Introduction to Computer Networks

CS640

Network Applications

https://pages.cs.wisc.edu/~mgliu/CS640/F22/

Ming Liu
mgliu@cs.wisc.edu

Today

Last lecture

- What are infrastructure services used for?

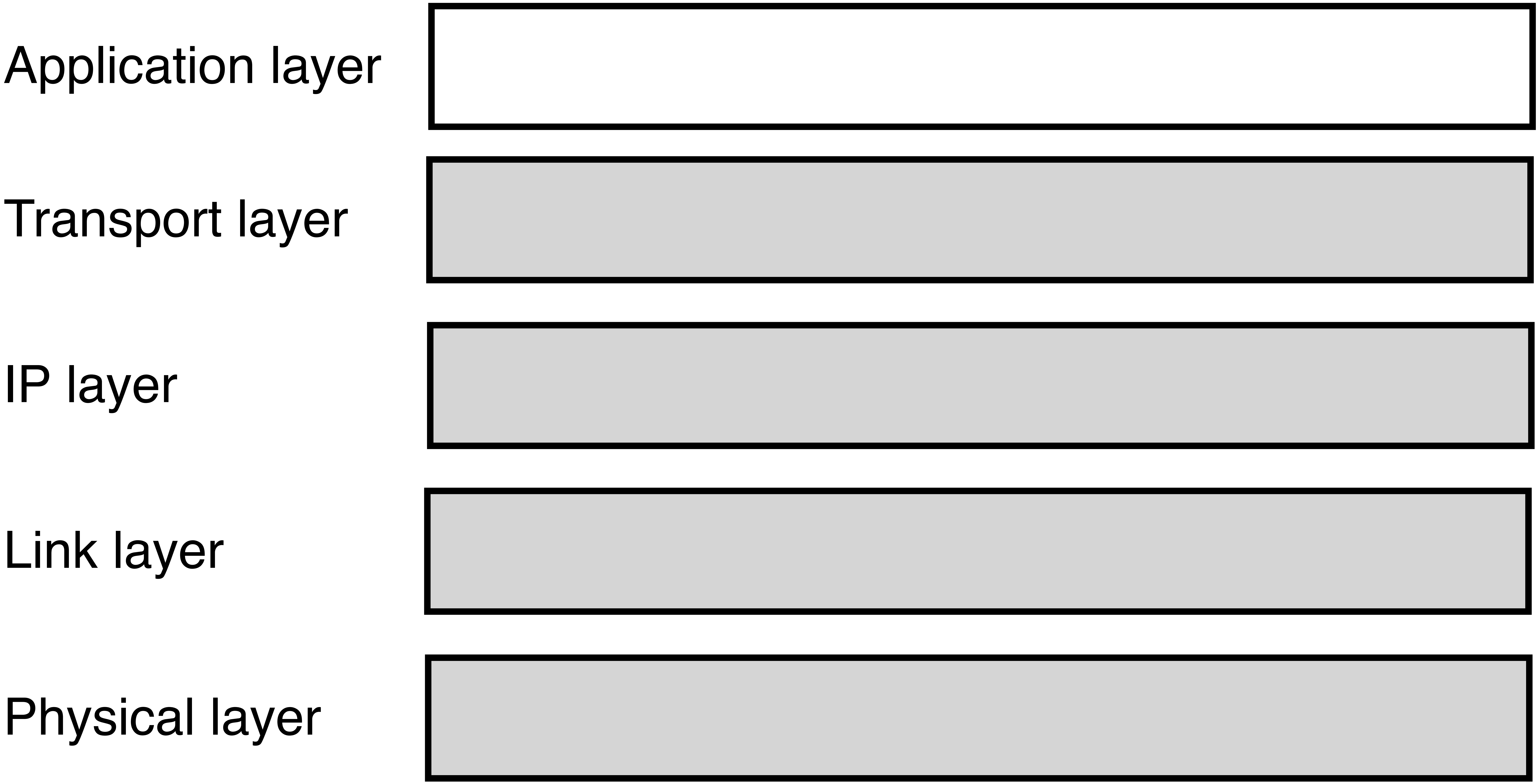
Today

- What are the learned lessons on building network applications?

Announcements

- Lab5 is due 12/14/2022, 11:59 PM
- Lab6 is due 12/19/2022, 11:59 PM
- Final exam: Dec 17, 2022 5:05 PM – 7:05 PM @Engineering Hall 1800

Application Layer in the TCP/IP Model



Application Layer in the TCP/IP Model

Application layer

Domain-specific?

Application Layer in the TCP/IP Model

Application layer

Common design questions:

#1: Underlying network assumption

- How large is the communication pipe?
- Is the pipe reliable or not?

#2: Coordination logic

- How to design the application header?
- How to minimize the maintained states and provide certain multi-tendency?

#3: Execution logic

- How much execution parallelism does the system preserve (or how to do flow-thread mapping)?
- What is the average per-packet computing density?

Q: What are the learned lessons on building network applications?

Q: What are the learned lessons on building network applications?

A: Three apps

- #1: Web/HTTP
- #2: P2P
- #3: Web Caching and CDNs

#1: Web/HTTP

A mechanism to organize and retrieve information

- Original goal of the web

Inspired by hypertext — one document links to another

- Hypertext Markup Language (HTML): define the basic content and layout of a web page
- Supplemented by Cascading Style Sheets (CSS), JavaScript, images, documents, Flash/Silverlight, and other files

Web

Uniform Resource Locator

- Specify the location of an object
- Perform DNS lookup to obtain the IP address of the web server to contact

Client and web server then communicate using HTTP

HyperText Transfer Protocol (HTTP)

Underlying network

- A reliable communication pipe with arbitrary bandwidth ==> runs atop TCP

Plain text messages in a request/response sequence

- lines terminated by \r\n

Coordination Logic: HTTP Request

#1: Start line

- Method to execute
  - GET: retrieve a document
  - HEAD: retrieve metadata about the document
  - POST: send data to the server
- URL: may exclude domain name (DN) and put this in an option
- HTTP/1.0 or HTTP/1.1 or HTTP/2.0

#2: Options/parameters

- User-Agent browser name/version, OS name/version
- Host: DN portion of URL

Coordination Logic: HTTP Request

#3: Blank line

#4: Data: only for methods like POST

Coordination Logic: HTTP Reply

#1: Start line

- HTTP/1.0 or HTTP/1.1 or HTTP/2.0
- Status
  - 200 OK
  - 404 Not found
  - 403 Forbidden
  - 301 Moved permanently

Coordination Logic: HTTP Reply

#2: Options/parameters

- Content-Length
- Content-Type
- Server – server name/version
- Cache-Control – how long object can be cached
- Last-Modified

#3: Blank line

#4: Data

Example: Fetching a Web Page (www.wisc.edu)

#1: DNS lookup

#2: Establish TCP connection

#3: Send HTTP request

#4: Receive HTTP reply

#5: Close TCP connection

#6: Parse HTML

→ **Other objects (e.g., image, etc)**

Example: Fetching a Web Page (www.wisc.edu)

#7: Establish TCP connection

#8: Send HTTP request for image

#9: Receive HTTP reply for image

#10: Close TCP connection

……

Example: Fetching a Web Page (www.wisc.edu)

#N: Request other objects in a page

#N+1: Perform more DNS lookups if objects (e.g., ads) are in different domain (e.g., CDN)

#N+2: Render page while other objects are being fetched

Example: Fetching a Web Page (www.wisc.edu)

#N: Request other objects in a page

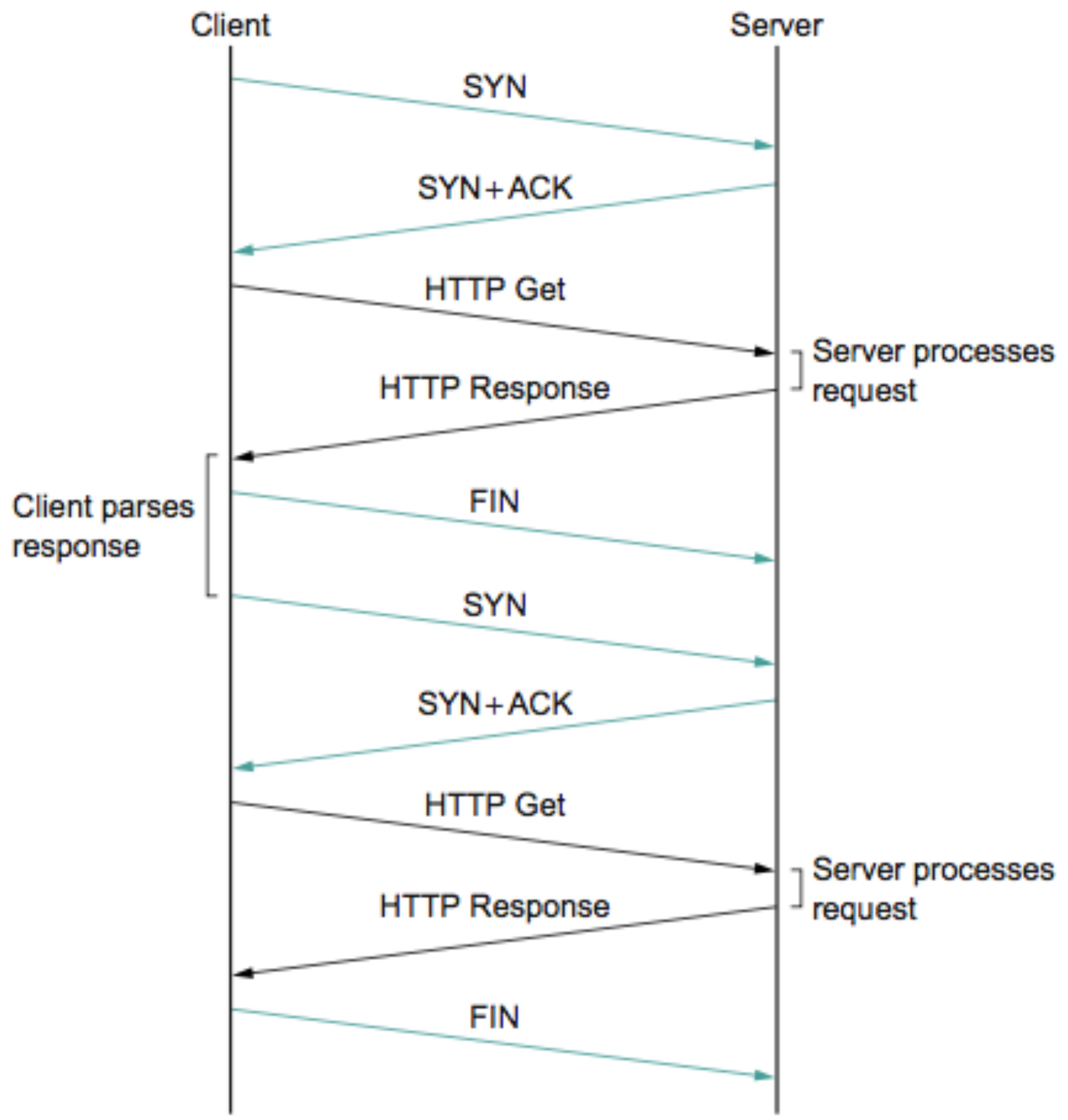
Early HTTP used to this (HTTP 1.0)

#N+2: Render page while other objects are being fetched

Execution Logic: inefficiencies in HTTP 1.0

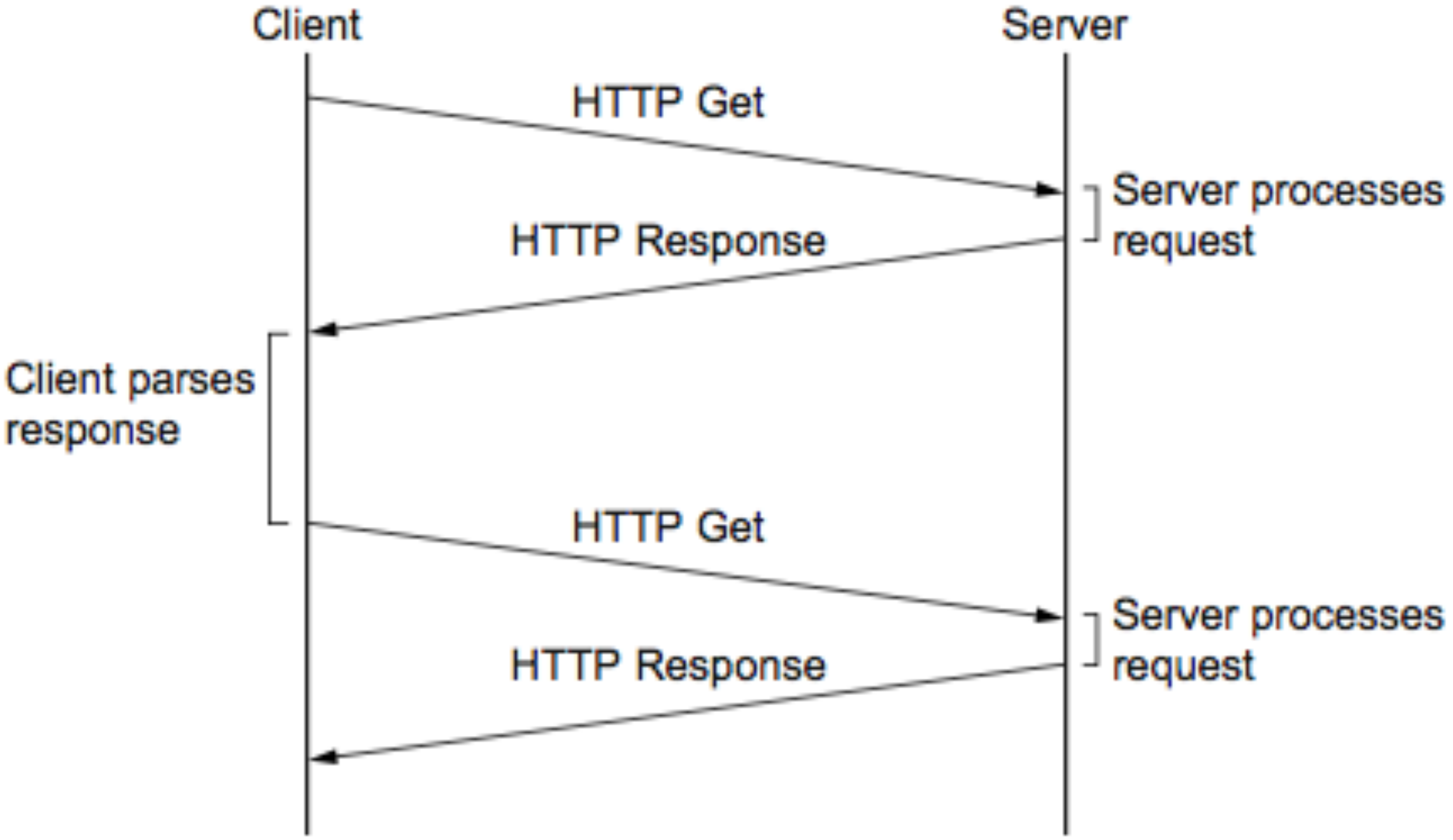
Problem: Using a separate TCP connection for each object in a web page has a lot of overheads for connection setup and teardown

- For each object: 2 RTTs for connection setup + at least 1 RTT for feting data



Execution Logic: inefficiencies in HTTP 1.0

Solution: HTTP 1.1. introduced persistent connections



HTTP Persistent Connections

Key idea: exchange multiple request/response messages over the same TCP connection

Only need to establish one connection to each server providing content for a page

- If the content is coming from multiple servers (e.g., the main page and ads come from 2 different domains), you still need +1 connection

HTTP Persistent Connections Discussion

Also benefits throughput

- For each connection, the congestion window starts at 1 and is increasing exponentially using a slow start
- Using one connection means the initial slow start only occurs once
  - Still invoke slow start later if a timeout occurs due to loss, but ideally, losses are handled through fast retransmit/fast recovery where slow start is not invoked

HTTP Persistent Connections Discussion

Also benefits throughput

- For each connection, the congestion window starts at 1 and is increasing exponentially using a slow start
- Using one connection means the initial slow start only occurs once
  - Still invoke slow start later if a timeout occurs due to loss, but ideally, losses are handled through fast retransmit/fast recovery where slow start is not invoked

Challenge: how long should a connection stay open?

- Overhead at a server to maintain a connection for 100s clients
- Throughput benefits far outweigh this overhead

#2: Peer-to-Peer (P2P)

A peer-to-peer (P2P) network allows a community of users to pool their resources

- Storage
- CPU
- …

P2P networks are decentralized, self-organizing

Why do we care about these networks?

- It is challenging to achieve decentralization and scalability at the same time

BitTorrent

BitTorrent is a peer-to-peer file-sharing protocol based on replicating the file, or rather, replicating segments of the file, which are called pieces or chunks

Any particular piece can usually be downloaded from multiple peers, even if only one peer has the entire file

BitTorrent

The benefit of BitTorrent’s replication is avoiding the bottleneck of having only one source for a file

- This is particularly useful when you consider that any given computer has a limited speed at which it can serve files over its uplink to the Internet

Arbitrary underlying network

Execution Logic: Replication

The beauty of BitTorrent is that replication is a natural side-effect of the downloading process:

- As soon as a peer downloads a particular piece, it becomes another source

The more peers downloading pieces of the file, the more piece replication occurs

- Distributing the load proportionally
- Receiving more aggregation bandwidth to share the file with others

Execution Logic: Replication

Pieces are downloaded in random order to avoid a situation where peers find themselves lacking the same set of pieces

Coordination Logic: Swarms

Each file is shared via its own independent BitTorrent network, called a swarm

The lifecycle of a typical swarm is as follows:

- The swarm starts as a singleton peer with a complete copy of the file
- A node wants to download the file and join the swam, becoming its second member
- A node begins downloading pieces of the file from the original peer
- In doing so, it becomes another source for the pieces it has download, even it has not yet downloaded the entire file

Coordination Logic: a New Node P

#1: P joins the swarm

#2: Tracker replies to P with a partial list of peers

#3: P establishes TCP connections with a random subset

#4: P exchanges swarm ID to make sure peers are in the same swarm as itself

Coordination Logic: a New Node P

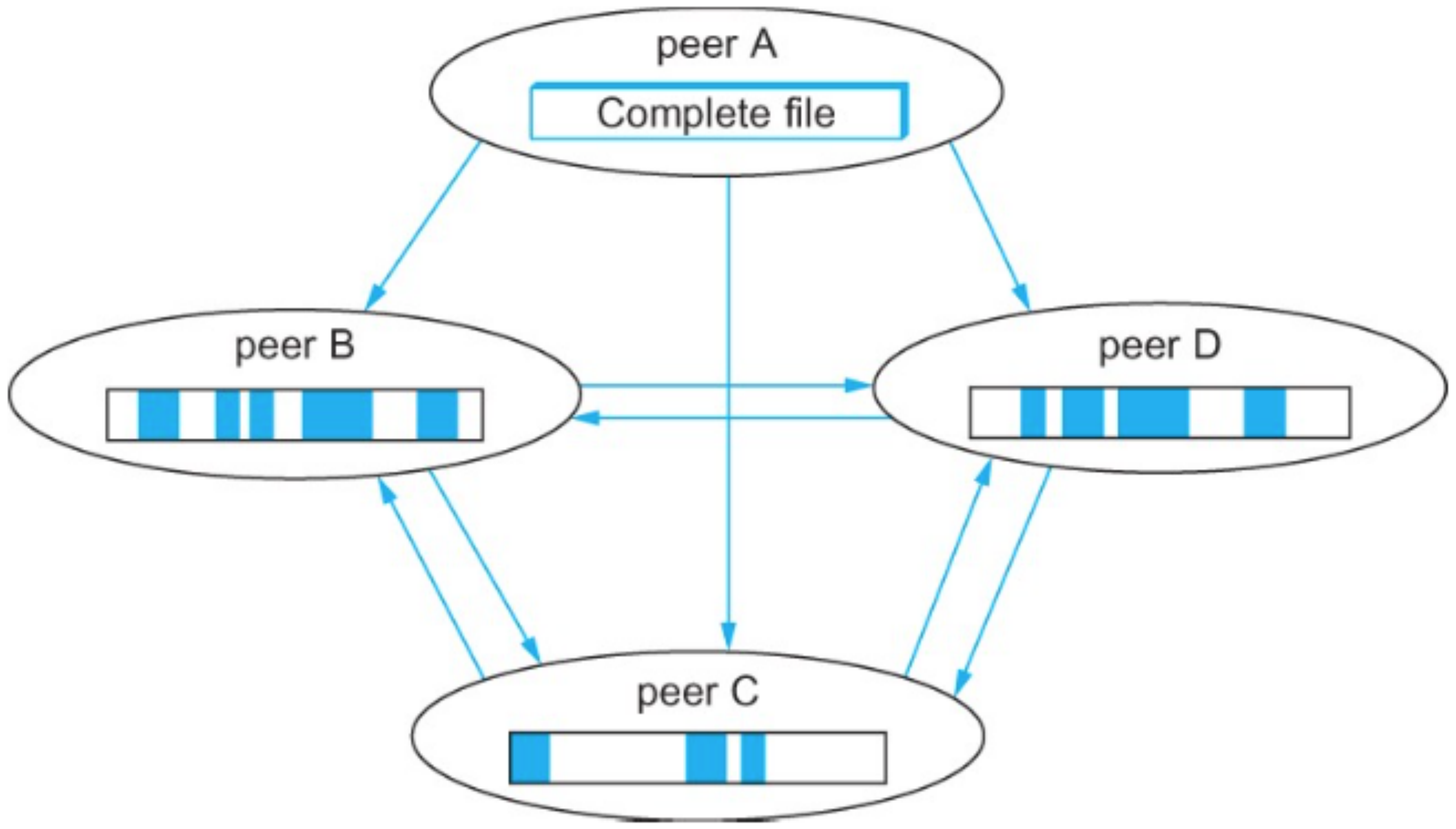
#5: If these checks pass, each peer begins by sending a bitmap of blocks it has. This is used by P to decide what block to get

#6: When download of a block is finished, exchange bitmaps with all connected peers

#7: Peers download blocks in random order to avoid getting blocked on the same piece

Illustration

Peers in a BitTorrent swarm download from other peers that may not yet have the complete file



Where are the communication bottlenecks?

First mile: client to its ISPs

Last mile: server to its ISP

Server: compute/memory limitations

ISP interconnections or peerings: congestion inside the network

Where are the communication bottlenecks?

First mile: client to its ISPs

Last mile: server to its ISP

Server: compute/memory limitations

Caching at various locations to overcome the latter three bottlenecks (first one can't be helped!)

Proxy Caches

Cache "close" to the client

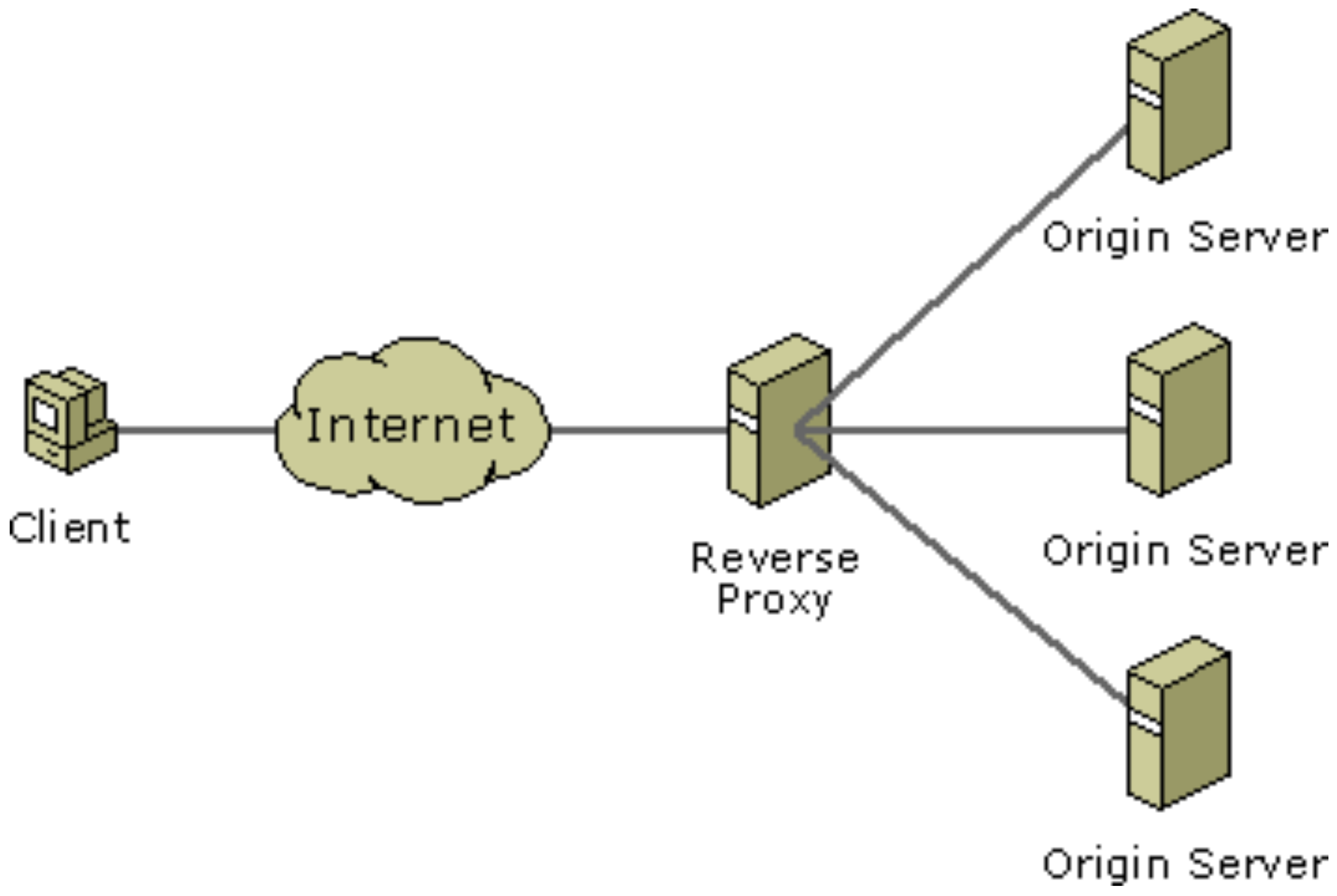
- Under administrative control of client-side AS

Explicit proxy

- Requires configuring browser

Implicit proxy

- Service provider deploys an on-path proxy
- It intercepts and handles web requests



Limitations of Web Caching

Much content is not cacheable => Caching policy

- Dynamic data: stock prices, scores, webcams
- Cookies: results may depend on passed data
- SSL: encrypted data is not cacheable
- Analytics: owner wants to measure hits

Stale data => Eviction policy

- Or, overhead of refreshing the cached data

Content Distribution Network (CDN)

Proactive content replication

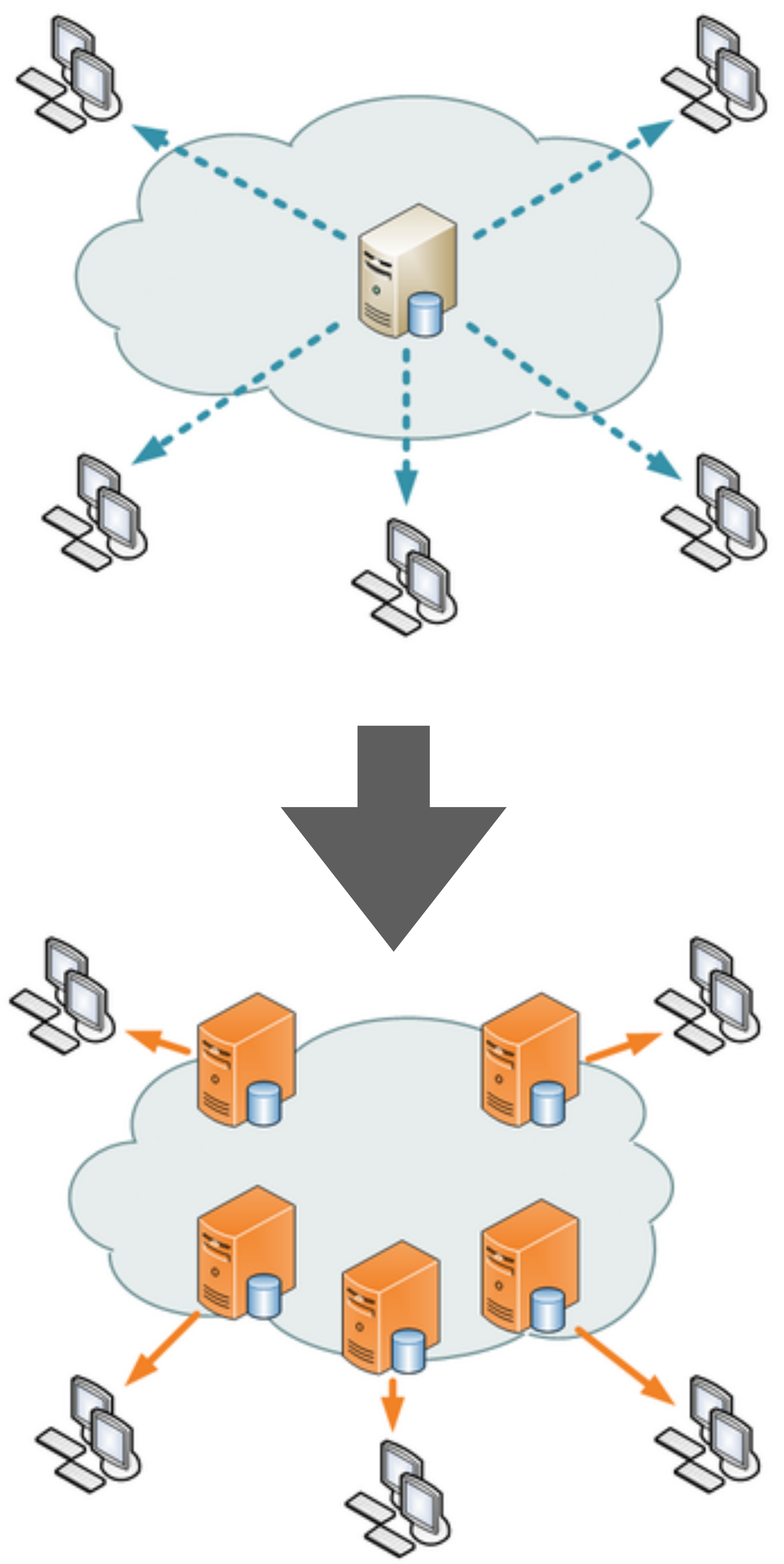
- Content provider (e.g., CNN) contracts with a CDN

CDN replicates the content

- On many servers, which spread throughput the Internet

Updating the replicas

- Updates pushed to replicas when the content changes



Server Selection Policy

Live server

- For availability

Lowest load

- To balance load across the servers

Closet

- Nearest geographically, or in round-trip time

Best performance

- Throughput, latency, …

Cheapest bandwidth, electricity, …

Server Selection Policy

Live server

- For availability

Lowest load

- To balance load across the servers

Closet

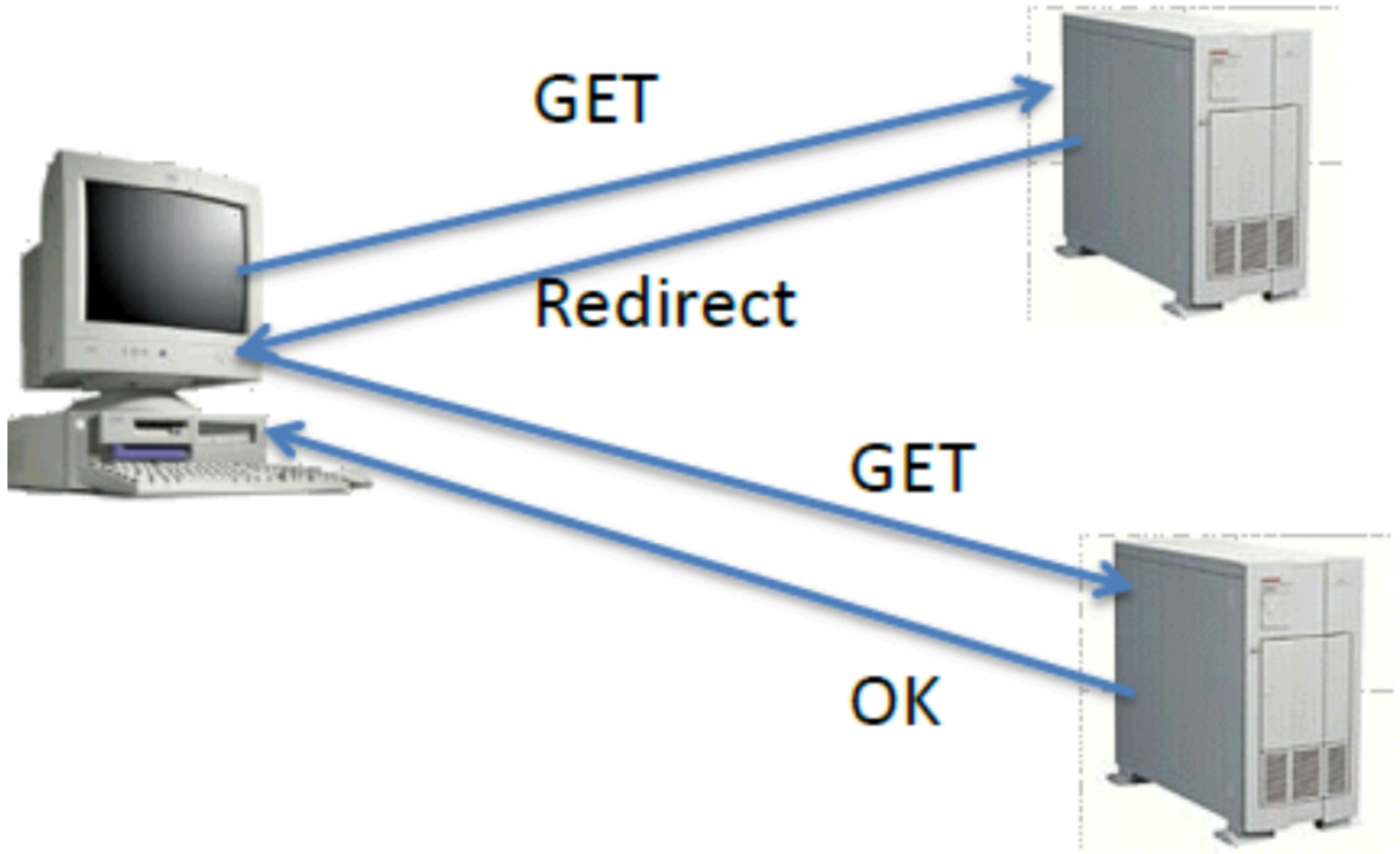
- Nearest geographically, or in round-trip time

Requires continuous monitoring of liveness, load, and performance

Server Selection Mechanism #1

Application

- HTTP redirection



Advantages

- Fine-grain control
- Selection based on client IP address

Disadvantages

- Extra round-trips for TCP connecting to the server
- Overhead on the server

Server Selection Mechanism #2

Naming

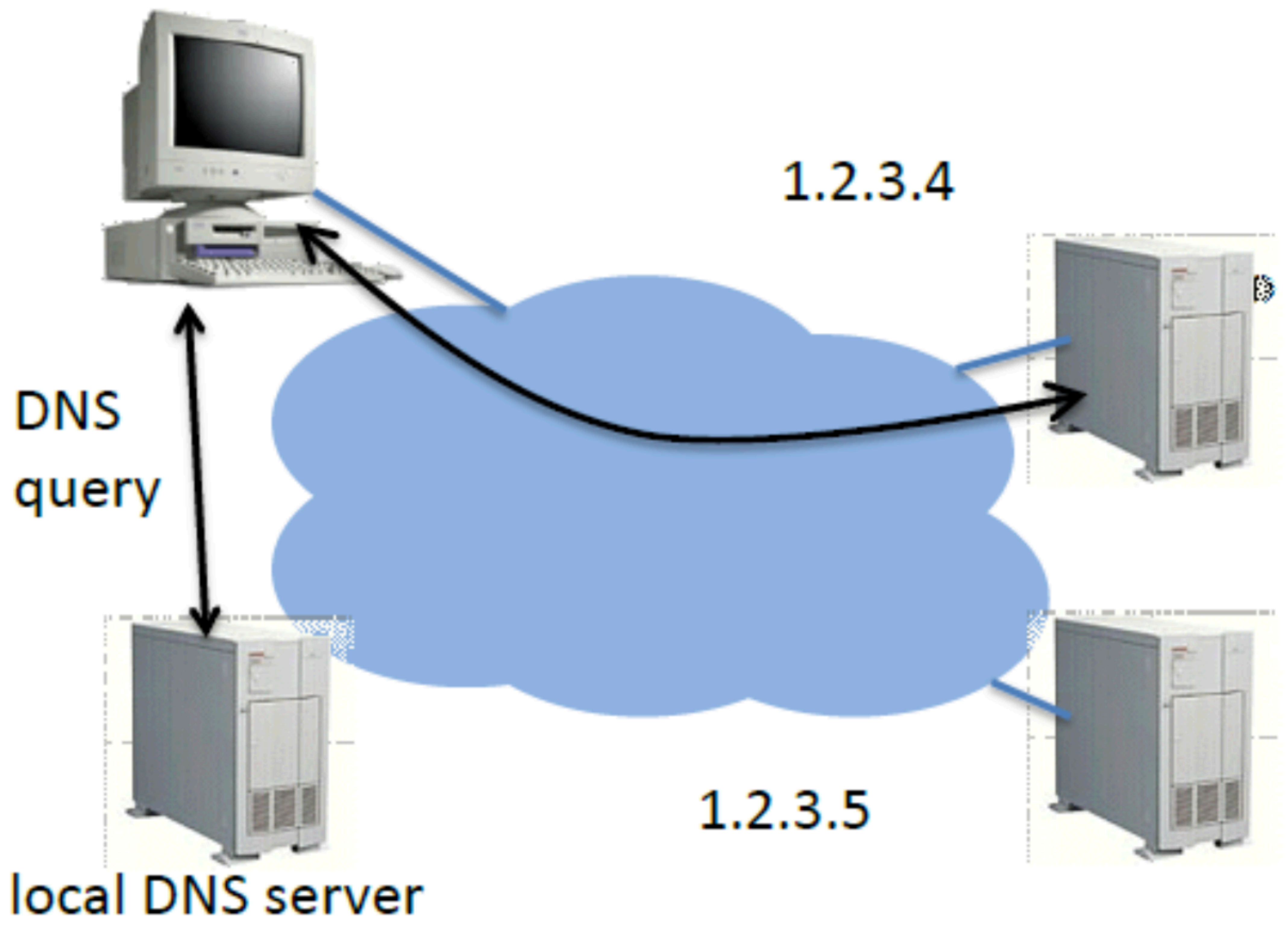
- DNS-based server selection

Advantages

- Avoid TCP set-up delay
- DNS caching reduces overhead
- Relatively fine control

Disadvantages

- Based on IP address of local DNS server
- “Hidden load” effect
- DNS TTL limits adaption



Q: How does Web Caching/CDN address the three design questions?

A: Web Caching/CDN:

- #1: Underlying network => Any
- #2: Coordination logic => Interception
- #3: Execution logic => Caching

Think carefully before building networking applications!

Terminology

1. Host
2. NIC
3. Multi-port I/O bridge
4. Protocol
5. RTT
6. Packet
7. Header
8. Payload
9. BDP
10. Baud rate
11. Frame/Framing
12. Parity bit
13. Checksum
14. Ethernet
15. MAC
16. (L2) Switch
17. Broadcast
18. Acknowledgement
19. Timeout
20. Datagram
21. TTL
22. MTU
23. Best effort
24. (L3) Router
25. Subnet mask
26. CIDR
27. Converge
28. Count-to-infinity
29. Line card
30. Network processor
31. Gateway
32. Private network
33. IPv6
34. Multicast
35. IGMP
36. SDN
37. (Transport) port
38. Pseudo header
39. SYN/ACK
40. Incarnation
41. Flow
42. SYN flood
43. TCP Segment
44. Window
45. Advertised Window
46. Effective Window
47. TCP Reno
48. Duplicated ACK
49. Congestion Window
50. Congestion Threshold
51. Selective Acknowledgment
52. Active Queue Management (AQM)
53. URL
54. HTML
55. Peer-to-peer (P2)
56. Swarm
57. CDN

Principle

1. Layering
2. Minimal States
3. Hierarchy
4. Mechanism/policy separation

Technique

1. NRZ Encoding
2. NRZI Encoding
3. Manchester Encoding
4. 4B/5B Encoding
5. Byte Stuffing
6. Byte Counting
7. Bit Stuffing
8. 2-D Parity
9. CRC
10. MAC Learning
11. Store-and-Forward
12. Cut-through
13. Spanning Tree
14. CSMA/CD
15. Stop-and-Wait
16. Sliding Window
17. Fragmentation and Reassembly
18. Path MTU discovery
19. DHCP
20. Subnetting
21. Supernetting
22. Longest prefix match
23. Distance vector routing (RIP)
24. Link state routing (OSPF)
25. Boarder gateway protocol (BGP)
26. Network address translation (NAT)
27. User Datagram Protocol (UDP)
28. Transmission Control Protocol (TCP)
29. Three-way Handshake
30. TCP state transition
31. EWMA
32. Sliding window

Technique

33. Flow control
34. AIMD
35. Slow start
36. Fast retransmit
37. Fast recovery
38. Nagle's algorithm
39. Karn/Partridge algorithm
40. TCP Vegas
41. Bit-by-bit Round Robin
42. Fair Queueing (FQ)
43. Random Early Detection (RED)
44. Explicit Congestion Notification (ECN)
45. Domain Name System (DNS)
46. Simple Network Management Protocol (SNMP)
47. HyperText Transfer Protocol (HTTP)
48. Persistent Connection
49. BitTorrent

Summary

Today's takeaways

#1: When developing network applications, there are three design questions: underlying network assumption, coordination logic, and execution logic

- Web
- P2P
- CDN

Next lecture

- Network Security