第3周 课堂教学-应用层（下）

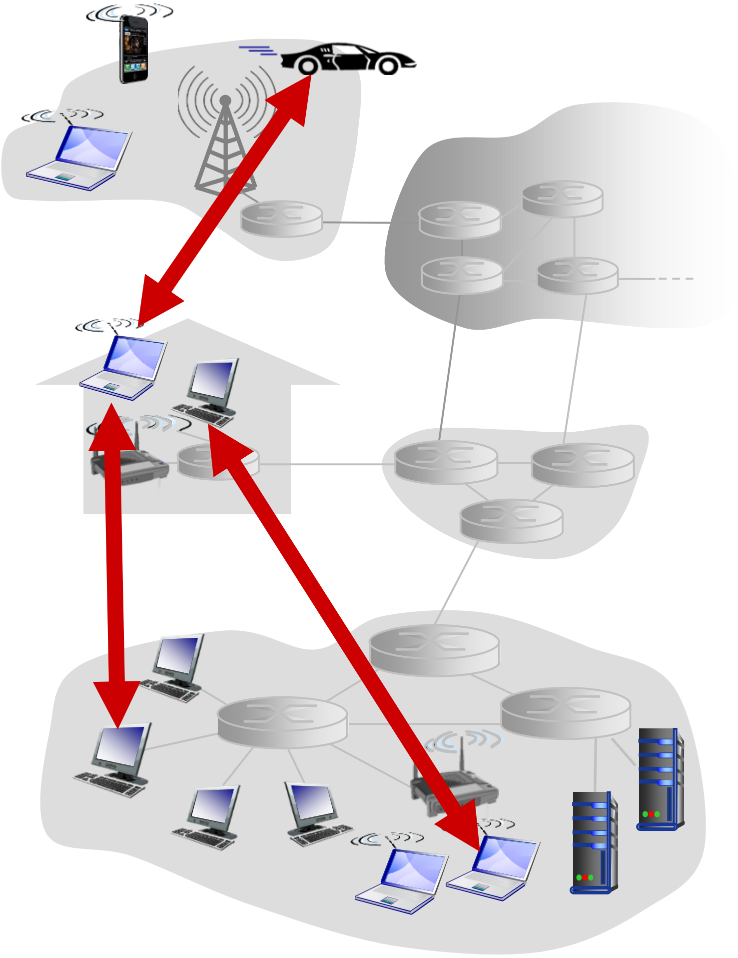
❖ 東广就狭：

- 2.6 P2P应用
- 2.7 Socket编程

第3周 课堂教学-应用层（下）

❖ 质疑辨惑：

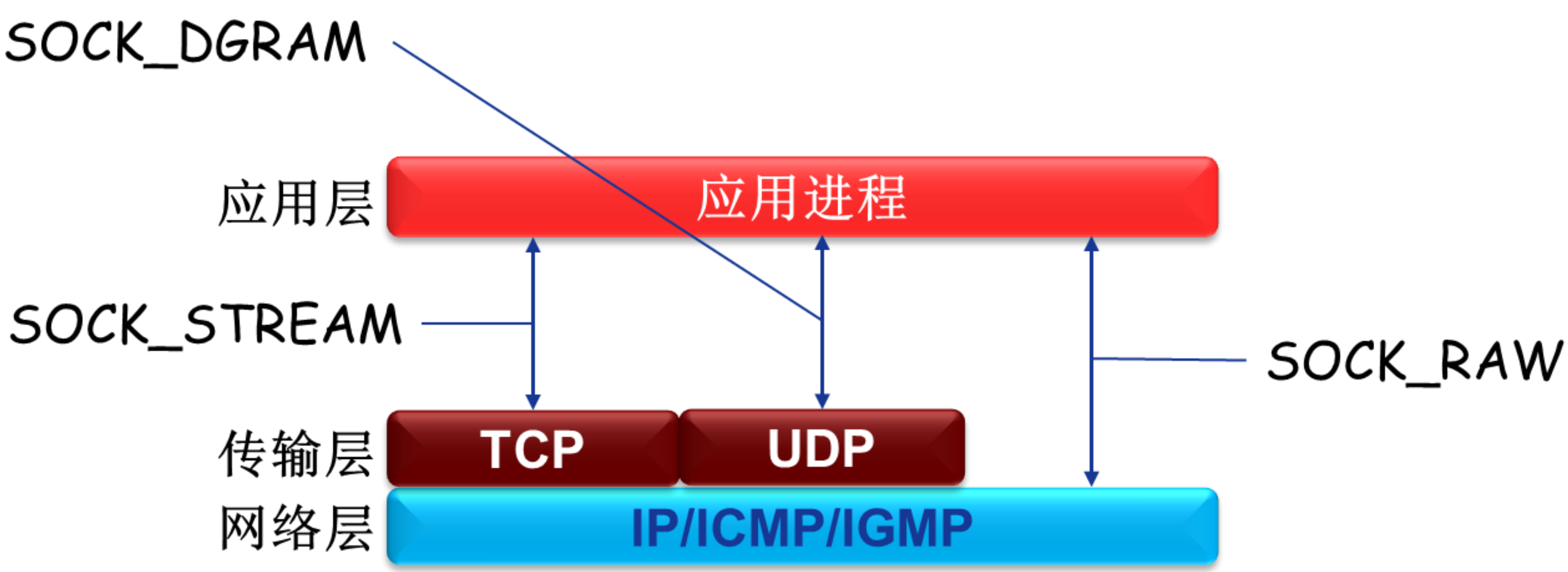
- 1. P2P网络应用的通信过程实质是什么？可能的P2P应用程序结构有哪些？P2P网络应用是否适合基于DNS实现Peer寻址？



第3周 课堂教学-应用层（下）

❖ 质疑辨惑：

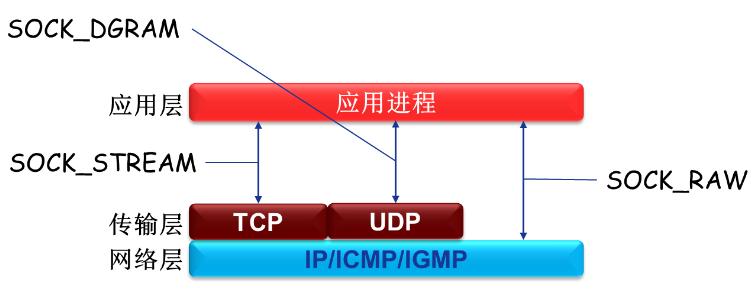
- 1. P2P网络应用的通信过程实质是什么？可能的P2P应用程序结构有哪些？P2P网络应用是否适合基于DNS实现Peer寻址？
- 2. 为什么网络应用进程间通信时，不利用进程ID来标识进程？Socket可以提供哪几类编程接口？如何理解端口号？



第3周 课堂教学-应用层（下）

❖ 质疑辨惑：

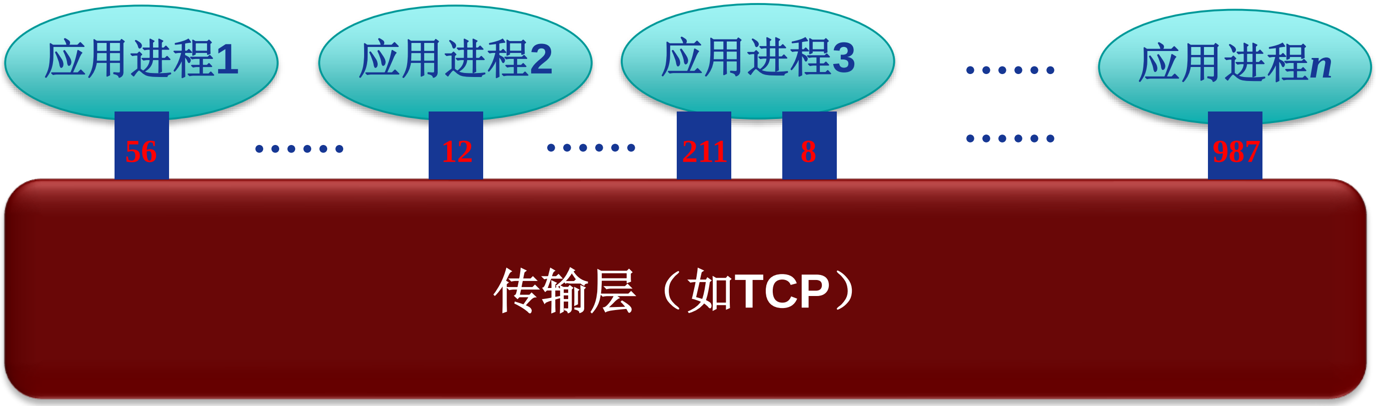
- 1. P2P网络应用的通信过程实质是什么？可能的P2P应用程序结构有哪些？P2P网络应用是否适合基于DNS实现Peer寻址？
- 2. 为什么网络应用进程间通信时，不利用进程ID来标识进程？Socket可以提供哪几类编程接口？如何理解端口号？



第3周 课堂教学-应用层（下）

❖ 质疑辨惑：

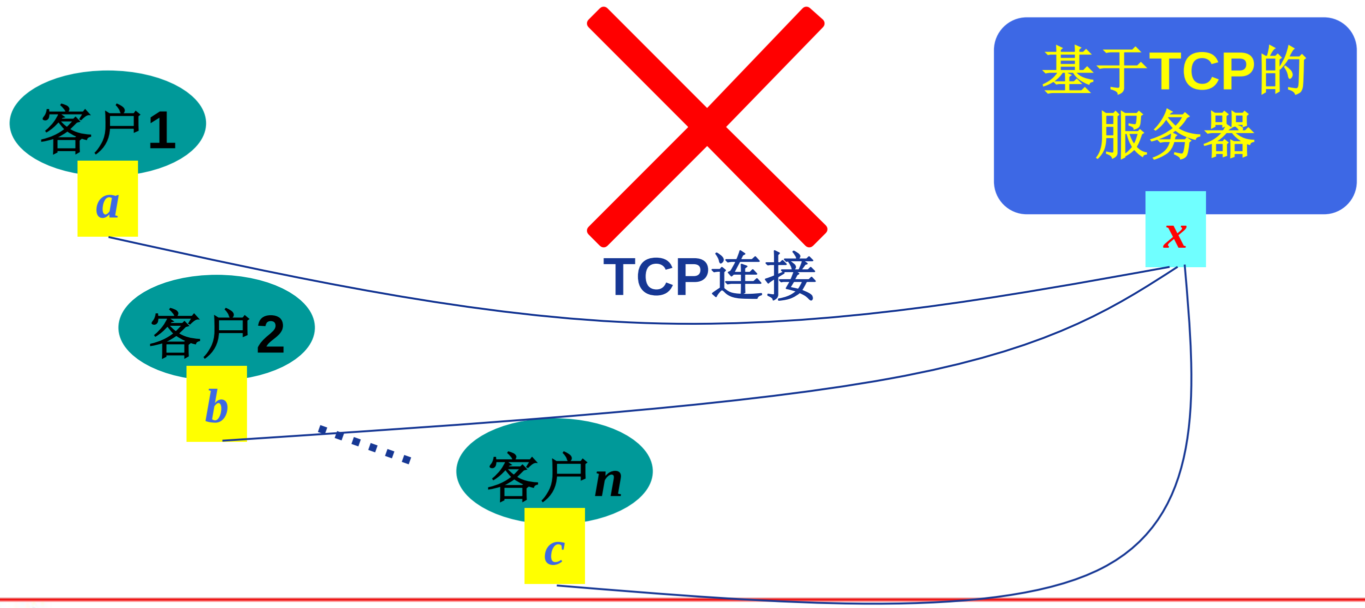
- 1. P2P网络应用的通信过程实质是什么？可能的P2P应用程序结构有哪些？P2P网络应用是否适合基于DNS实现Peer寻址？
- 2. 为什么网络应用进程间通信时，不利用进程ID来标识进程？Socket可以提供哪几类编程接口？如何理解端口号？



第3周 课堂教学-应用层（下）

❖ 质疑辨惑：

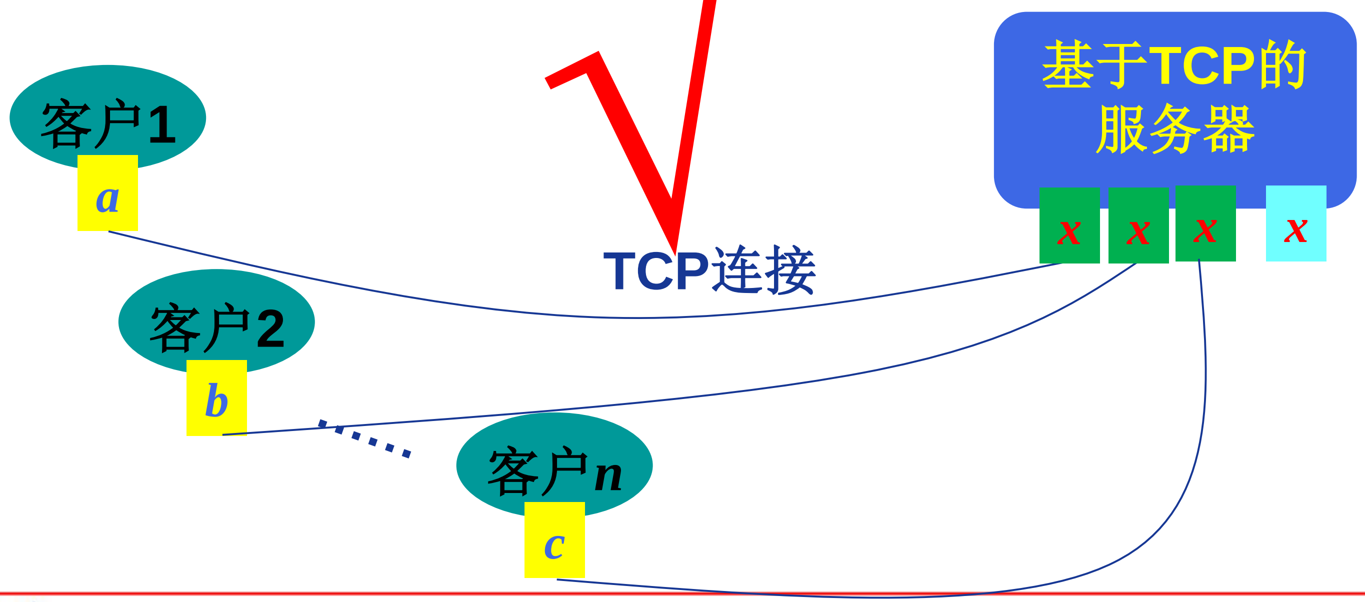
- 1. P2P网络应用的通信过程实质是什么？可能的P2P应用程序结构有哪些？P2P网络应用是否适合基于DNS实现Peer寻址？
- 2. 为什么网络应用进程间通信时，不利用进程ID来标识进程？Socket可以提供哪几类编程接口？如何理解端口号？
- 3.基于TCP的服务器如何实现同时与多客户通信？为什么？



第3周 课堂教学-应用层（下）

❖ 质疑辨惑：

- 1. P2P网络应用的通信过程实质是什么？可能的P2P应用程序结构有哪些？P2P网络应用是否适合基于DNS实现Peer寻址？
- 2. 为什么网络应用进程间通信时，不利用进程ID来标识进程？Socket可以提供哪几类编程接口？如何理解端口号？
- 3.基于TCP的服务器如何实现同时与多客户通信？为什么？



第3周 课堂教学-应用层（下）

❖ 质疑辨惑：

- 1.P2P网络应用DNS吗？为什么？
- 2.如何理解Socket编程接口？
- 3.基于TCP的服务器如何实现同时与多客户通信？为什么？
- 4.如何优化Web应用的响应时间？
  - 优化HTTP
  - Web缓存
  - CDN

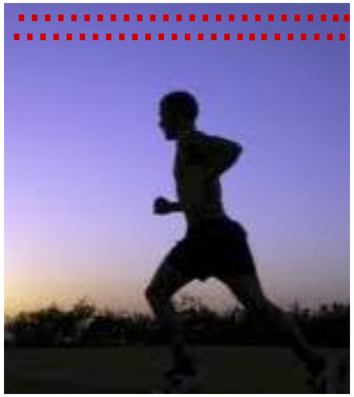
Video Streaming and CDNs: context

- video traffic: major consumer of Internet bandwidth
  - Netflix, YouTube: 37%, 16% of downstream residential ISP traffic
  - ~1B YouTube users, ~75M Netflix users
- challenge: scale - how to reach ~1B users?
  - single mega-video server won't work (why?)
- challenge: heterogeneity
  - different users have different capabilities (e.g., wired versus mobile; bandwidth rich versus bandwidth poor)
- *solution:* distributed, application-level infrastructure

Multimedia: video

- video: sequence of images displayed at constant rate
  - e.g., 24 images/sec
- digital image: array of pixels
  - each pixel represented by bits
- coding: use redundancy *within* and *between* images to decrease # bits used to encode image
  - spatial (within image)
  - temporal (from one image to next)

spatial coding example: **instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)**



frame *i*

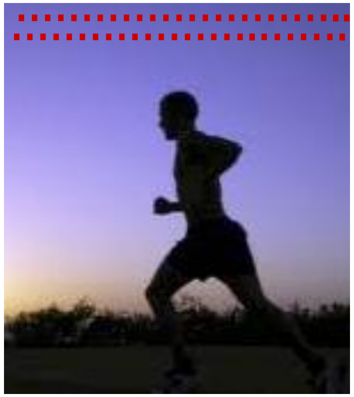
temporal coding example: **instead of sending complete frame at i+1, send only differences from frame i**

frame *i+1*

Multimedia: video

- **CBR: (constant bit rate):** video encoding rate fixed
- **VBR: (variable bit rate):** video encoding rate changes as amount of spatial, temporal coding changes
- **examples:**
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (often used in Internet, < 1 Mbps)

spatial coding example: instead of sending *N* values of same color (all purple), send only two values: color value (*purple*) *and number of repeated values* (N)



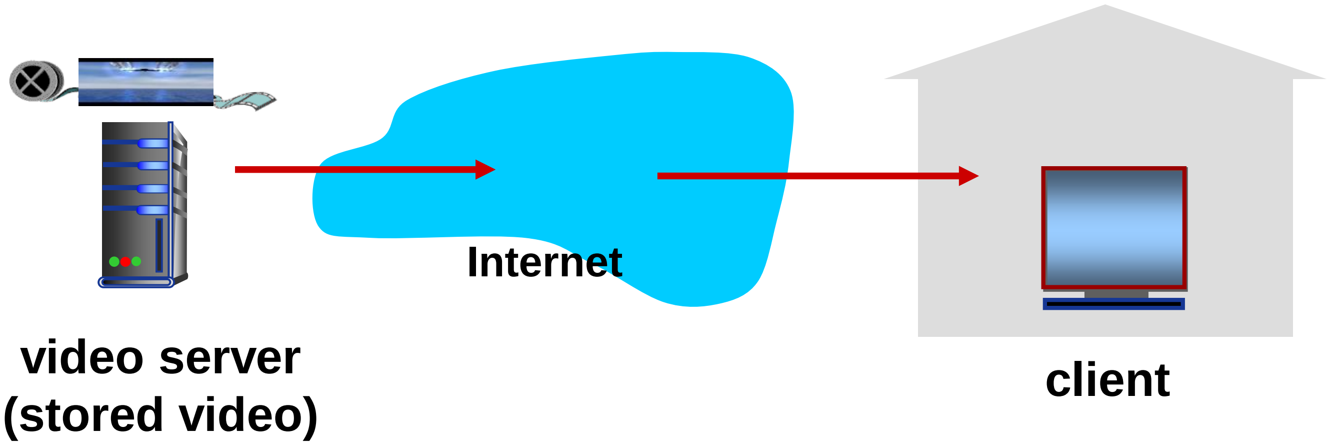
frame *i*

temporal coding example: instead of sending complete frame at i+1, send only differences from frame i

frame *i+1*

Streaming stored video:

simple scenario:



Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *H*TTP
- *server:*
  - divides video file into multiple chunks
  - each chunk stored, encoded at different rates
  - *manifest file:* provides URLs for different chunks
- *client:*
  - periodically measures server-to-client bandwidth
  - consulting manifest, requests one chunk at a time
    - chooses maximum coding rate sustainable given current bandwidth
    - can choose different coding rates at different points in time (depending on available bandwidth at time)

Streaming multimedia: DASH

- *DASH: D*ynamic, *A*daptive *S*treaming over *HTTP*
- *"intelligence"* at client: client determines
  - *when* to request chunk (so that buffer starvation, or overflow does not occur)
  - *what encoding rate* to request (higher quality when more bandwidth available)
  - *where* to request chunk (can request from URL server that is "close" to client or has high available bandwidth)

Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?

- *option 1:* single, large "mega-server"
  - single point of failure
  - point of network congestion
  - long path to distant clients
  - multiple copies of video sent over outgoing link

….quite simply: this solution *doesn't scale*

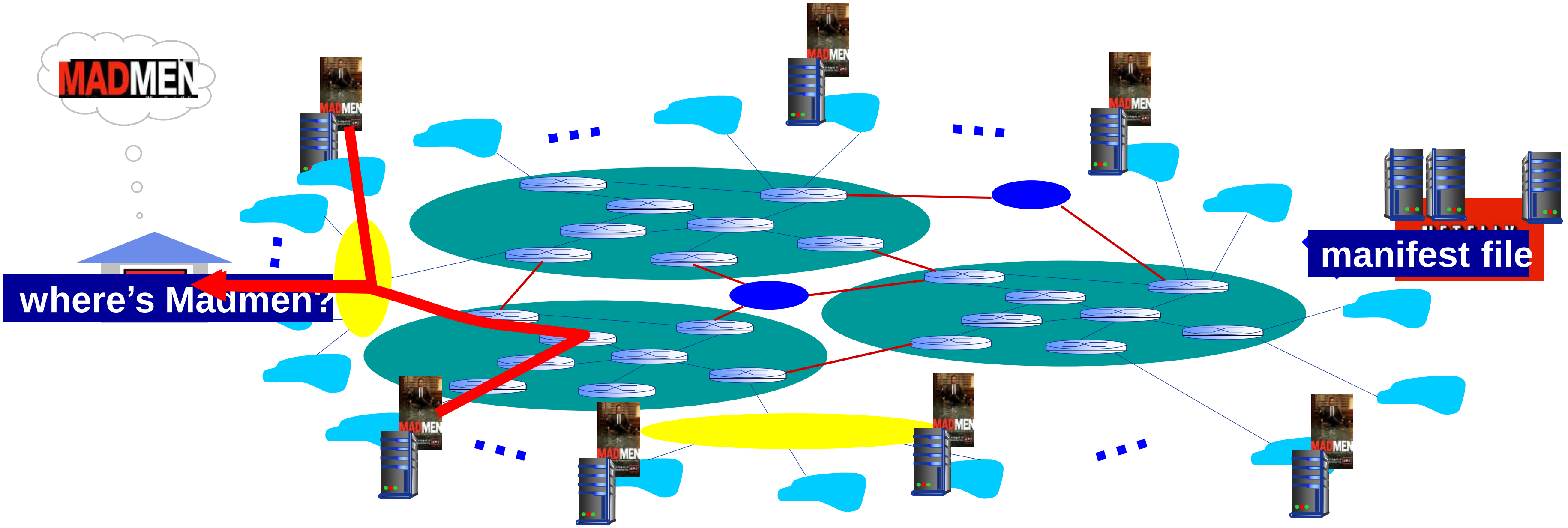
Content distribution networks

- *challenge:* how to stream content (selected from millions of videos) to hundreds of thousands of simultaneous users?

- *option 2:* store/serve multiple copies of videos at multiple geographically distributed sites *(CDN)*
  - *enter deep:* push CDN servers deep into many access networks
    - close to users
    - used by Akamai, 1700 locations
  - *bring home:* smaller number (10's) of larger clusters in POPs near (but not within) access networks
    - used by Limelight

Content Distribution Networks (CDNs)

- **CDN: stores copies of content at CDN nodes**
  - **e.g. Netflix stores copies of MadMen**
- subscriber requests content from CDN
  - **directed to nearby copy, retrieves content**
  - **may choose different copy if network path congested**



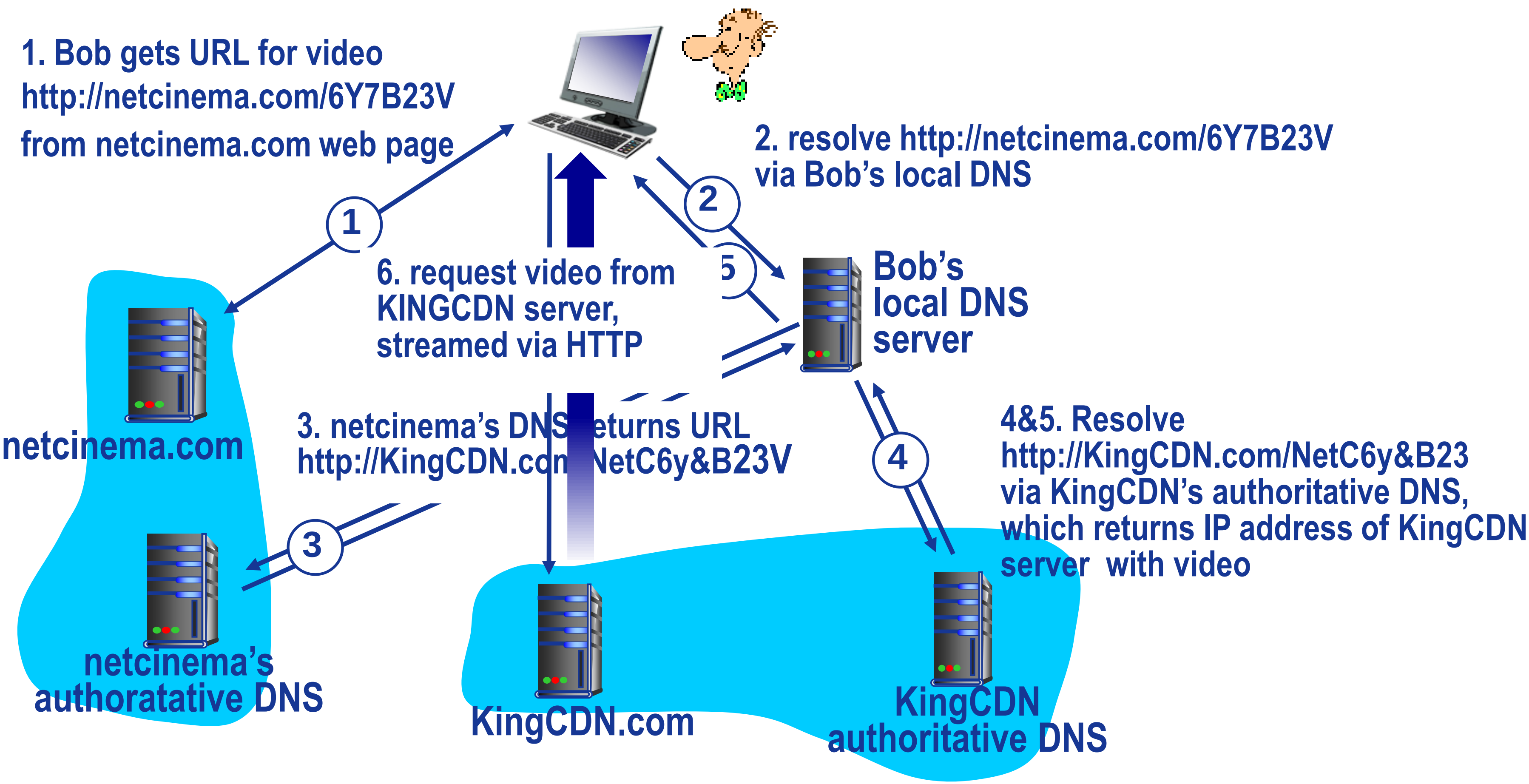
Content Distribution Networks (CDNs)

OTT challenges: coping with a congested Internet

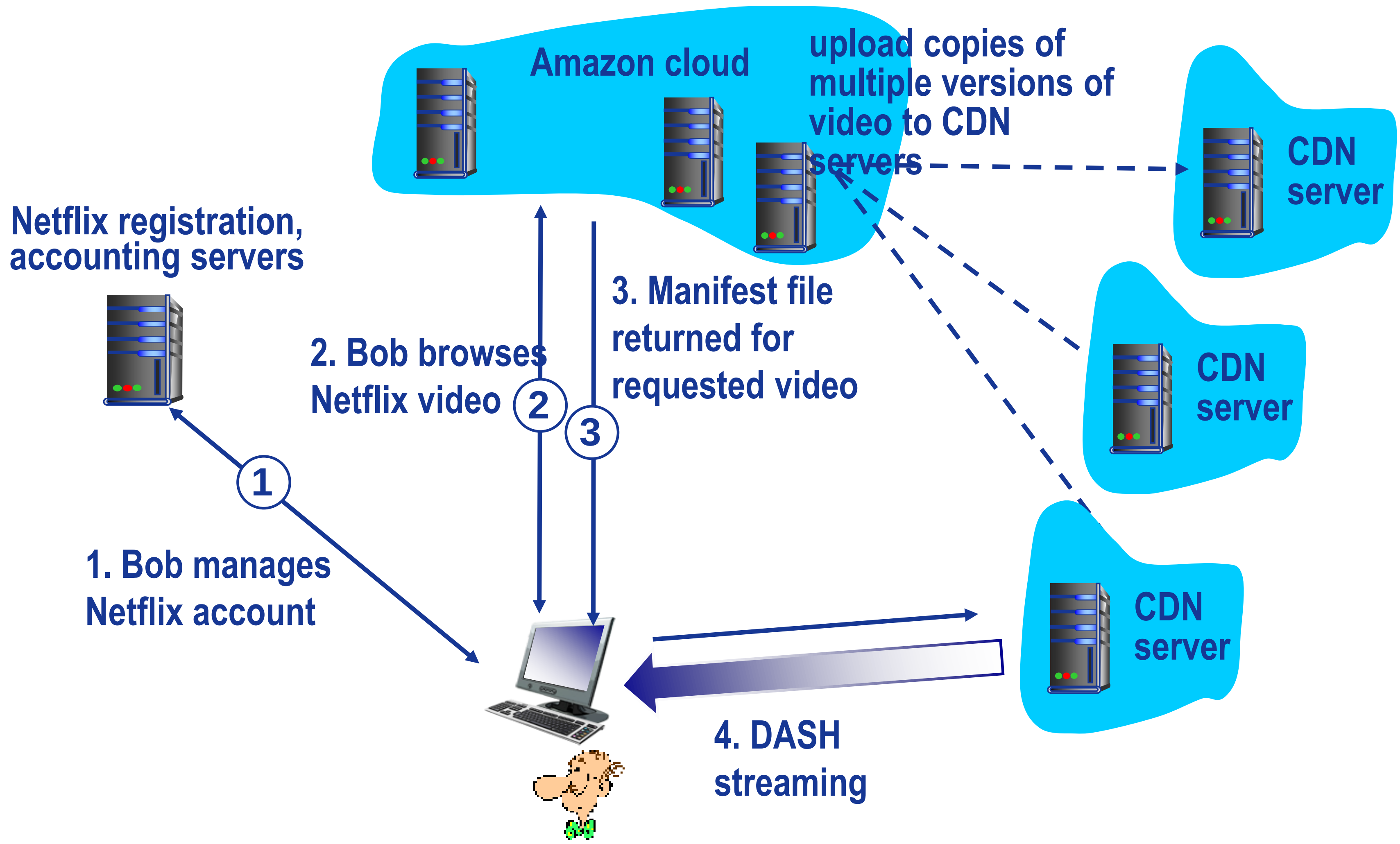
- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

CDN content access: a closer look

Bob (client) requests video http://netcinema.com/6Y7B23V
- video stored in CDN at http://KingCDN.com/NetC6y&B23V



Case study: Netflix



主观题 10分

设置

❖质疑辨惑：

1.P2P网络应用DNS吗？为什么？

2.如何理解Socket编程接口？

3.基于TCP的服务器如何实现同时与多客户通信？为什么？

4.如何优化Web应用的响应时间？

5.P2P应用如何实现对等端（Peer）与内容的索引？

作答

- 开疆拓土：P2P应用的检索技术
  - 集中式
  - 分布式

P2P: searching for information

Index in P2P system: maps information to peer location (location = IP address & port number).

File sharing (eg e-mule)

- Index dynamically tracks the locations of files that peers share.
- Peers need to tell index what they have.
- Peers search index to determine where files can be found

Instant messaging

- Index maps user names to locations.
- When user starts IM application, it needs to inform index of its location
- Peers search index to determine IP address of user.

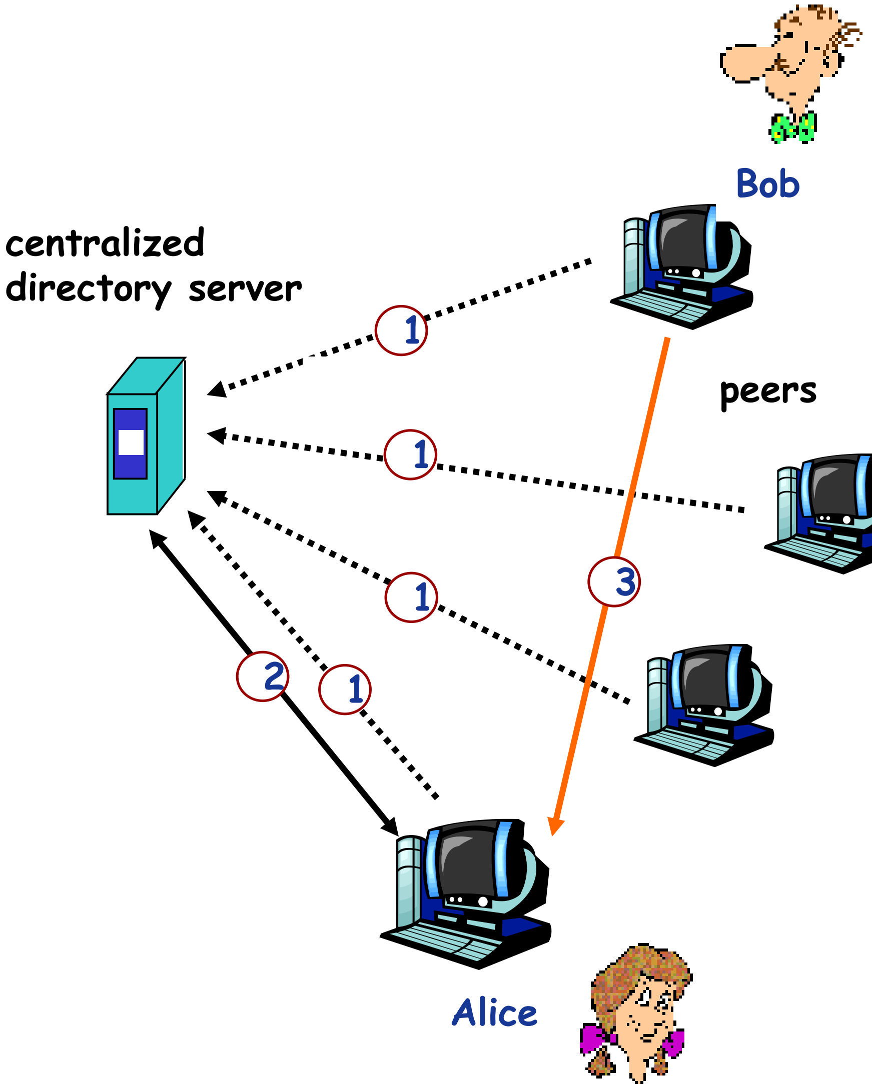
P2P: centralized index

original "Napster" design

1) when peer connects, it informs central server:
   - IP address
   - content

2) Alice queries for "Hey Jude"

3) Alice requests file from Bob



P2P: problems with centralized directory

- single point of failure
- performance bottleneck
- copyright infringement: "target" of lawsuit is obvious

file transfer is decentralized, but locating content is highly centralized

P2P: distributed index

Query flooding

- fully distributed
  - no central server
- used by Gnutella
- Each peer indexes the files it makes available for sharing (and no other files)

overlay network: graph

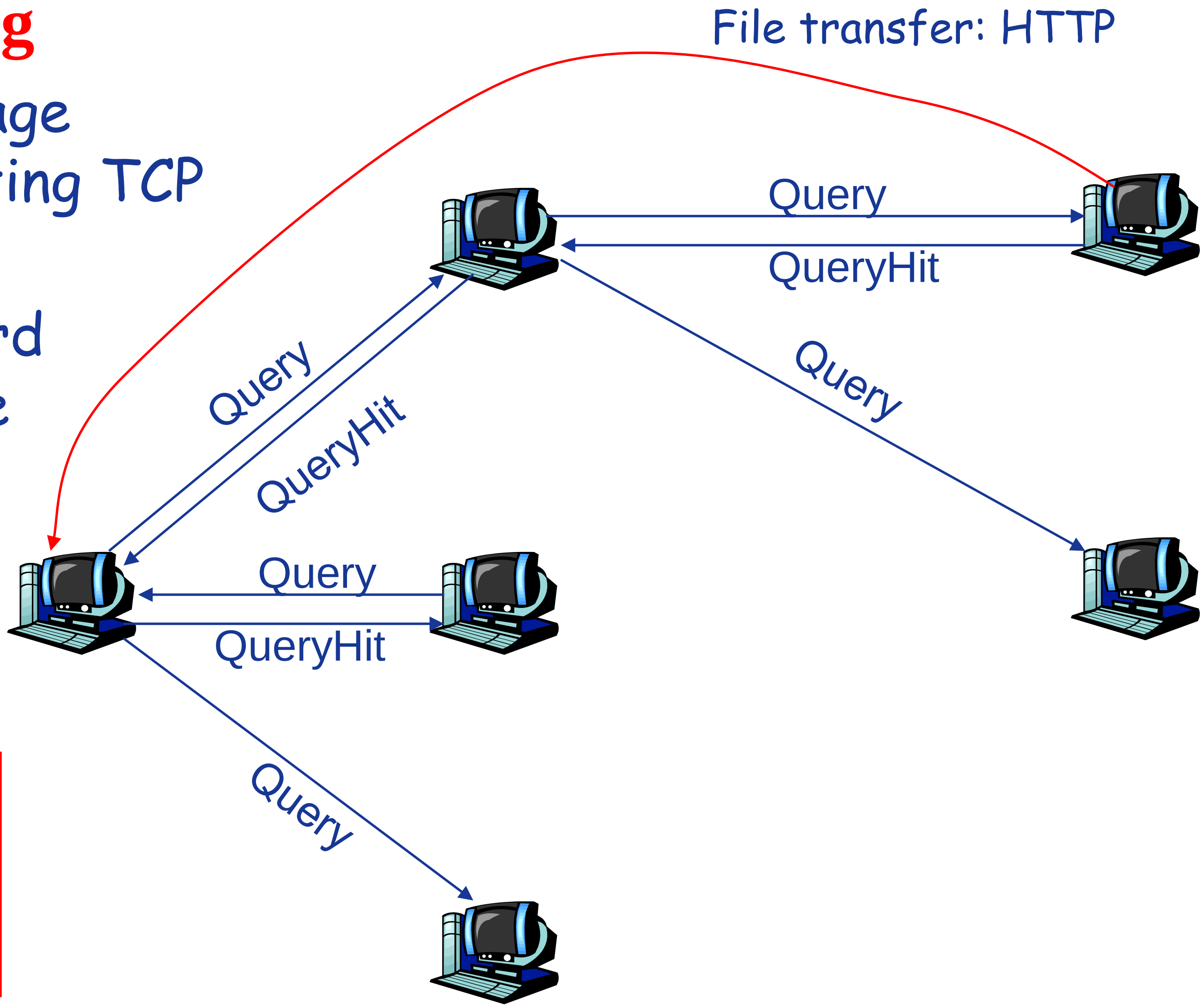
- edge between peer X and Y if there's a TCP connection
- all active peers and edges form overlay net
- edge: virtual (*not* physical) link
- given peer typically connected with < 10 overlay neighbors

P2P: distributed index

Query flooding

- Query message sent over existing TCP connections
- peers forward Query message
- QueryHit sent over reverse path

Scalability: limited scope flooding



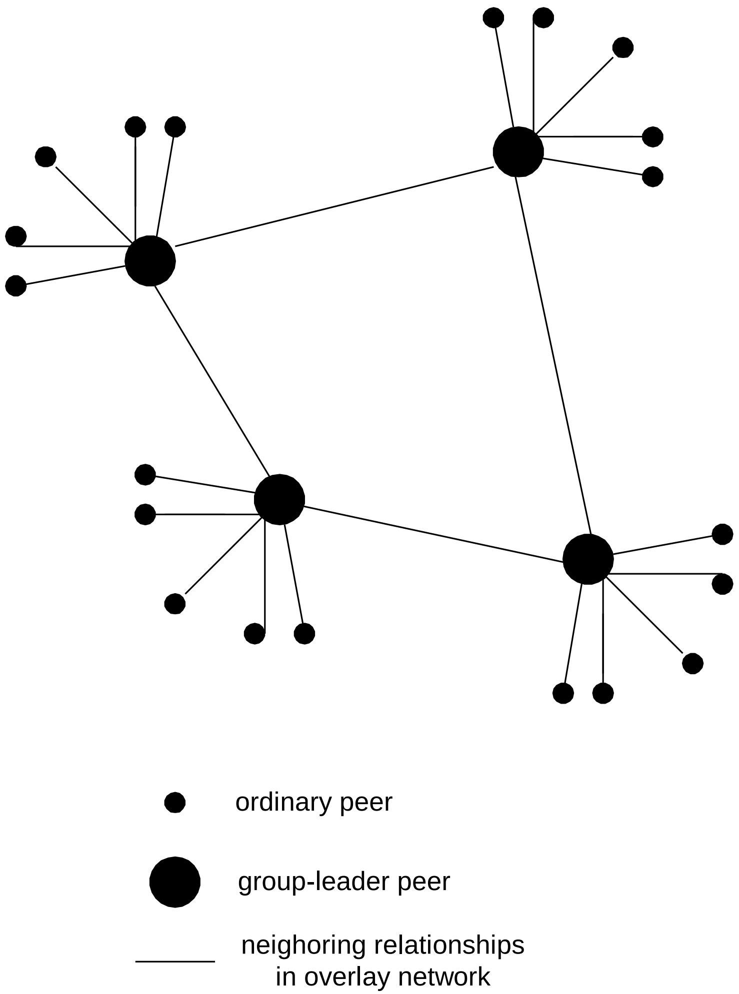
Gnutella: Peer joining

1. joining peer Alice must find another peer in Gnutella network: use list of candidate peers
2. Alice sequentially attempts TCP connections with candidate peers until connection setup with Bob
3. *Flooding:* Alice sends Ping message to Bob; Bob forwards Ping message to his overlay neighbors (who then forward to their neighbors….)
   - peers receiving Ping message respond to Alice with Pong message
4. Alice receives many Pong messages, and can then setup additional TCP connections

P2P: distributed index

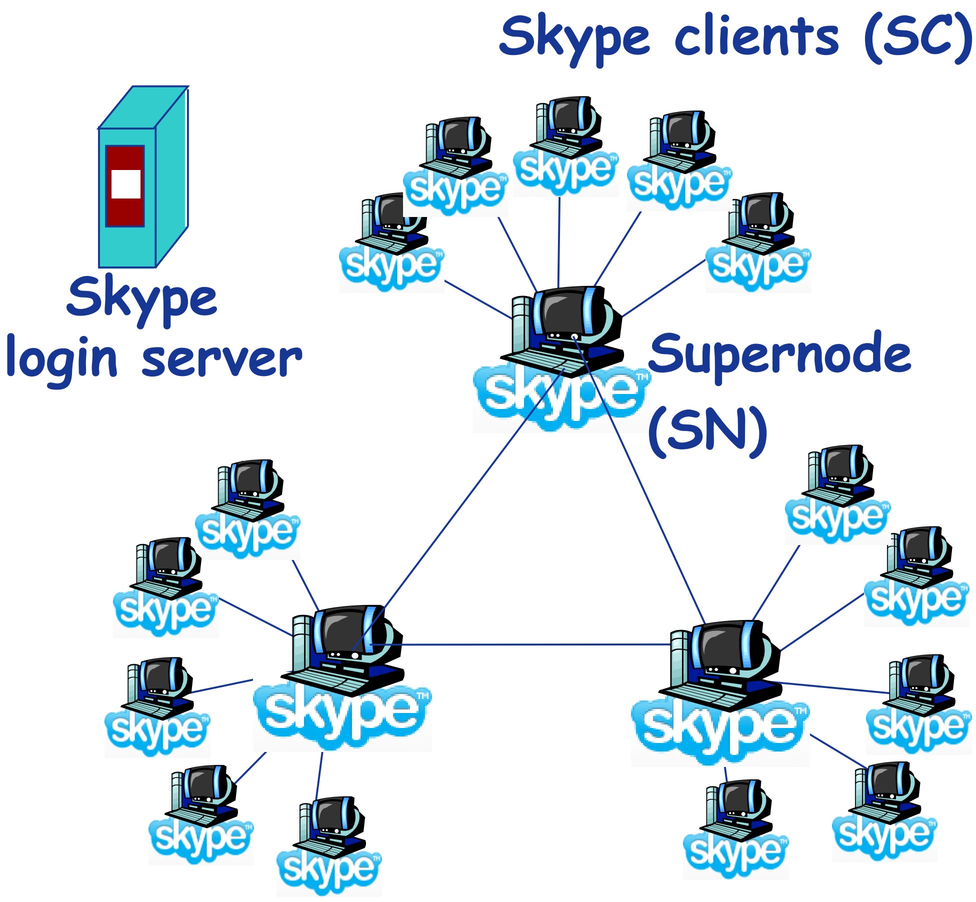
Hierarchical Overlay

- between centralized index, query flooding approaches
- each peer is either a *super node* or assigned to a super node
  - TCP connection between peer and its super node.
  - TCP connections between some pairs of super nodes.
- Super node tracks content in its children



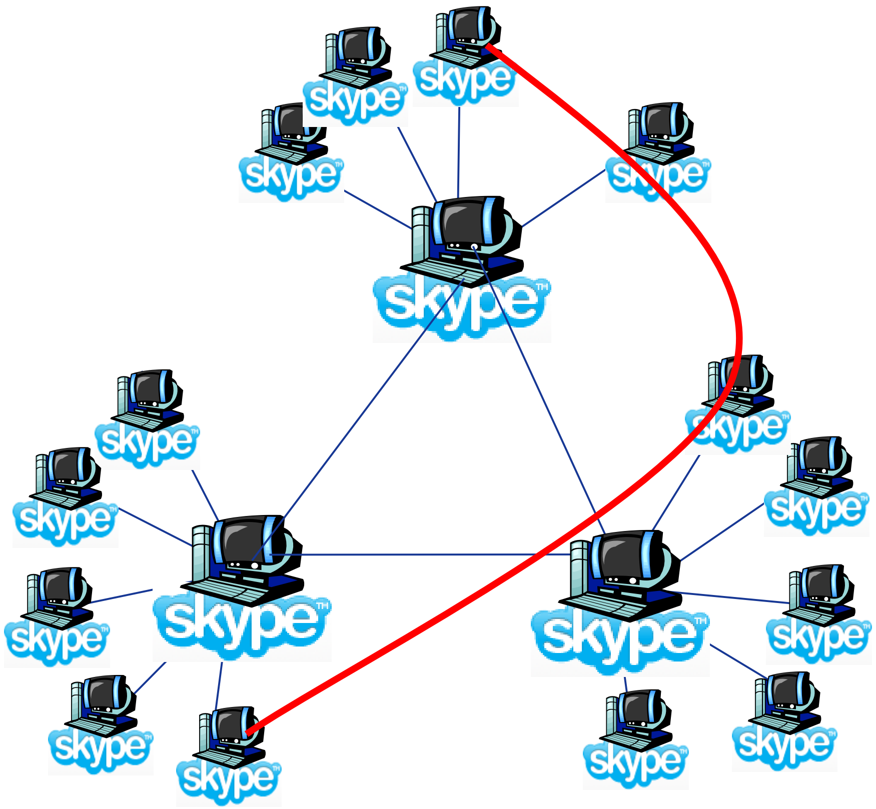
P2P Case study: Skype

- inherently P2P: pairs of users communicate.
- proprietary application-layer protocol (inferred via reverse engineering)
- hierarchical overlay with SNs
- Index maps usernames to IP addresses; distributed over SNs



Skype: Peers as relays

- Problem when both Alice and Bob are behind "NATs".
  - NAT prevents an outside peer from initiating a call to insider peer
- Solution:
  - Using Alice's and Bob's SNs, Relay is chosen
  - Each peer initiates session with relay.
  - Peers can now communicate through NATs via relay



P2P: distributed index

Distributed Hash Table (DHT)

- DHT: a *distributed P2P database*
- database has (key, value) pairs; examples:
  - key: ss number; value: human name
  - key: movie title; value: IP addresses
- a peer queries DHT with key
  - DHT returns values that match the key
- peers can also insert (key, value) pairs
- Distribute the (key, value) pairs over the (millions of peers)

Q: how to assign keys to peers?

- central issue:
  - assigning (key, value) pairs to peers.
- basic idea:
  - convert each key to an integer
  - Assign integer to each peer
  - put (key,value) pair in the peer that is closest to the key

DHT identifiers

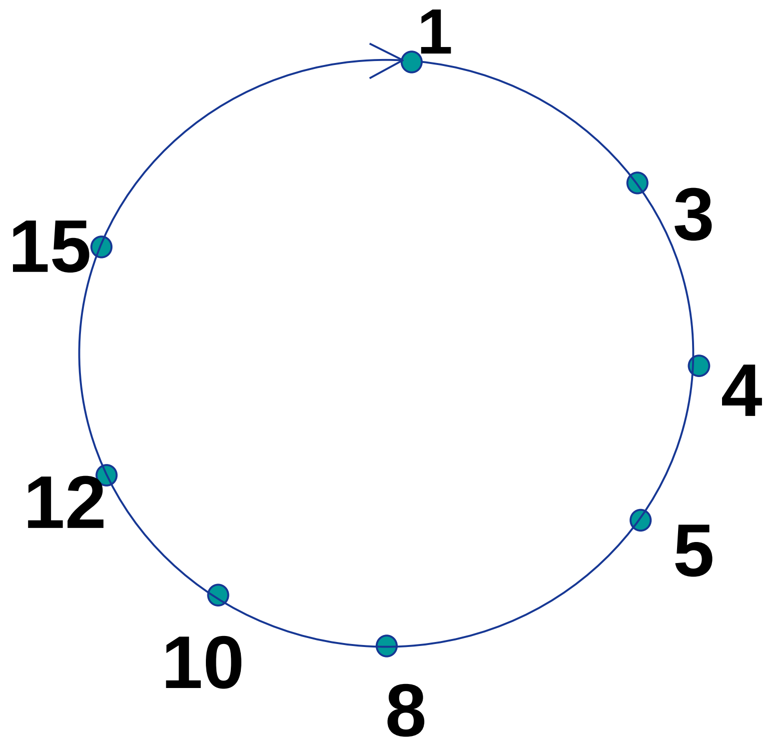
- assign integer identifier to each peer in range [0,2^n-1] for some *n*.
  - each identifier represented by *n* bits.

- require each key to be an integer in same range
- to get integer key, hash original key
  - *e.g.*, key = hash("Led Zeppelin IV")
  - this is why its is referred to as a *distributed "hash" table*

Assign keys to peers

- rule: assign key to the peer that has the *closest* ID.
- convention in lecture: closest is the *immediate successor* of the key.
- e.g., n=4; peers: 1,3,4,5,8,10,12,14;
  - key = 13, then successor peer = 14
  - key = 15, then successor peer = 1

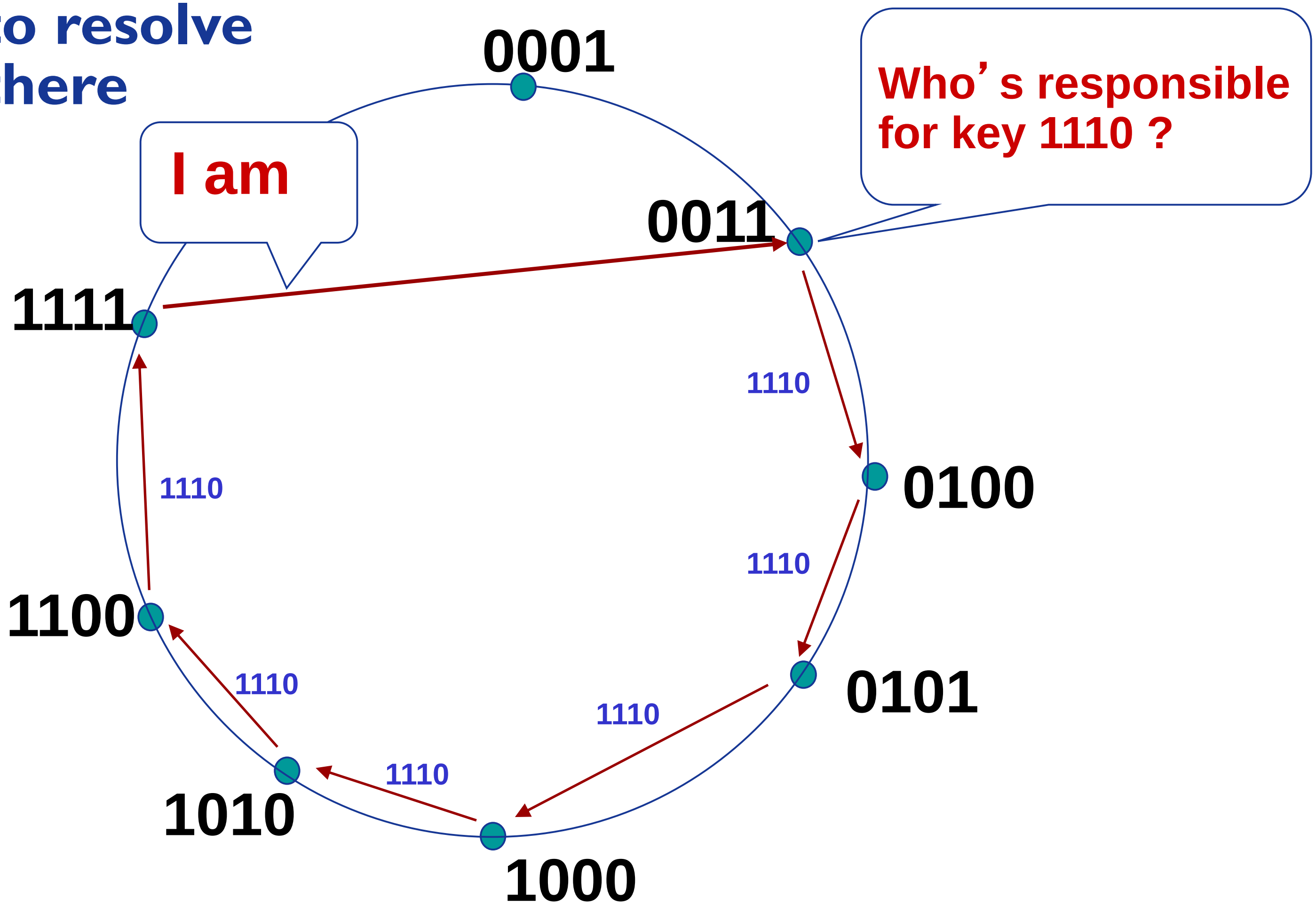
Circular DHT (I)



- each peer *only* aware of immediate successor and predecessor.
- "overlay network"

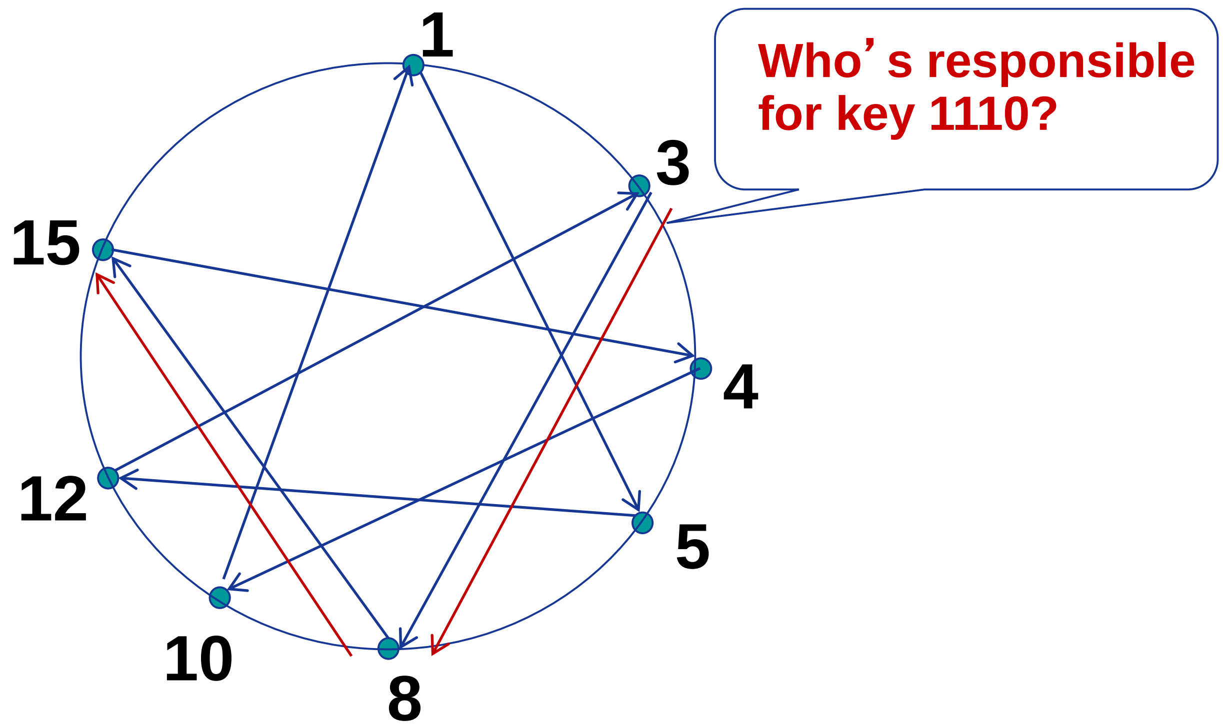
Circular DHT (I)

***O*(*N*) messages on avgerage to resolve query, when there are *N* peers**



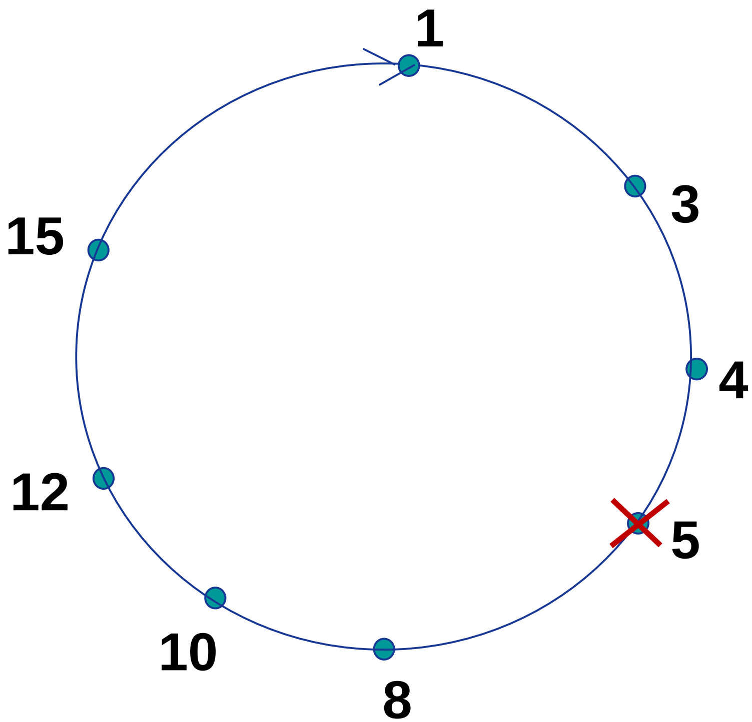
Define closest as closest successor

Circular DHT with shortcuts



- each peer keeps track of IP addresses of predecessor, successor, short cuts.
- reduced from 6 to 2 messages.
- possible to design shortcuts so *O(log N)* neighbors, *O(log N)* messages in query

Peer churn



handling peer churn:

- peers may come and go (churn)
- each peer knows address of its two successors
- each peer periodically pings its two successors to check aliveness
- if immediate successor leaves, choose next successor as new immediate successor

example: peer 5 abruptly leaves

- peer 4 detects peer 5 departure; makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8’s immediate successor its second successor.
- what if peer 13 wants to join?

❖ 解疑释惑：

- 覆盖网络（overlay network）中的连接是不是一定是TCP连接？
- 纯P2P网络应用在Peer加入时如何知道与哪些Peers连接/联系？
- Email算P2P应用吗？
- 为什么基于UDP的客户端也能调用connect（）？ 真的会建立连接吗？
- ……

第3周 课堂教学-应用层（下）

❖实战拓展：

- 开发一个支持文件共享的P2P应用
  - 基于DHT
- 设计并实现一个简单的路由器
  - 基于WinPcap/Pcap
  - 支持多网卡/单网卡
- 以组为单位

第4周 课堂教学-传输层（上）

- 束广就狭：（20分钟，**第3组总结**）
  - 传输层服务、复用与分用、UDP协议、可靠数据传输（停等协议、滑动窗口协议）
- 质疑辨惑：（50分钟）
  - 1.如何理解复用与分用？复用与分用只在传输层进行吗？
  - 2.实现可靠数据传输的主要机制有哪些？能实现100%可靠吗？
  - 3.如何理解滑动窗口协议？
  - 4.滑动窗口协议窗口大小与序列号比特位数有什么关系？为什么？
  - ……
- 解疑释惑：（10分钟）
  - 解答疑问
- 演武修文：（20分钟）
  - 课堂测验
  - 讲解

假设采用P2P方式为1000个客户分发文件F，文件F初始位于某服务器上；服务器接入Internet链路的上行带宽u_s=1000Mbps；每个客户接入Internet的链路下行带宽d=10Mbps，上行带宽u=1Mbps（注：M=10^6）。若F=1MB，则完成文件F分发所需时间至少为

A 0.008s

B 0.8s

C 4s

D 8s

提交

谢 谢！