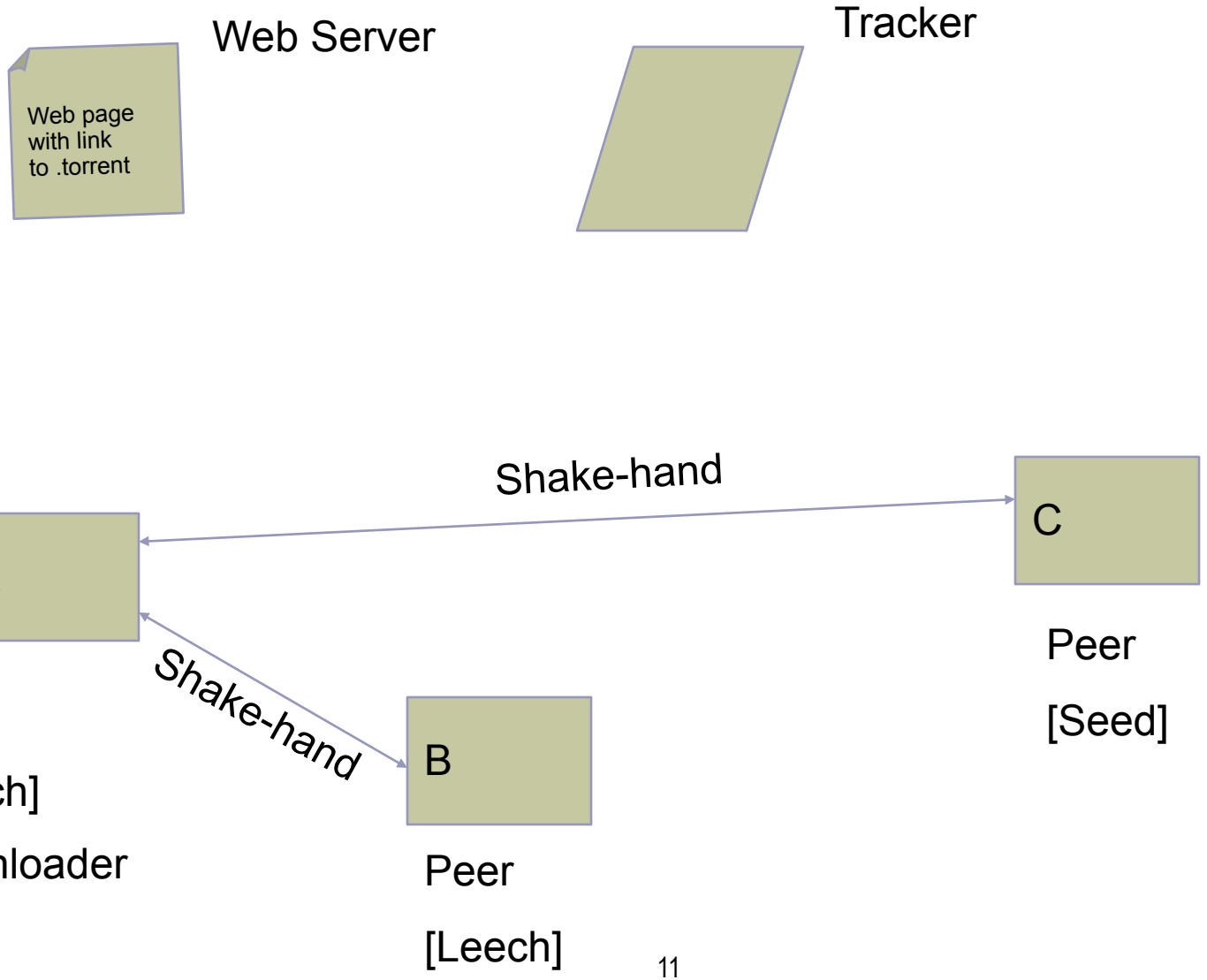
BitTorrent: Overall Architecture



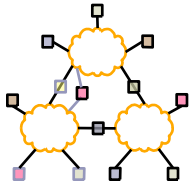


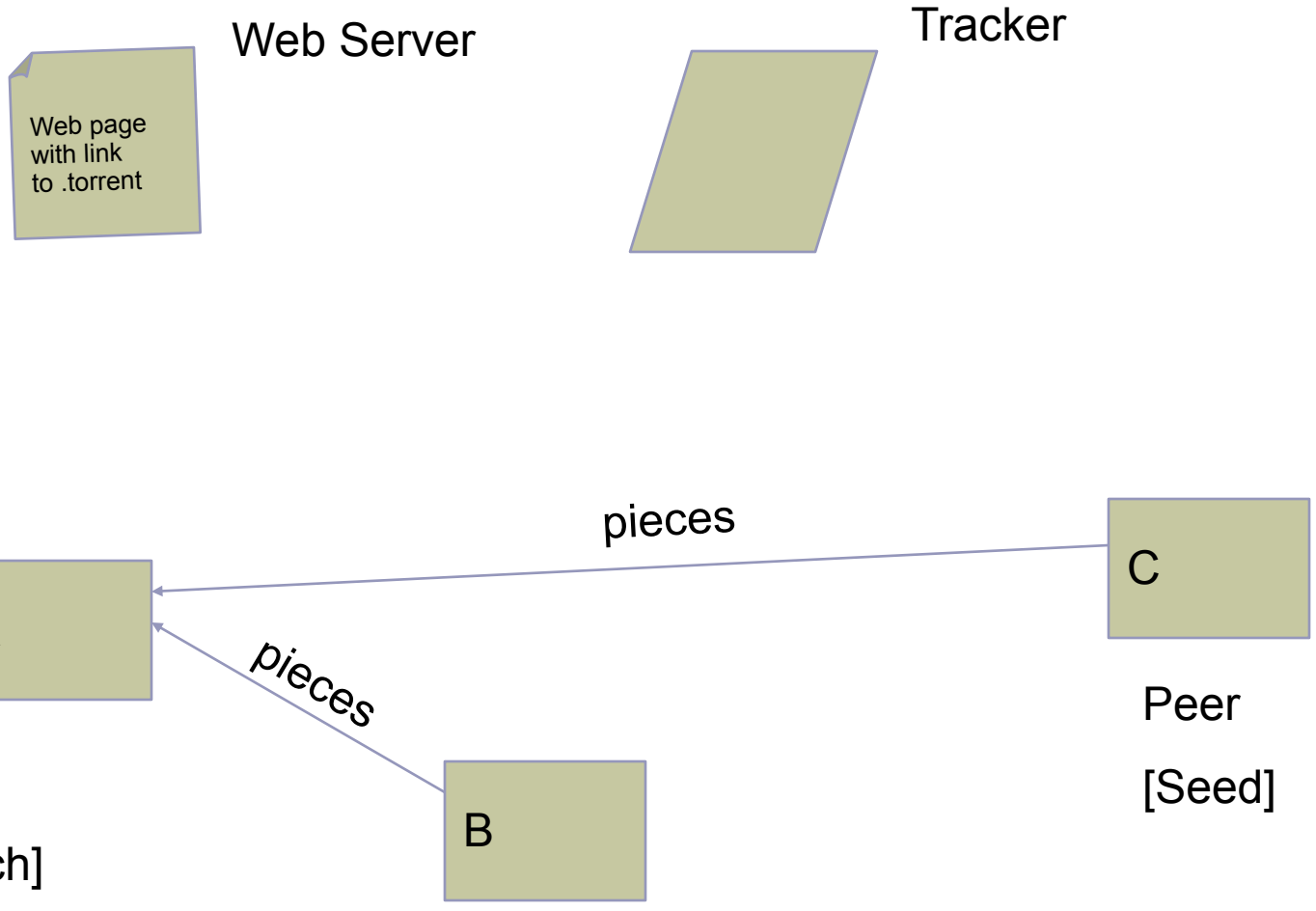
BitTorrent: Overall Architecture



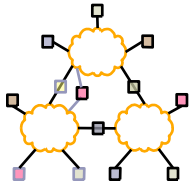


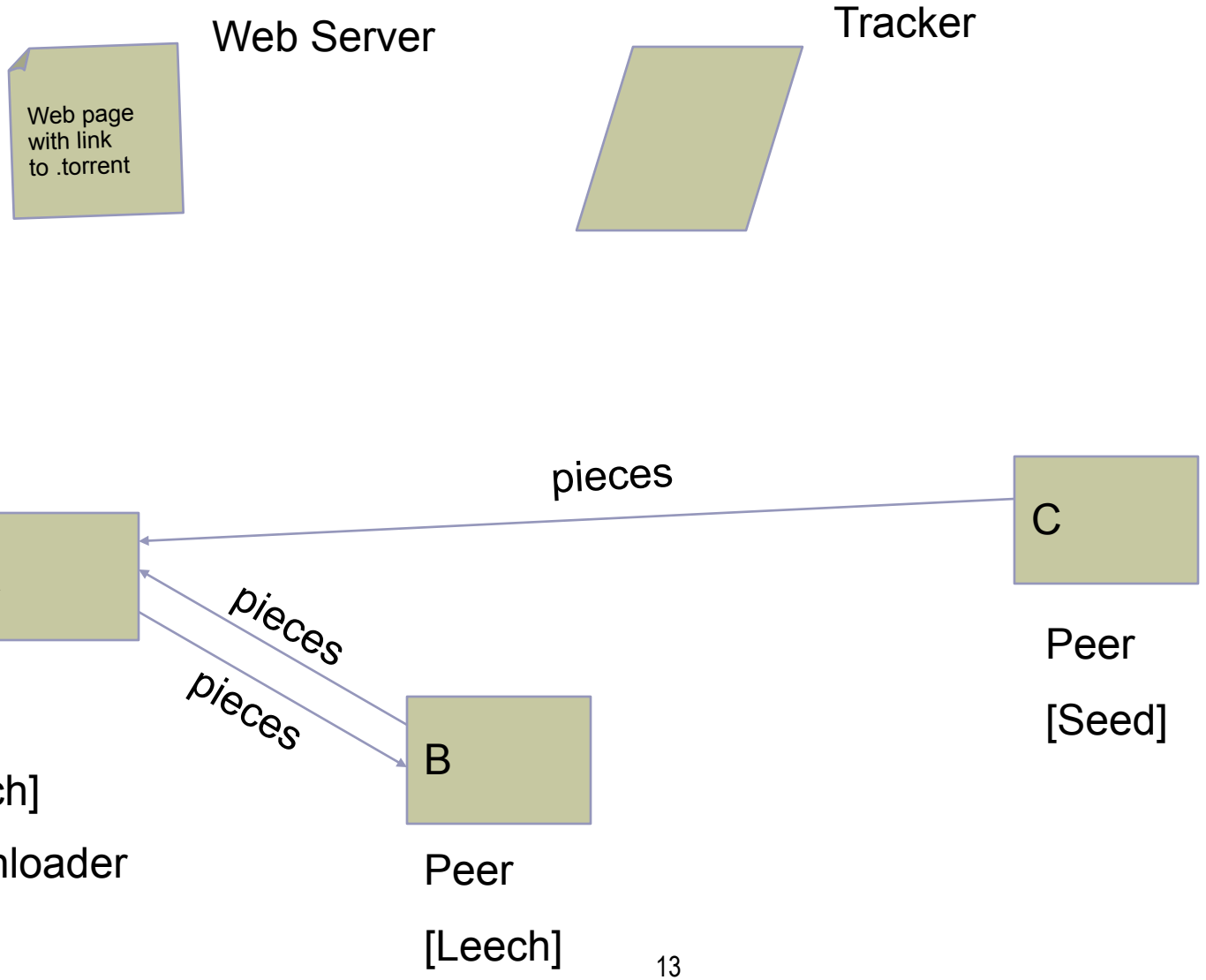
BitTorrent: Overall Architecture



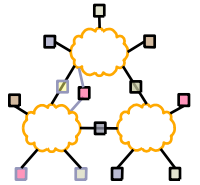


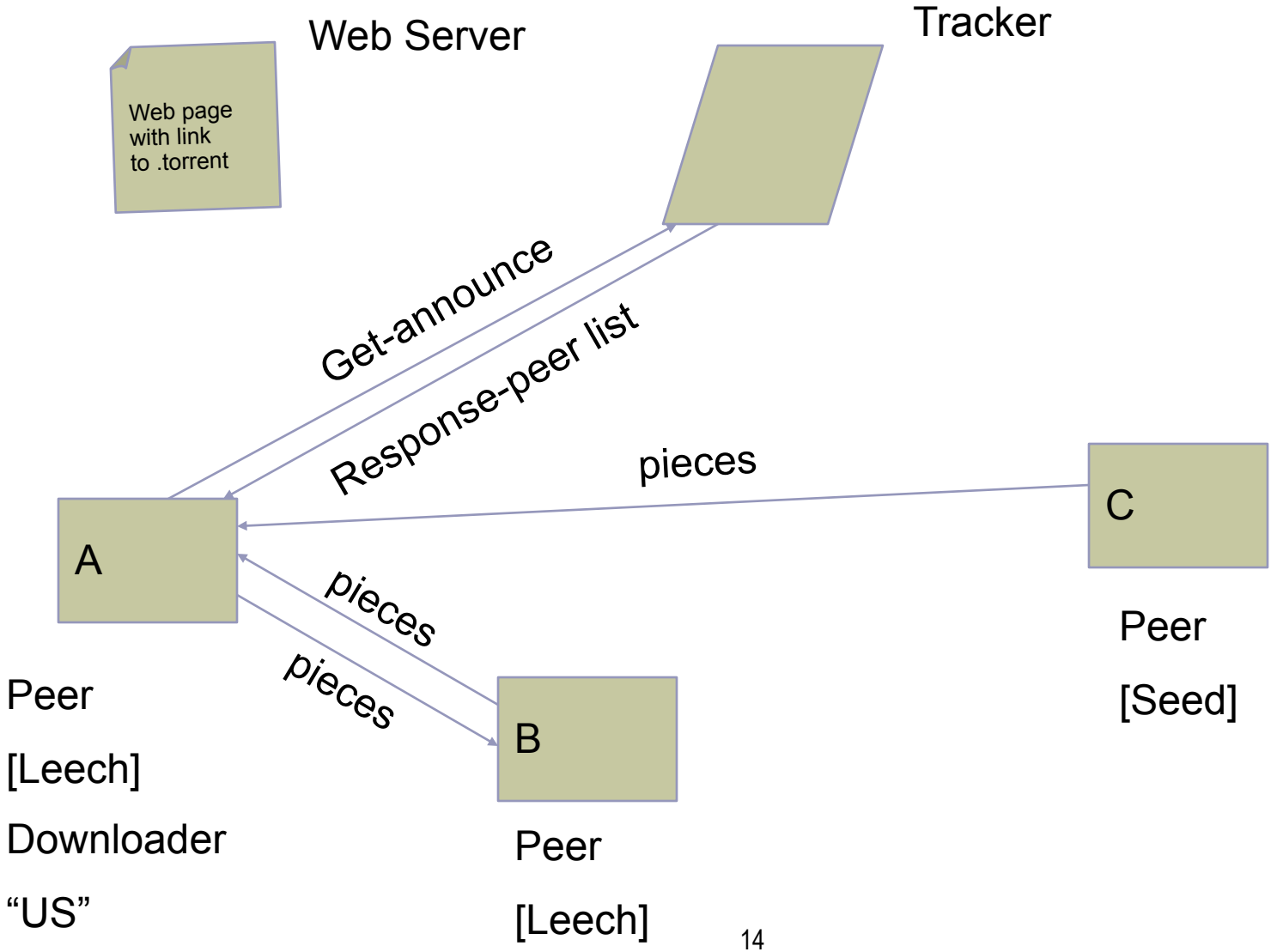
BitTorrent: Overall Architecture



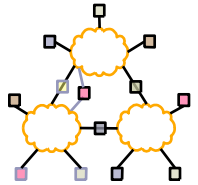


BitTorrent: Overall Architecture



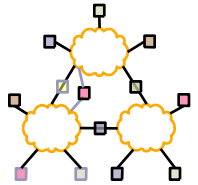


BitTorrent: Chunk Request Order



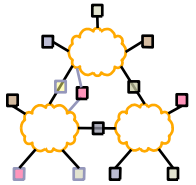
- Which chunks to request?
  - Could download in order
  - Like an HTTP client does
- Problem: many peers have the early chunks
  - Peers have little to share with each other
  - Limiting the scalability of the system
- Problem: eventually nobody has rare chunks
  - E.g., the chunks need the end of the file
  - Limiting the ability to complete a download
- Solutions: random selection and rarest first

BitTorrent: Rarest Chunk First



- Which chunks to request first?
  - The chunk with the fewest available copies
  - I.e., the rarest chunk first
- Benefits to the peer
  - Avoid starvation when some peers depart
- Benefits to the system
  - Avoid starvation across all peers wanting a file
  - Balance load by equalizing # of copies of chunks

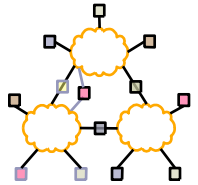
Free-Riding Problem in P2P Networks



- Vast majority of users are free-riders
  - Most share no files and answer no queries
  - Others limit # of connections or upload speed
- A few “peers” essentially act as servers
  - A few individuals contributing to the public good
  - Making them hubs that basically act as a server

- BitTorrent prevent free riding
  - Allow the fastest peers to download from you
  - Occasionally let some free loaders download

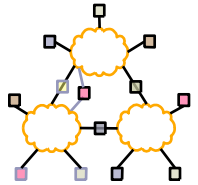
Bit-Torrent: Preventing Free-Riding



- Peer has limited upload bandwidth
  - And must share it among multiple peers
- Prioritizing the upload bandwidth: tit for tat
  - Favor neighbors that are uploading at highest rate

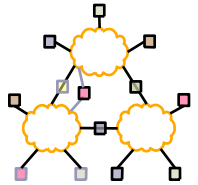
- Rewarding the top few (e.g. four) peers
  - Measure download bit rates from each neighbor
  - Reciprocates by sending to the top few peers
  - Recompute and reallocate every 10 seconds
- Optimistic unchoking
  - Randomly try a new neighbor every 30 seconds
  - To find a better partner and help new nodes startup

BitTyrant: Gaming BitTorrent



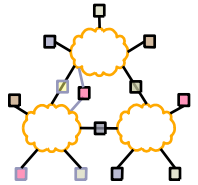
- Lots of altruistic contributors
- High contributors take a long time to find good partners
- Active sets are statically sized
  - Peer uploads to top N peers at rate 1/N
  - E.g., if N=4 and peers upload at 15, 12, 10, 9, 8, 3
  - … then peer uploading at rate 9 gets treated quite well

BitTyrant: Gaming BitTorrent



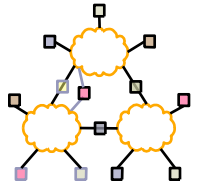
- Best to be the N^{th} peer in the list, rather than 1^{st}
  - Distribution of BW suggests 14KB/s is enough
  - Dynamically probe for this value
- Use saved bandwidth to expand peer set
  - Choose clients that maximize download/upload ratio
- Discussion
  - Is “progressive tax” so bad?
  - What if everyone does this?

Overview



- P2P
- DNS

Obvious Solutions (1)

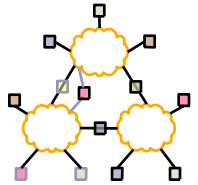


Why not centralize DNS?

- Single point of failure
- Traffic volume
- Distant centralized database
- Single point of update

- Doesn't *scale!*

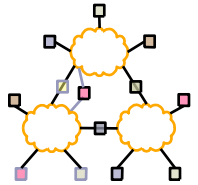
Obvious Solutions (2)



Why not use /etc/hosts?

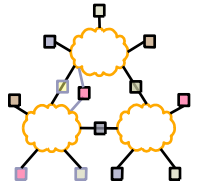
- Original Name to Address Mapping
  - Flat namespace
  - /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: machine per domain → machine per user
  - Many more downloads
  - Many more updates

Domain Name System Goals



- Basically building a wide area distributed database
- Scalability
- Decentralized maintenance
- Robustness
- Global scope
  - Names mean the same thing everywhere

DNS Records



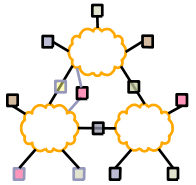
RR format: **(class, name, value, type, ttl)**

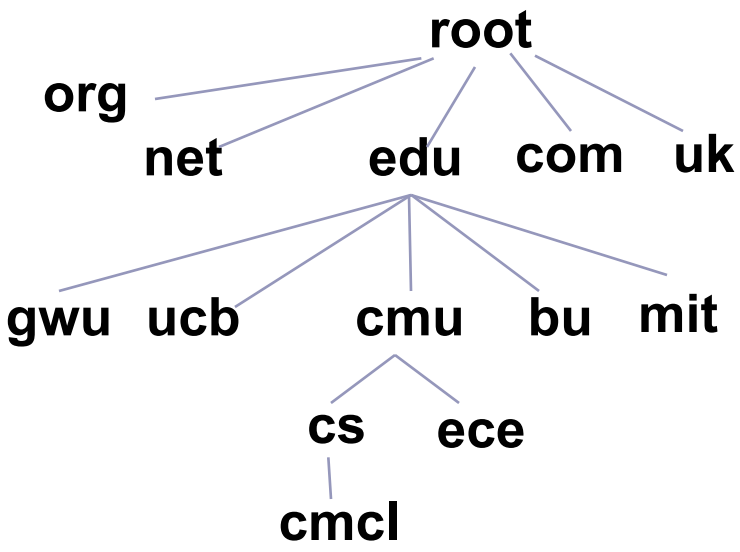
- DB contains tuples called resource records (RRs)
  - Classes = Internet (IN), Chaosnet (CH), etc.
  - Each class defines value associated with type

FOR IN class:

- Type=A
  - **name** is hostname
  - **value** is IP address
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is name of authoritative name server for this domain
- Type=CNAME
  - **name** is an alias name for some "canonical" (the real) name
  - **value** is canonical name
- Type=MX
  - **value** is hostname of mailserver associated with **name**

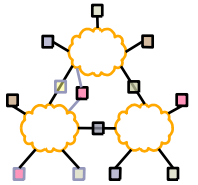
DNS Design: Hierarchy Definitions

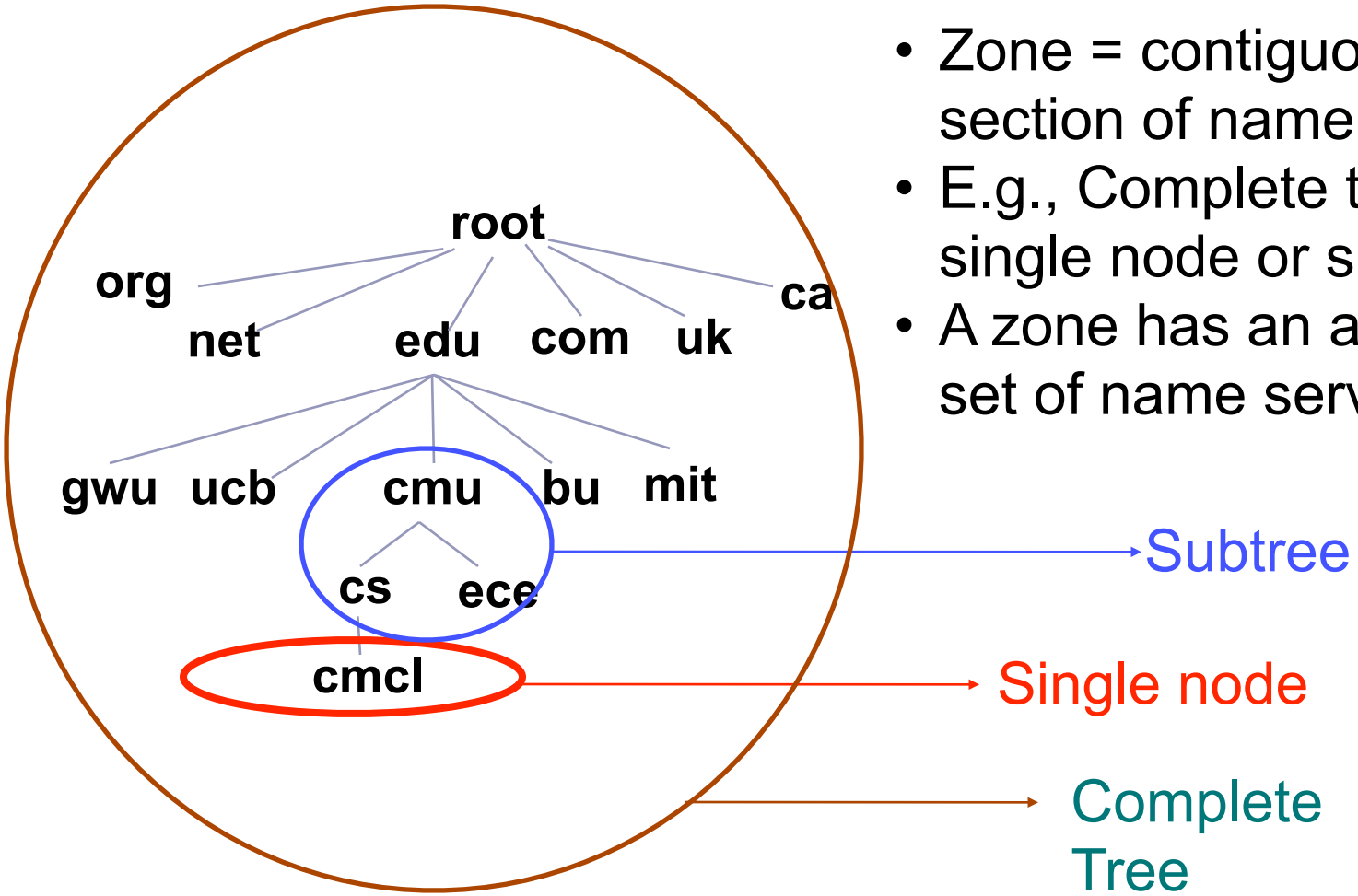




- Each node in hierarchy stores a list of names that end with same suffix
  - Suffix = path up tree
- E.g., given this tree, where would following be stored:
  - Fred.com
  - Fred.edu
  - Fred.cmu.edu
  - Fred.cmcl.cs.cmu.edu
  - Fred.cs.mit.edu

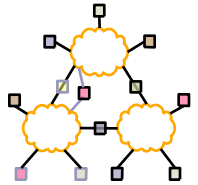
DNS Design: Zone Definitions





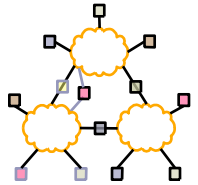
- Zone = contiguous section of name space
- E.g., Complete tree, single node or subtree
- A zone has an associated set of name servers

DNS Design: Cont.



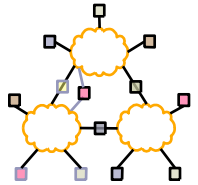
- Zones are created by convincing owner node to create/delegate a subzone
  - Records within zone stored multiple redundant name servers
  - Primary/master name server updated manually
  - Secondary/redundant servers updated by zone transfer of name space
    - Zone transfer is a bulk transfer of the “configuration” of a DNS server – uses TCP to ensure reliability
- Example:
  - CS.CMU.EDU created by CMU.EDU administrators

Servers/Resolvers

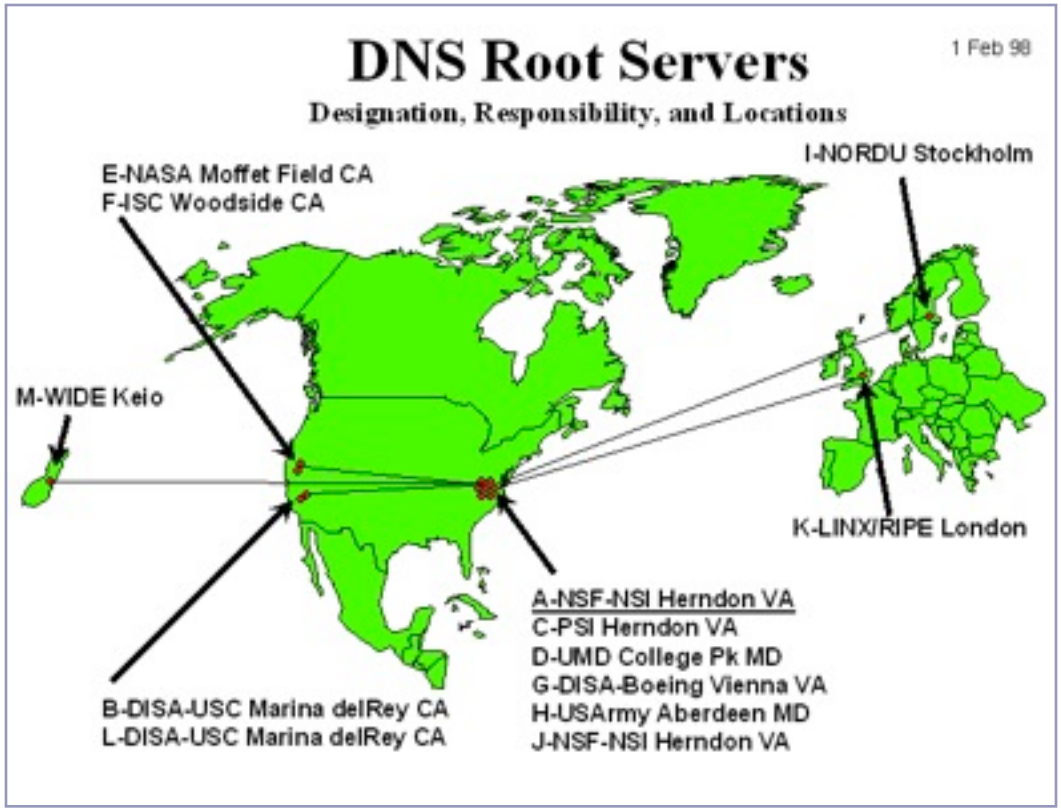


- Each host has a resolver
  - Typically a library that applications can link to
  - Local name servers hand-configured (e.g. /etc/resolv.conf)
- Name servers
  - Either responsible for some zone or…
  - Local servers
    - Do lookup of distant host names for local hosts
    - Typically answer queries about local zone

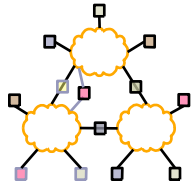
DNS: Root Name Servers

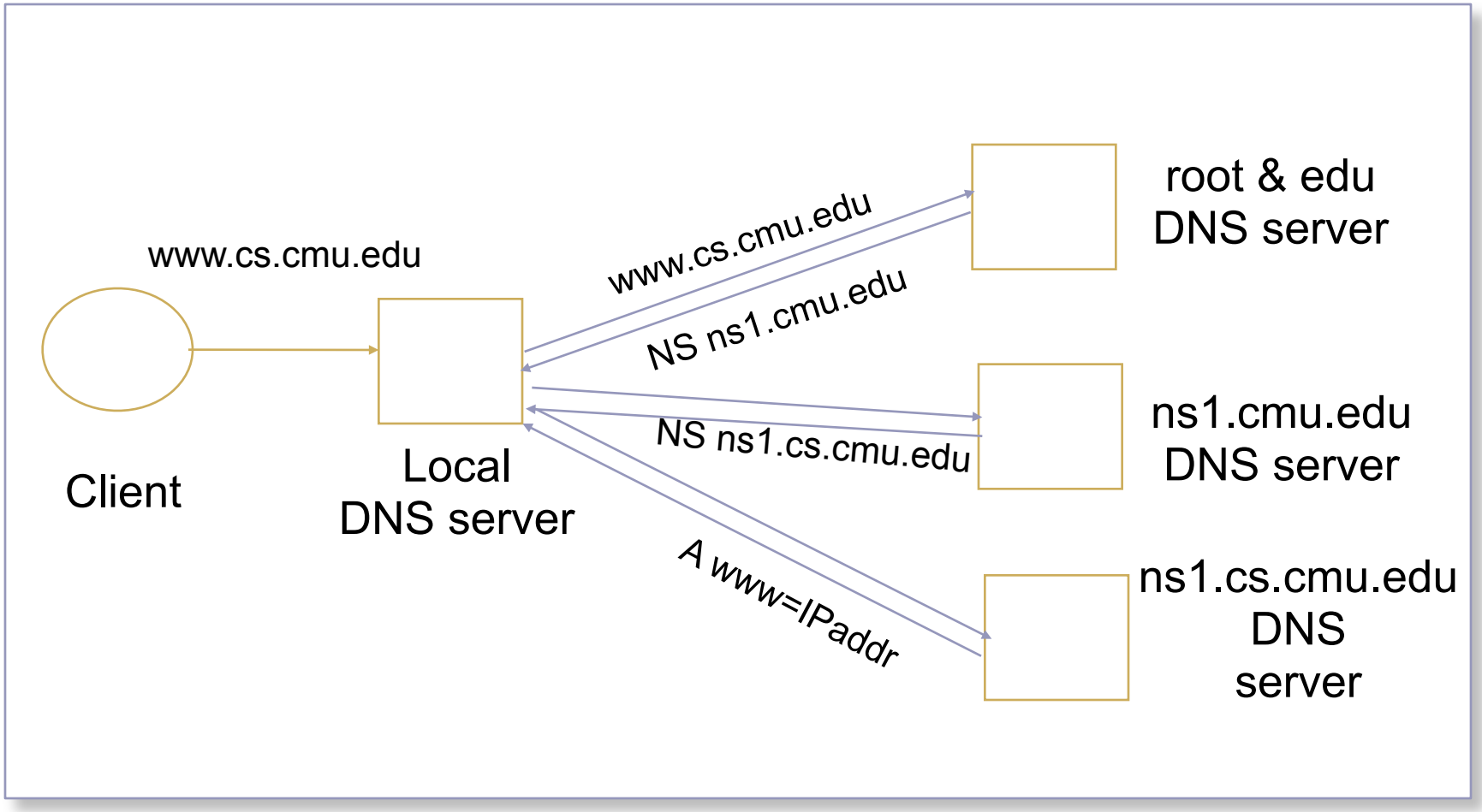


- Responsible for "root" zone
- Approx. dozen root name servers worldwide
  - Currently {a-m}.root-servers.net
- Local name servers contact root servers when they cannot resolve a name
  - Configured with well-known root servers

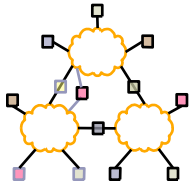


Typical Resolution





Lookup Methods



Recursive query:

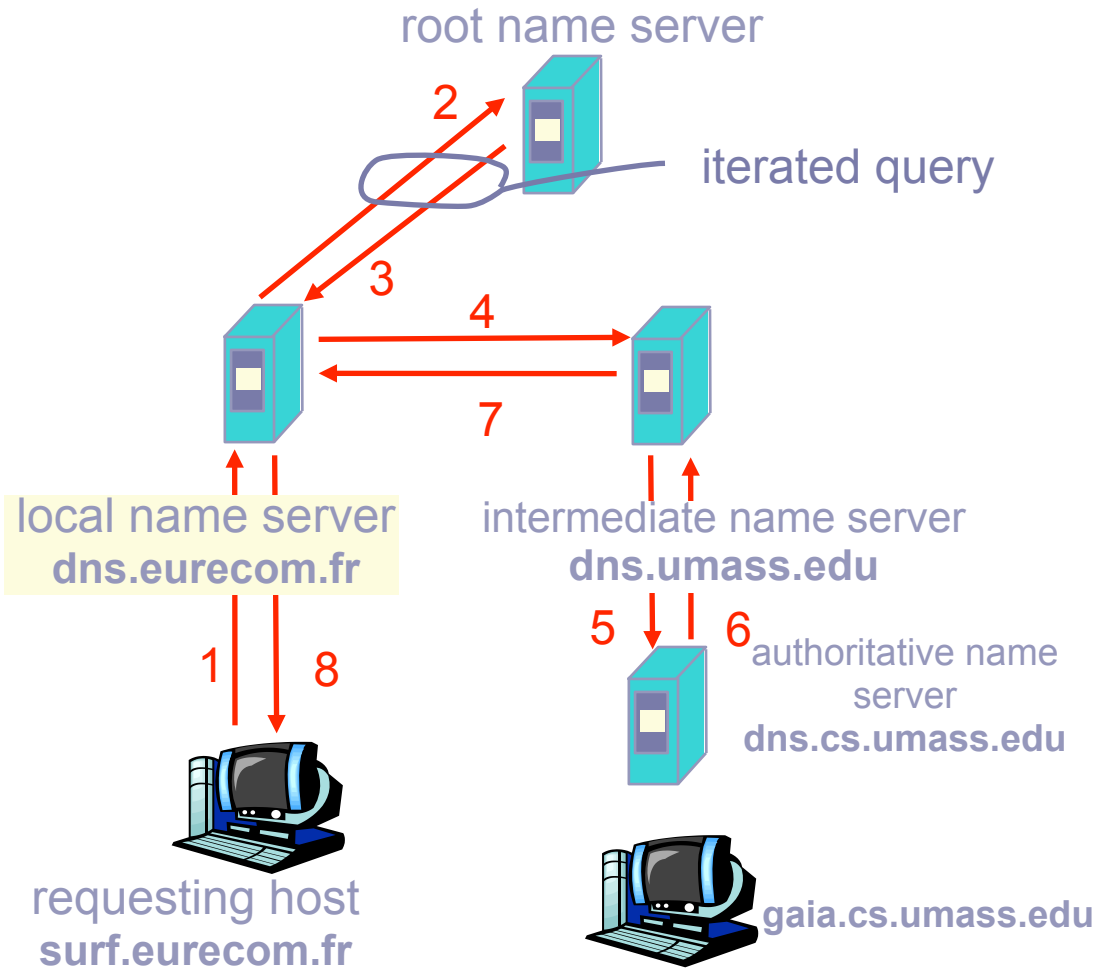
- Server goes out and searches for more info (recursive)
- Only returns final answer or "not found"

Iterative query:

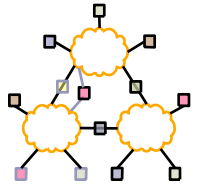
- Server responds with as much as it knows (iterative)
- "I don't know this name, but ask this server"

Workload impact on choice?

- Local server typically does recursive
- Root/distant server does iterative

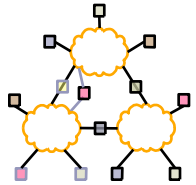


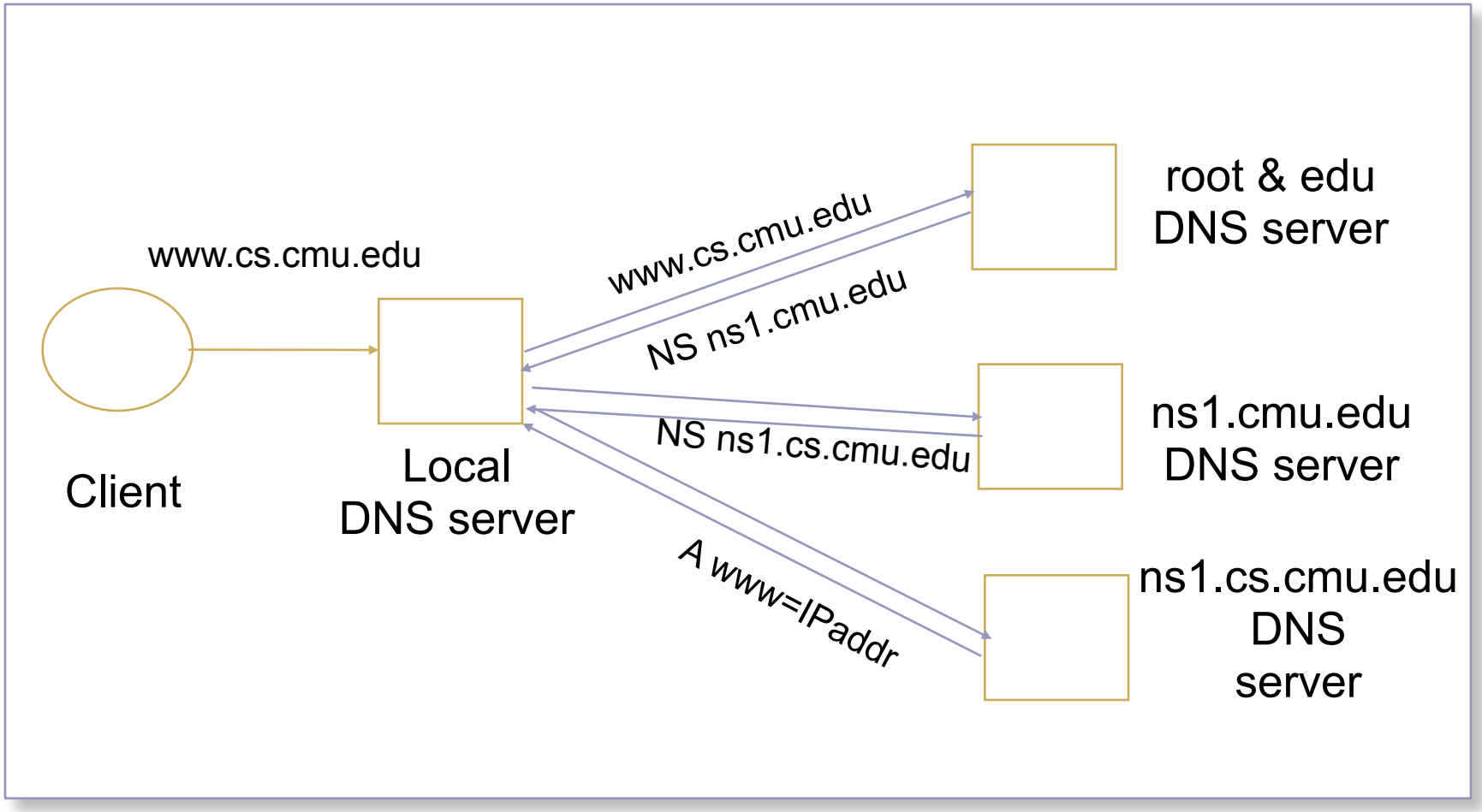
Workload and Caching



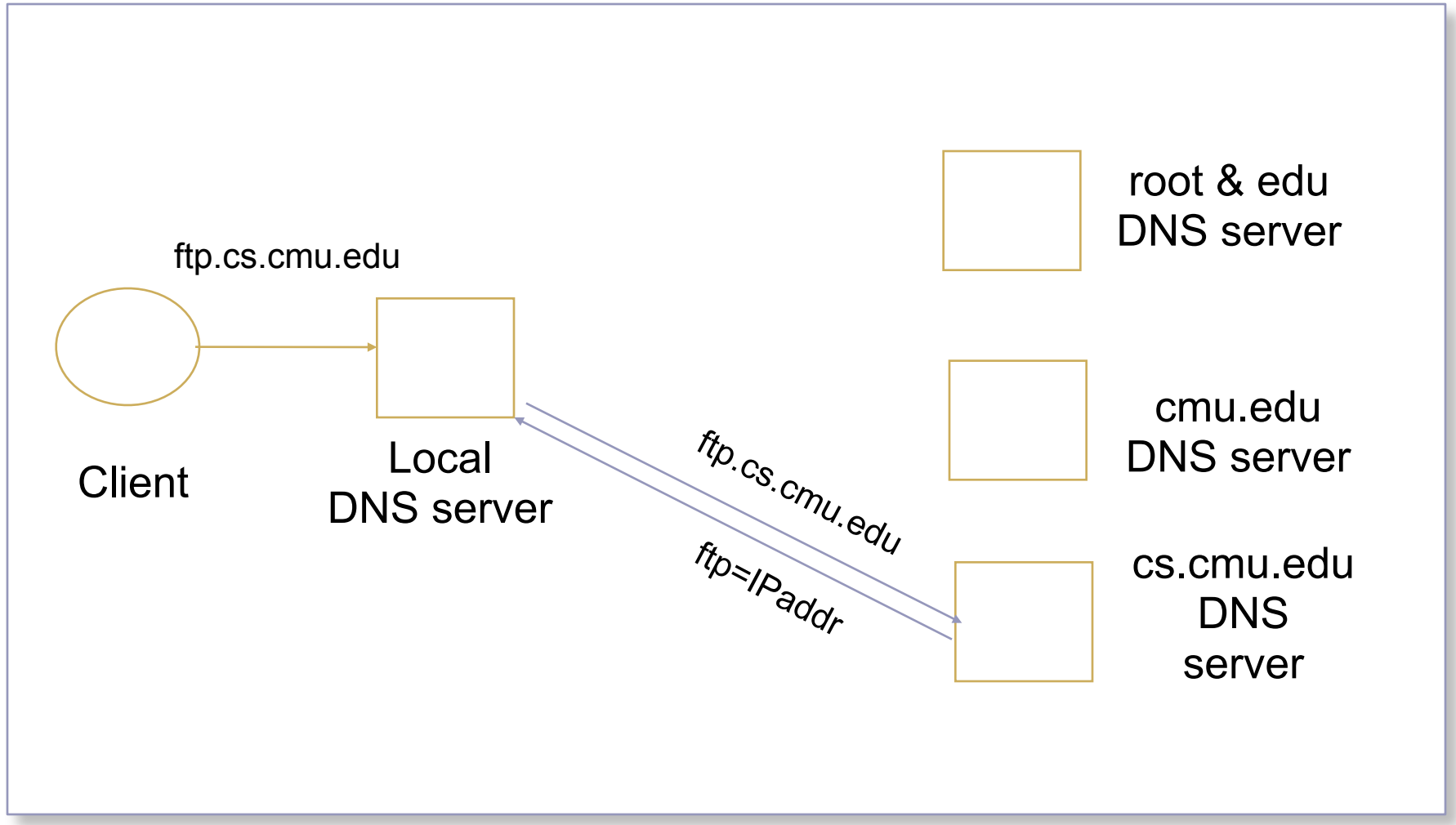
- What workload do you expect for different servers/names?
  - Why might this be a problem? How can we solve this problem?
- DNS responses are cached
  - Quick response for repeated translations
  - Other queries may reuse some parts of lookup
    - NS records for domains
- DNS negative queries are cached
  - Don’t have to repeat past mistakes
  - E.g. misspellings
- Cached data periodically times out
  - Lifetime (TTL) of data controlled by owner of data
  - TTL passed with every record

Typical Resolution

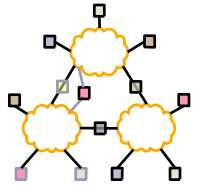




Subsequent Lookup Example

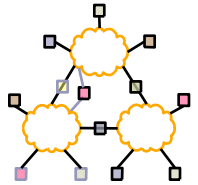


Reliability



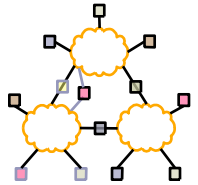
- DNS servers are replicated
  - Name service available if ≥ one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability → must implement this on top of UDP!
  - Why not just use TCP?
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don’t care which server responds

Prefetching



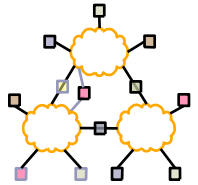
- Name servers can add additional data to any response
- Typically used for prefetching
  - CNAME/MX/NS typically point to another host name
  - Responses include address of host referred to in “additional section”

Root Zone



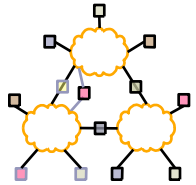
- Generic Top Level Domains (gTLD) = .com, .net, .org, etc…
- Country Code Top Level Domain (ccTLD) = .us, .ca, .fi, .uk, etc…
- Root server ({a-m}.root-servers.net) also used to cover gTLD domains
  - Load on root servers was growing quickly!
  - Moving .com, .net, .org off root servers was clearly necessary to reduce load → done Aug 2000

New gTLDs



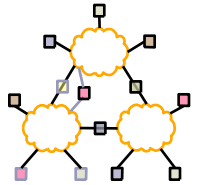
- .info → general info
- .biz → businesses
- .aero → air-transport industry
- .coop → business cooperatives
- .name → individuals
- .pro → accountants, lawyers, and physicians
- .museum → museums
- Only new one actives so far = .info, .biz, .name

New Registrars



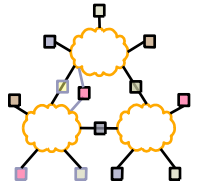
- Network Solutions (NSI) used to handle all registrations, root servers, etc…
  - Clearly not the democratic (Internet) way
  - Large number of registrars that can create new domains → However, NSI still handle root servers

Do you trust the TLD operators?



- Wildcard DNS record for all .com and .net domain names not yet registered by others
  - September 15 – October 4, 2003
  - February 2004: Verisign sues ICANN to restore SiteFinder
- Redirection for these domain names to Verisign web portal (SiteFinder)
- What services might this break?

Protecting the Root Nameservers



Attack On Internet Called Largest Ever

By David McGuire and Brian Krebs
washingtonpost.com Staff Writers
Tuesday, October 22, 2002; 5:40 PM

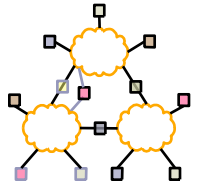
The heart of the Internet sustained its largest and most sophisticated attack ever, starting late Monday, according to officials at key online backbone organizations.

Around 5:00 p.m. EDT on Monday, a "distributed denial of service" (DDOS) attack struck the 13 "root servers" that provide the primary roadmap for almost all Internet communications. Despite the scale of the attack, which lasted about an hour, Internet users worldwide were largely unaffected, experts said.

Defense Mechanisms

- Redundancy: 13 root nameservers
- IP Anycast for root DNS servers {c,f,i,j,k}.root-servers.net
  - RFC 3258
  - Most *physical* nameservers lie outside of the US

DNS Hacks: Blackhole Lists



- First: Mail Abuse Prevention System (MAPS)
  - Paul Vixie, 1997
- Today: Spamhaus, spamcop, dnsrbl.org, etc.

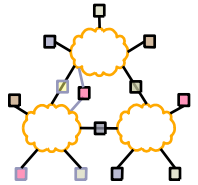
Different addresses refer to different reasons for blocking

% dig 91.53.195.211.bl.spamcop.net

;; ANSWER SECTION:
91.53.195.211.bl.spamcop.net. 2100 IN A 127.0.0.2

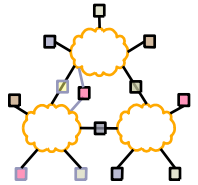
;; ANSWER SECTION:
91.53.195.211.bl.spamcop.net. 1799 IN TXT "Blocked - see http://www.spamcop.net/bl.shtml?211.195.53.91"

DNS Experience



- Hit rate for DNS = 80% → 1-(#DNS/#connections)
  - Most Internet traffic is Web
  - What does a typical page look like? → average of 4-5 imbedded objects → needs 4-5 transfers → accounts for 80% hit rate!
- 70% hit rate for NS records → i.e. don't go to root/ gTLD servers
  - NS TTLs are much longer than A TTLs
  - NS record caching is much more important to scalability

Next Lecture



- Naming and CDNs
- Required readings
  - Middleboxes No Longer Considered Harmful
  - Internet Indirection Infrastructure