

Computer Networks and Applications

COMP 3331/COMP 9331

Week 4
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro
P2P + CDN + Transfer Part 1

**Reading Guide: Chapter 2, 2.5, 2.6, 2.7 +
Chapter 3, Sections 3.1 – 3.4**

Application Layer: outline

2.1 principles of network applications

2.5 P2P applications

2.2 Web and ~~HTTP~~

2.6 video streaming and content distribution

2.3 electronic m <https://eduassistpro.github.io/> orks (CDNs)

- SMTP, POP3, IMAP

2.4 DNS

programming
DP and TCP

Self study

Pure P2P architecture

- ❖ no always-on server
- ❖ arbitrary end systems directly communicate
- ❖ peers are intermittently connected and change IP addresses

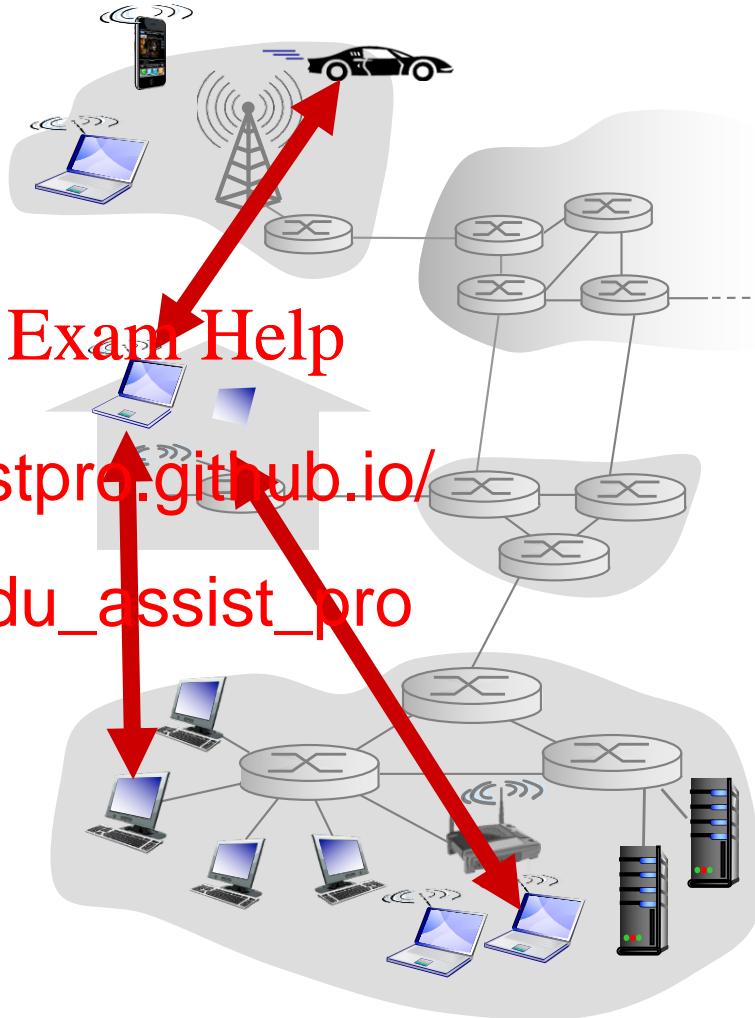
Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

examples:

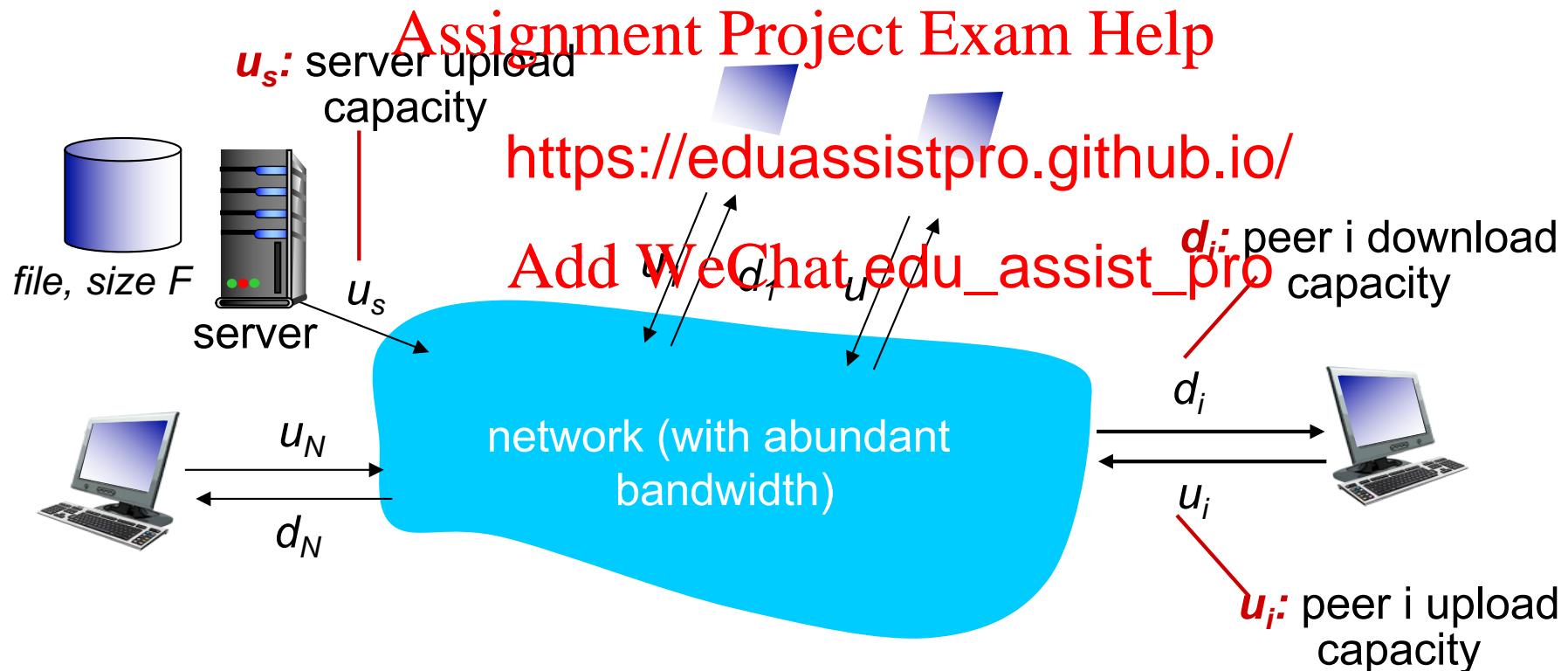
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



File distribution: client-server vs P2P

Question: how much time to distribute file (size F) from one server to N peers?

- peer upload/download capacity is limited resource



File distribution time: client-server

- ❖ **server transmission:** must send (upload) N file copies:

- time to send one copy: F/u_s
- time to send N copies: NF/u_s

Assignment Project Exam Help



- ❖ **client:** each client <https://eduassistpro.github.io/> download file cop

- d_{\min} = min client download rate
- client download time: F/d_{\min}

time to distribute F
to N clients using
client-server approach

$$D_{c-s} \geq \max\{NF/u_s, F/d_{\min}\}$$

increases linearly in N

File distribution time: P2P

- ❖ **server transmission:** must upload at least one copy
 - time to send one copy: F/u_s

- ❖ **client:** each client must download file copy
 - client download time:

- ❖ **clients:** as aggregate
 - max upload rate (limiting max download rate)

Assignment Project Exam Help

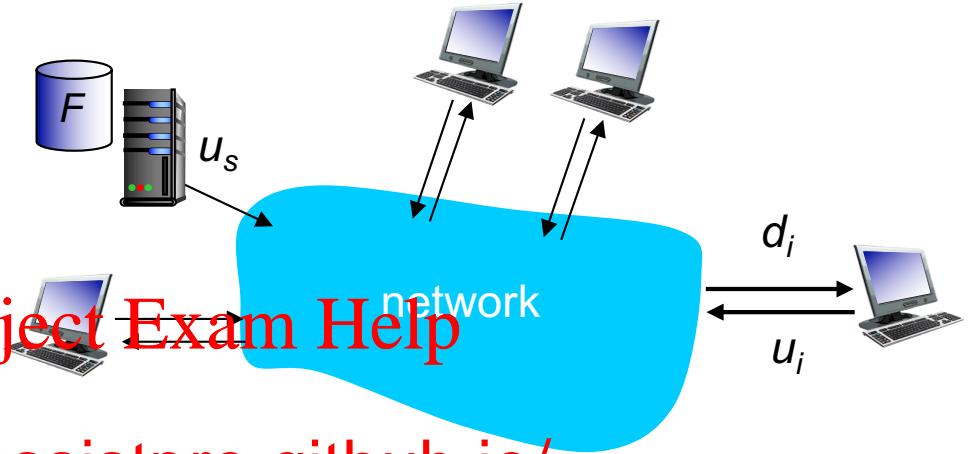
<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

time to distribute F
to N clients using
P2P approach

$$D_{P2P} \geq \max\{F/u_s, F/d_{min}, NF/(u_s + \sum_{i=1}^N u_i)\}$$

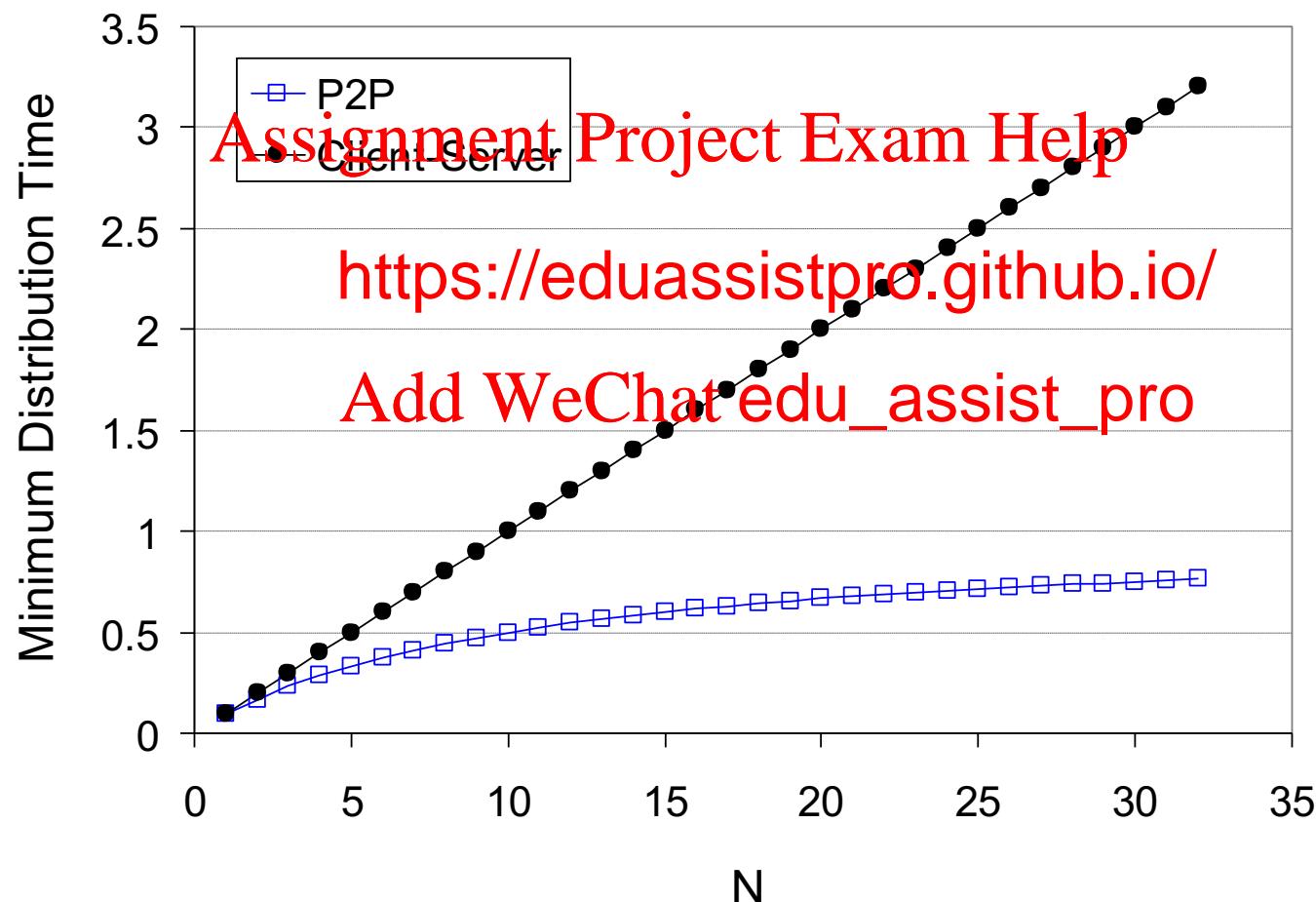
increases linearly in N ...

... but so does this, as each peer brings service capacity



Client-server vs. P2P: example

client upload rate = u , $F/u = 1$ hour, $u_s = 10u$

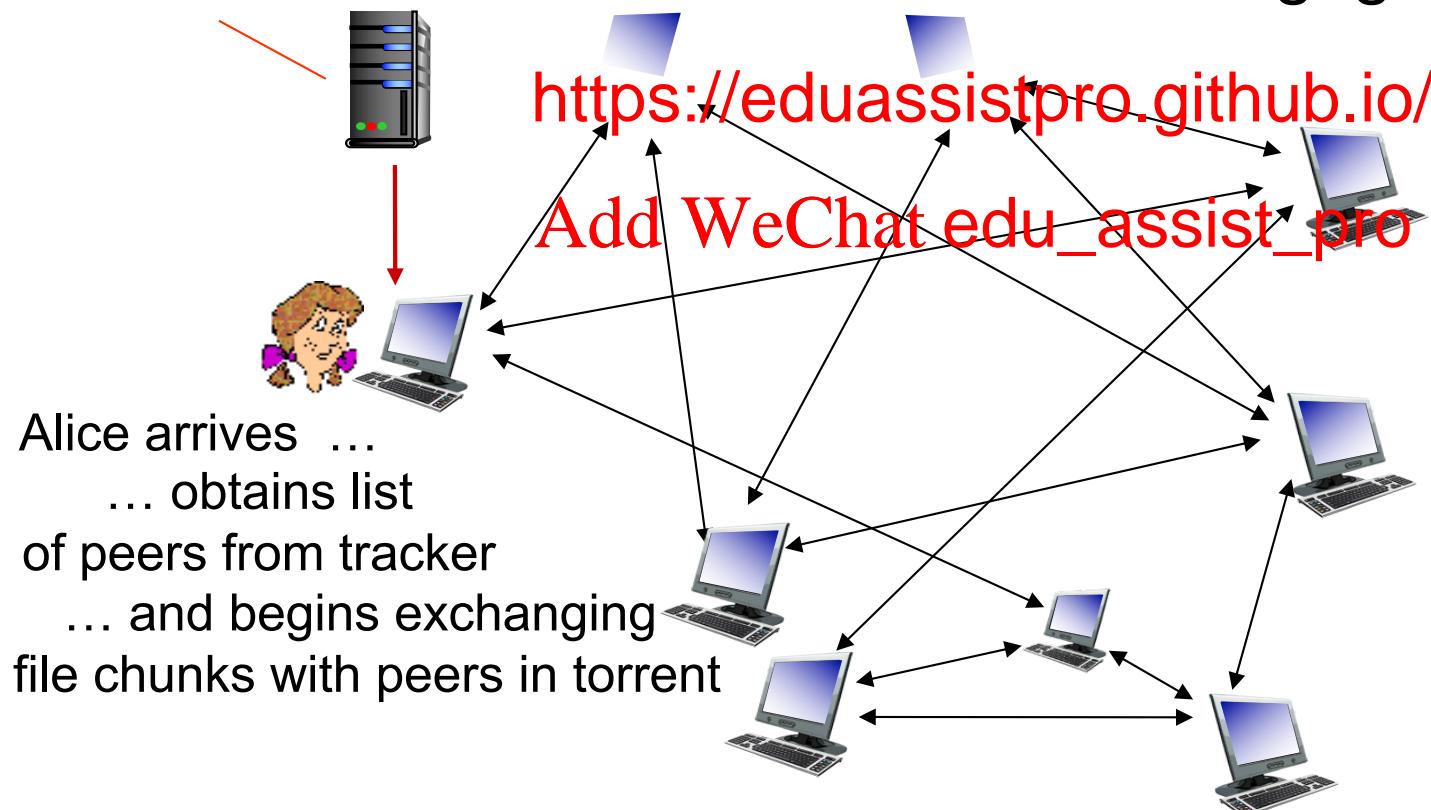


P2P file distribution: BitTorrent

- ❖ file divided into 256KB chunks
- ❖ peers in torrent send/receive file chunks

tracker: tracks peers
participating in torrents

torrent: group of peers
exchanging chunks of a file



.torrent files

- ❖ Contains address of trackers for the file
 - Where can I find other peers?
Assignment Project Exam Help
- ❖ Contain a list of trackers
 - This ensures that chunks are modified

Title	Trackers
House of Cards Season 4	Tracker1-url
Walking Dead Season 6	Tracker2-url
Game of Thrones Season 7	Tracker2-url, Tracker3-url

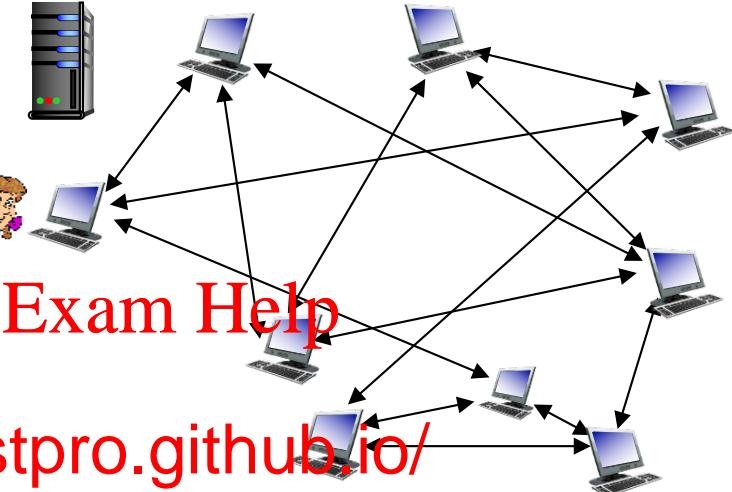
P2P file distribution: BitTorrent

- ❖ peer joining torrent:

- has no chunks, but will accumulate them over time
- registers with list of peers, c <https://eduassistpro.github.io/> subset of peer (“neighbours”)

Assignment Project Exam Help

Add WeChat edu_assist_pro



- ❖ while downloading, peer uploads chunks to other peers
- ❖ peer may change peers with whom it exchanges chunks
 - ❖ *churn*: peers may come and go
- ❖ once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

BitTorrent: requesting, sending file chunks

requesting chunks:

- ❖ at any given time, different peers have different subsets of file chunks
- ❖ periodically, Alice peer for list of chunks they have
- ❖ Alice requests missing chunks from peers, rarest first
- ❖ **Q:** Why rarest first?

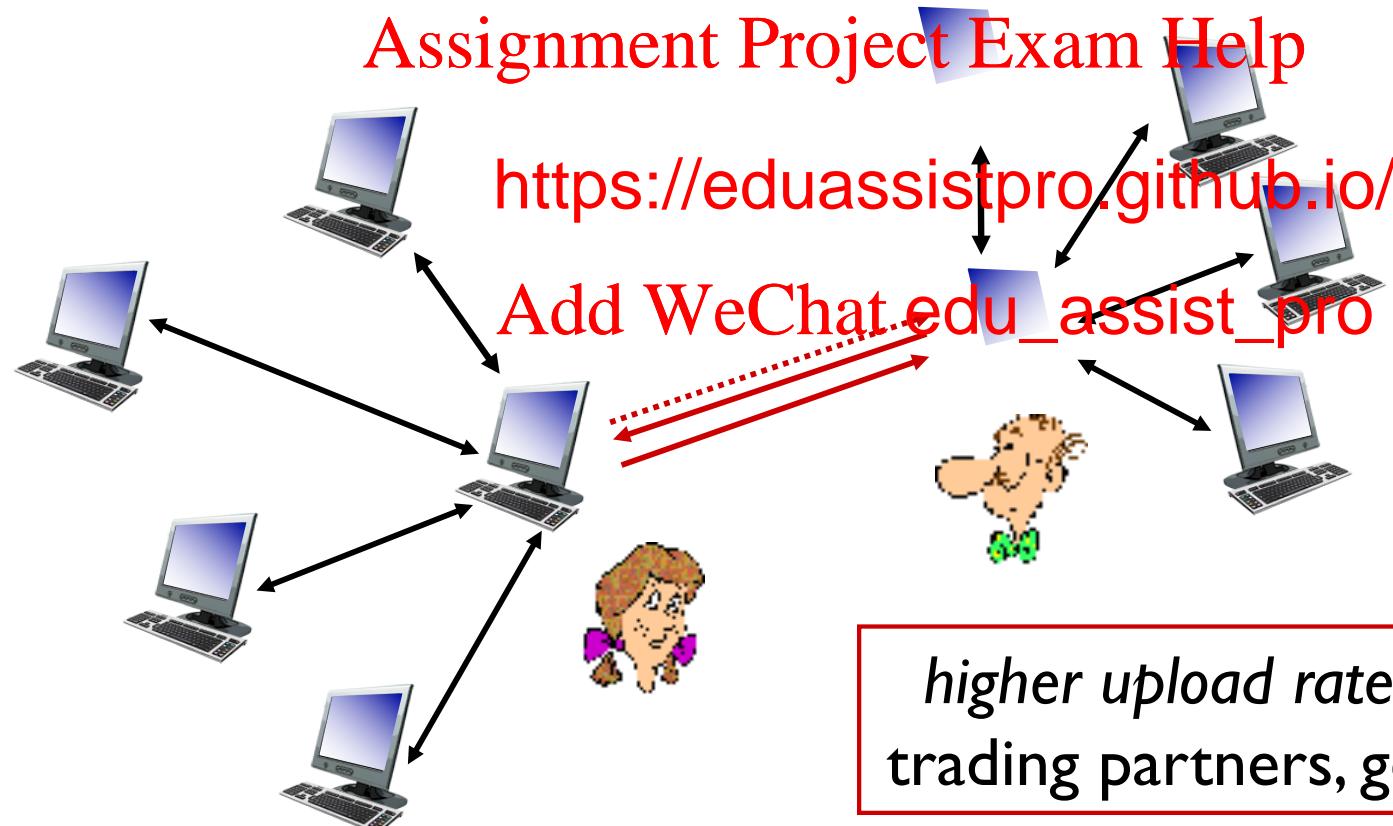
sending chunks: tit-for-tat

- ❖ Alice sends chunks to those four peers currently sending her chunks at *Highest rate*
- ❖ peers are choked by Alice (receive chunks from her) rate top 4 every 10 secs
- ❖ randomly select peer, starts sending chunks
 - “optimistically unchoke” this peer
 - newly chosen peer may join top 4

Add WeChat <https://eduassistpro.github.io/>

BitTorrent: tit-for-tat

- (1) Alice “optimistically unchoke” Bob
- (2) Alice becomes one of Bob’s top-four providers; Bob reciprocates
- (3) Bob becomes one of Alice’s top-four providers



Quiz: Free-riding



- ❖ Suppose Todd joins a BitTorrent torrent, but he does not want to upload any data to any other peers. Todd claims that he can receive a complete copy of the file that is shared by the swarm. Is Todd's claim <https://eduassistpro.github.io/> Assignment Project Exam Help? hy not?

Add WeChat edu_assist_pro

Getting rid of the server/tracker

- ❖ Distribute the tracker information using a Distributed Hash Table (DHT)
Assignment Project Exam Help
- ❖ A DHT is a <https://eduassistpro.github.io/>
 - Maps keys to an arbitrary value
 - Works a lot like, well ...

Content available in 6th Edition of the textbook Section 2.6.2

Hash table - review

- ❖ (key,value) pairs
- ❖ Centralised hash table – all (key,value) pairs on 1 node
- ❖ Distributed hash tables – each node has a “section” of (key,value) pairs

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Distributed Hash Table (DHT)

- ❖ DHT: a *distributed P2P database*
- ❖ database has (key, value) pairs; examples:
 - key: TFN number; value: human name
Assignment Project Exam Help
 - key: file name; value: peer(s)
https://eduassistpro.github.io/
- ❖ Distribute t
s over the millions of peers
Add WeChat edu_assist_pro
- ❖ a peer **queries** DHT with key
 - DHT returns values that match the key
- ❖ peers can also **insert** (key, value) pairs

Challenges

- ❖ How do we assign (key, value) pairs to nodes?

Assignment Project Exam Help

- ❖ How do we <https://eduassistpro.github.io/> quickly?

Add WeChat edu_assist_pro

- ❖ What happens if nodes e?

Q: how to assign keys to peers?

- ❖ basic idea:

- convert each key to an integer
- Assign integer to each peer
- put (key, val) to the key

<https://eduassistpro.github.io/>
Add WeChat edu_assist_pro

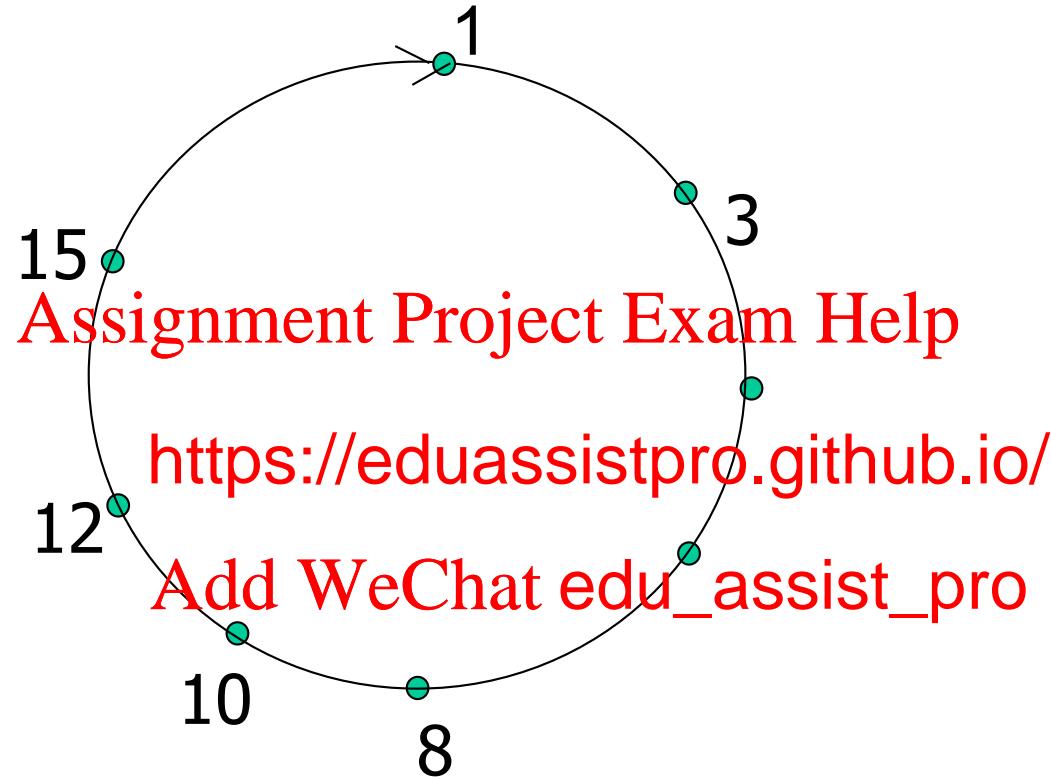
DHT identifiers: Consistent Hashing

- ❖ assign integer identifier to each peer in range $[0, 2^n - 1]$ for some n -bit hash function
 - E.g., node ID is hash of its IP address
- ❖ require each peer in same range
- ❖ to get integer key
 - e.g., key = hash("House season 4")
 - this is why it's referred to as a *distributed "hash" table*

Assign keys to peers

- ❖ rule: assign key to the peer that has the *closest* ID.
- ❖ common convention: closest is the *immediate successor* of the key.
- ❖ e.g., $n=4$; all peers in the range $[0, 15]$, peers ~~https://eduassistpro.github.io/~~^{Assignment Project Exam Help}~~Add WeChat edu_assist_pro~~ 8, 10, 12, 14;
 - key = 13, then successor = 4
 - key = 15, then successor peer = 1

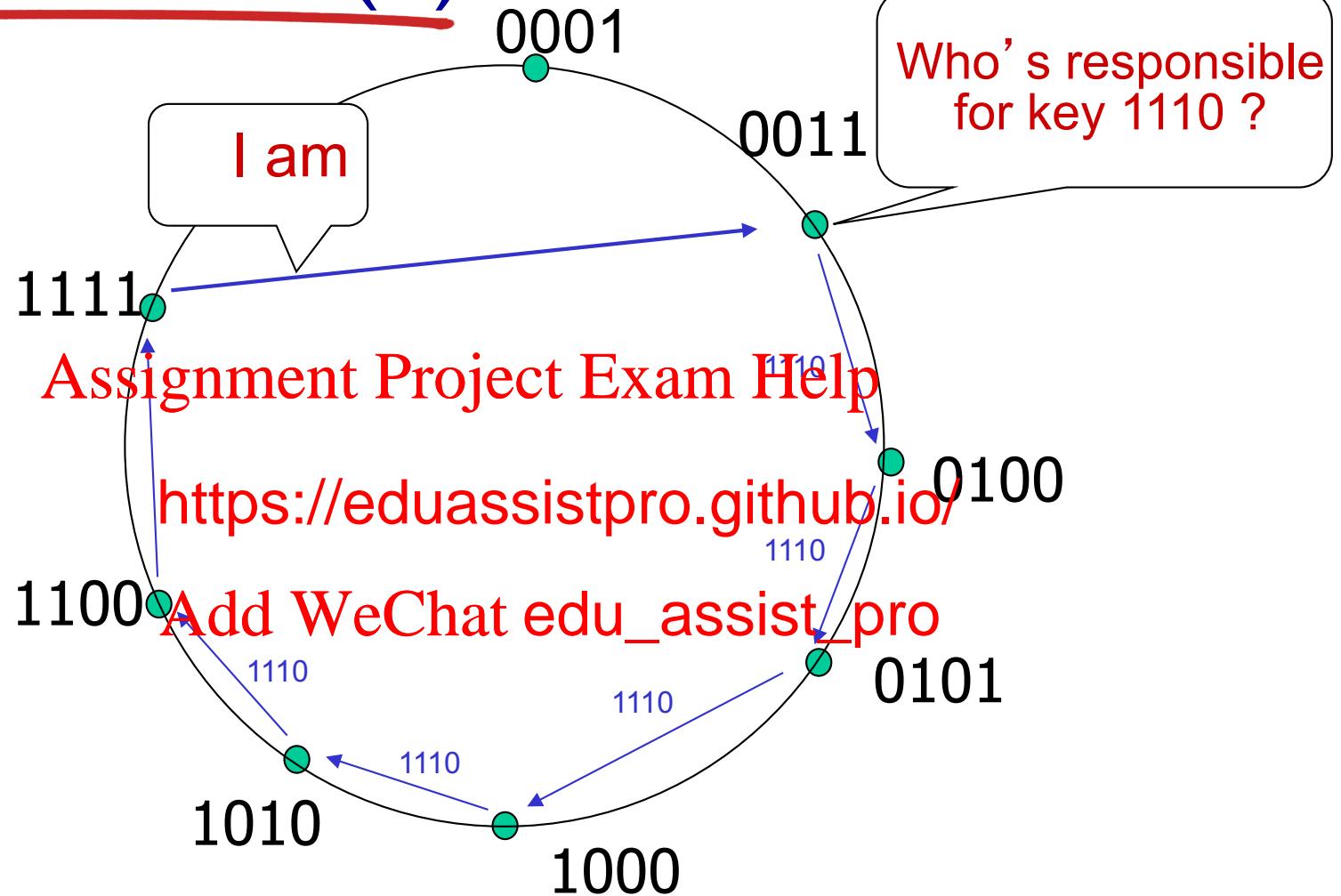
Circular DHT (I)



- ❖ each peer *only* aware of immediate successor and predecessor.
- ❖ “overlay network”

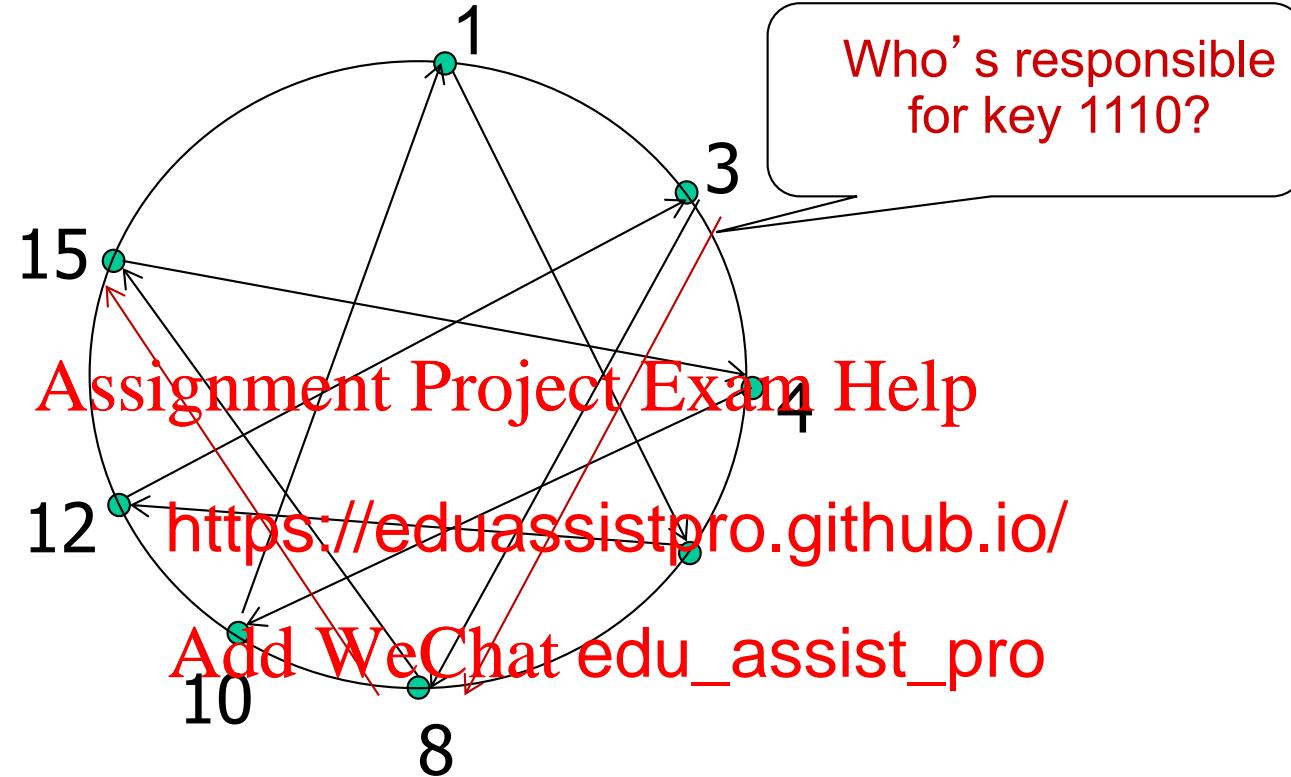
Circular DHT (2)

Define closest as closest successor



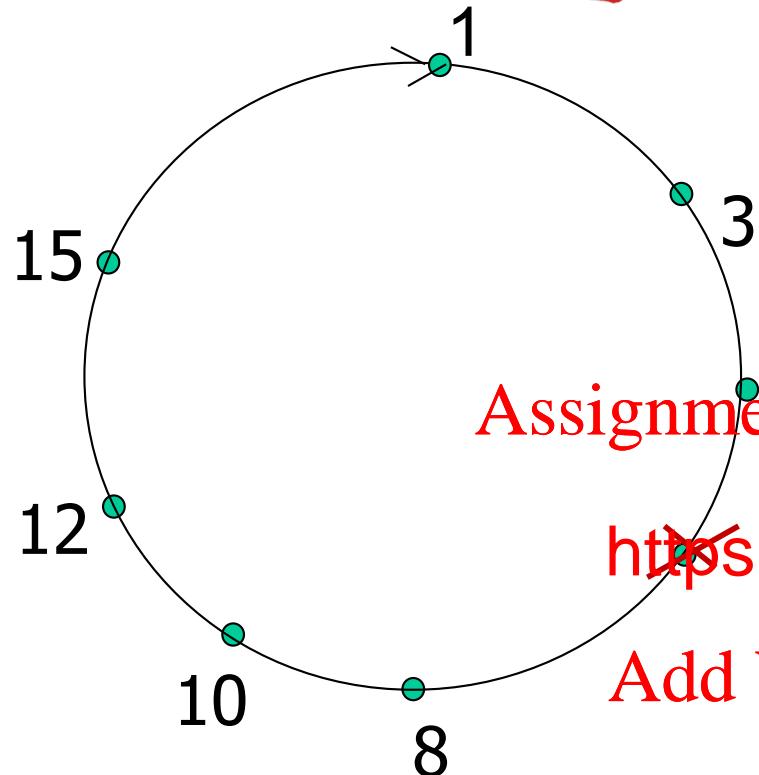
Worst case all peers probed, N messages, on average $N/2$
Mesh overlay (each peer tracks all other $N-1$ peers) only one message is sent per query

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 $O(\log N)$ neighbours, $O(\log N)$ messages in query

Peer churn



handling peer churn:

- ❖ peers may come and go (churn)
- ❖ each peer knows address of its two successors (I knows 3 & 4)
- ❖ each peer periodically pings its two successors to check aliveness
- ❖ If immediate successor leaves, t successor as new successor

example: peer 5 abruptly leaves

- ❖ peer 4 & 3 detect peer 5 departure; 4 makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8's immediate successor its second successor. 3 probes 4 for the new successor

More DHT info

- ❖ How do nodes join?
- ❖ How does Assignment Project Exam Help work?

<https://eduassistpro.github.io/>

- ❖ How much state does it store?

Research Papers (on WebCMS):
Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications
Dynamo: Amazon's Highly Available Key-value Store

Application Layer: outline

2.1 principles of network applications

2.5 P2P applications

2.2 Web and ~~HTTP~~ Assignment Project Exam Help

2.6 video streaming and content distribution

2.3 electronic m <https://eduassistpro.github.io/> orks (CDNs)

- SMTP, POP3, IMAP

2.4 DNS

programming
DP and TCP

Self study

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 <https://eduassistpro.github.io/>
- challenge: hete
 - different users have different capabilities versus mobile; bandwidth rich versus b
- *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 repr 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)

.....

<https://eduassistpro.github.io/>

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 c
- examples: <https://eduassistpro.github.io/>
 - MPEG I (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)

.....

Assignment Project Exam Help

Add WeChat edu_assist_pro

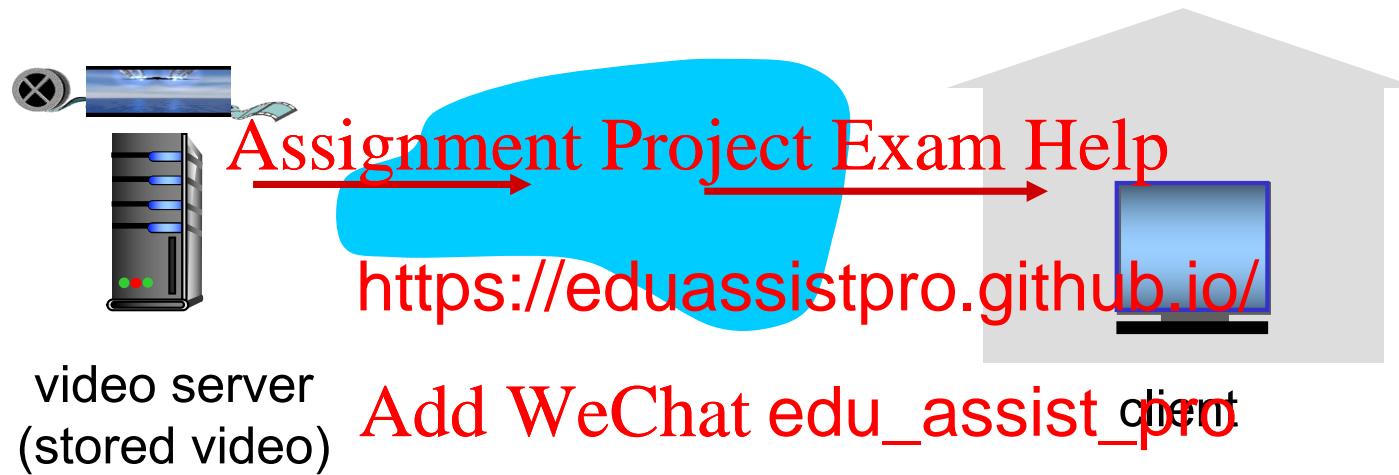
temporal coding example:
instead of sending complete frame at $i+1$, send only differences from frame i



frame $i+1$

Streaming stored video:

simple scenario:



Streaming multimedia: DASH

- ❖ *DASH: Dynamic, Adaptive Streaming over HTTP*
- ❖ *server:*
 - divides video file into multiple chunks
 - each chunk s fferent rates
 - *manifest file:* <https://eduassistpro.github.io/> fferent chunks
- ❖ *client:* Add WeChat edu_assist_pro
 - 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: Dynamic, Adaptive Streaming 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 (RL server that is “close” to client)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

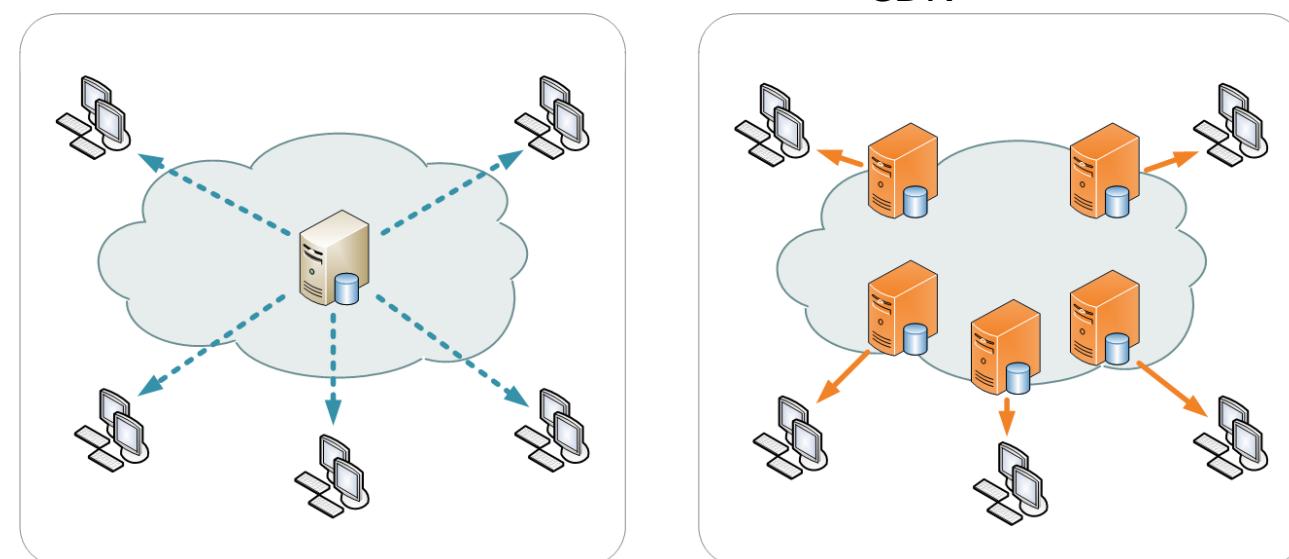
Add WeChat edu_assist_pro

DASH implementations

- ❖ Android through ExoPlayer
- ❖ Samsung, LG, Sony, Philips Smart TVs
- ❖ Youtube Assignment Project Exam Help
- ❖ Netflix <https://eduassistpro.github.io/>
- ❖ JavaScript Implementation https://edu_assist_pro5.AddWeChat.com
- ❖

Content distribution networks

- ❖ Caching and replication as a service (amortise cost of infrastructure)
- ❖ Goal: bring content close to the user
- ❖ Large-scale distributed storage infrastructure (usually) administered by one entity
 - e.g., Akamai has ser
- ❖ Combination of (pull) c <https://eduassistpro.github.io/>
 - Pull: Direct result of clients' request
 - Push: Expectation of high access rat



An example

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat `edu_assist_pro`

Many well-known sites
are hosted by CDNs. A
simple way to check
using dig is shown here.

Content distribution networks

- *challenge*: how to stream content (selected from millions of videos) to hundreds of thousands of *simultaneous* users?
Assignment Project Exam Help
- *option 1*: sin <https://eduassistpro.github.io/>
 - single point ~~of failure~~ Add WeChat edu_assist_pro
 - 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?

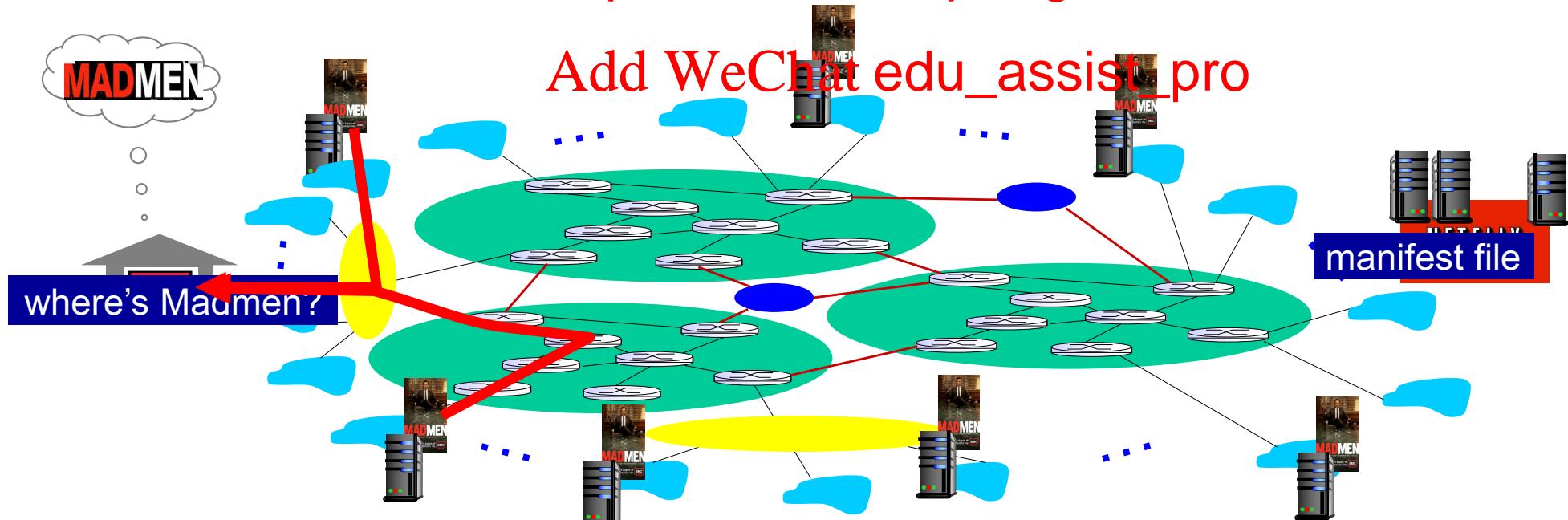
Assignment Project Exam Help

- ❖ *option 2*: store ~~https://eduassistpro.github.io/~~ copies of videos at multiple geographic locations ~~Add WeChat~~ ~~edu_assist_sites~~ (*CDN*)
 - *enter deep*: push CDN servers deep into many access networks
 - close to users
 - used by Akamai, thousands of locations
 - *bring home*: smaller number (10's) of larger clusters in POPs near (but not within) access networks
 - used by Limelight

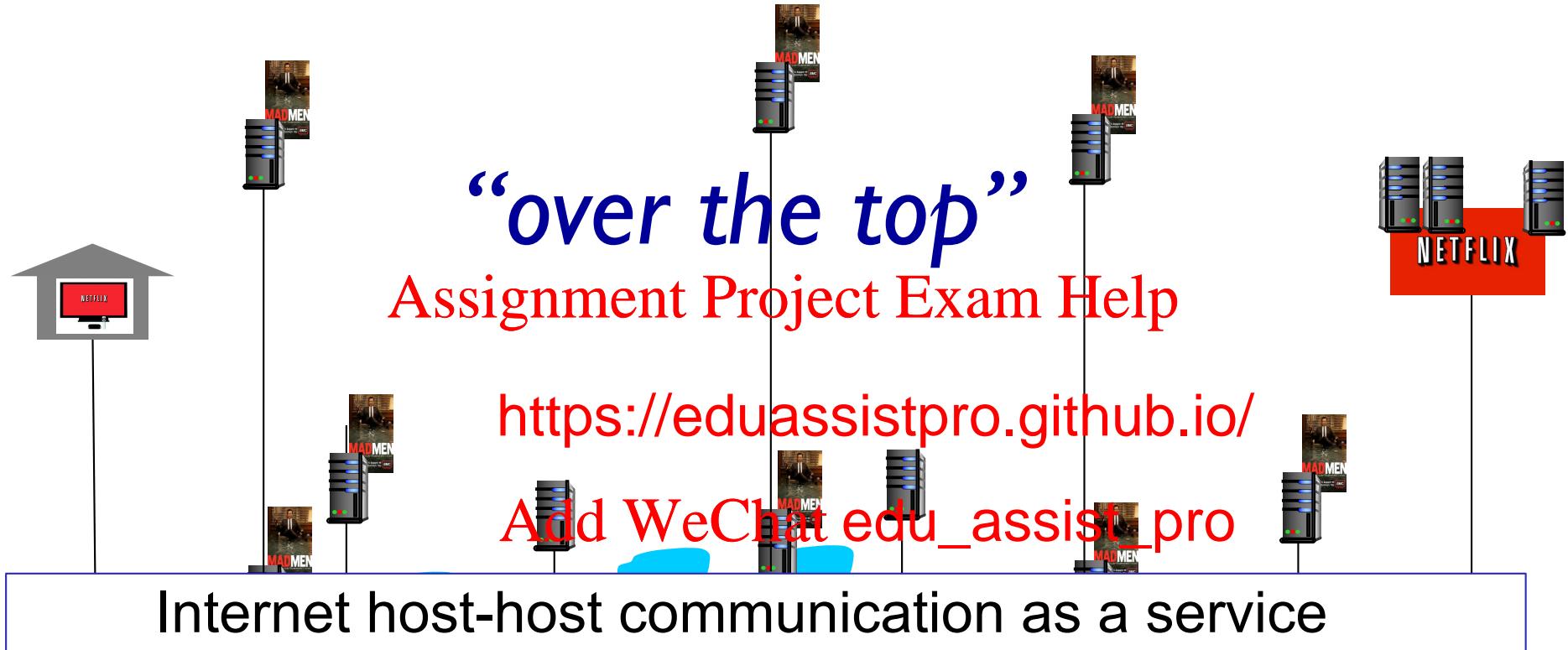
Content Distribution Networks (CDNs)

- 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 path if network path congested

<https://eduassistpro.github.io/>



Content Distribution Networks (CDNs)



OTT challenges: coping with a congested Internet

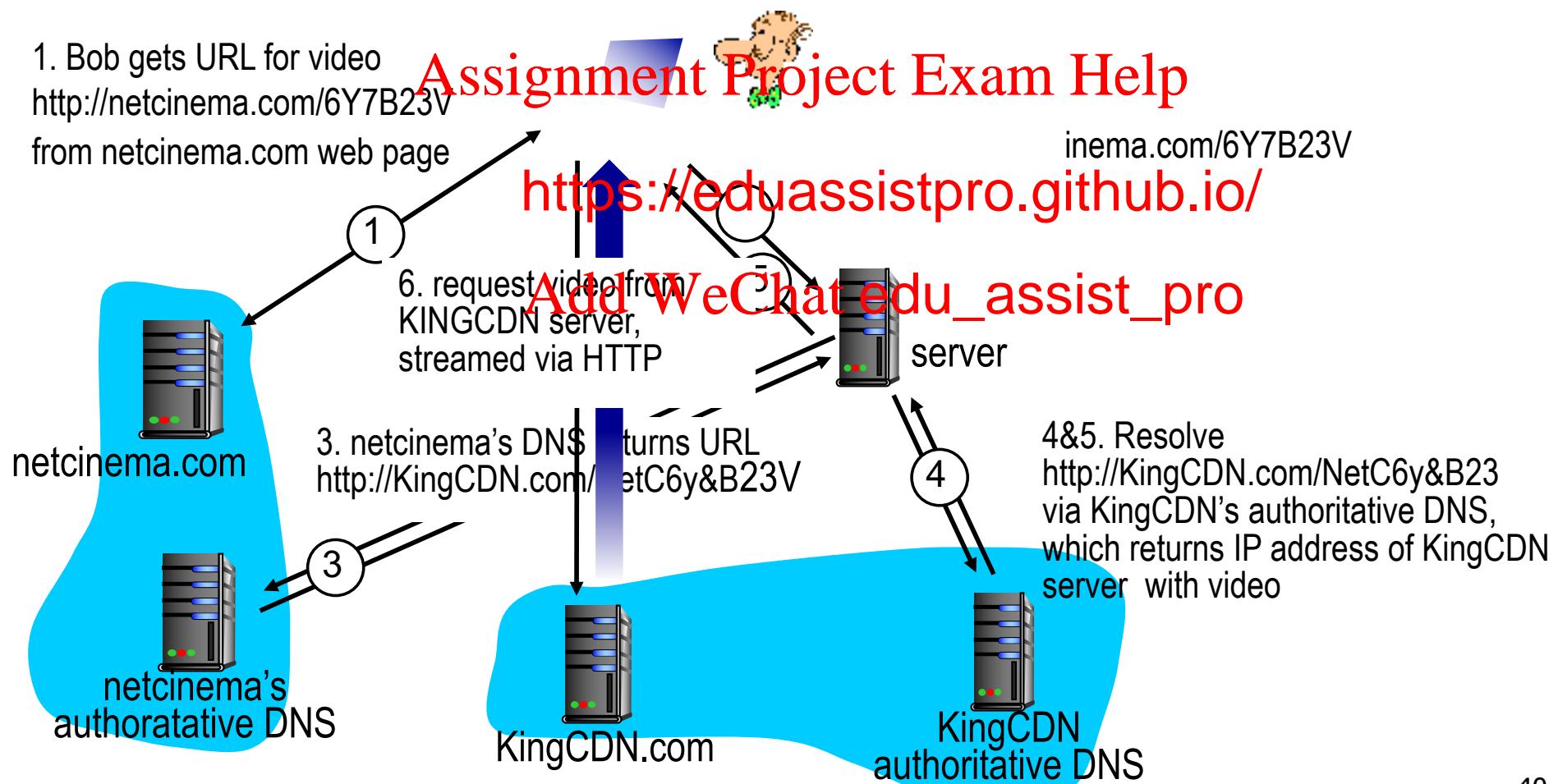
- from which CDN node to retrieve content?
- viewer behavior in presence of congestion?
- what content to place in which CDN node?

More later time permitting

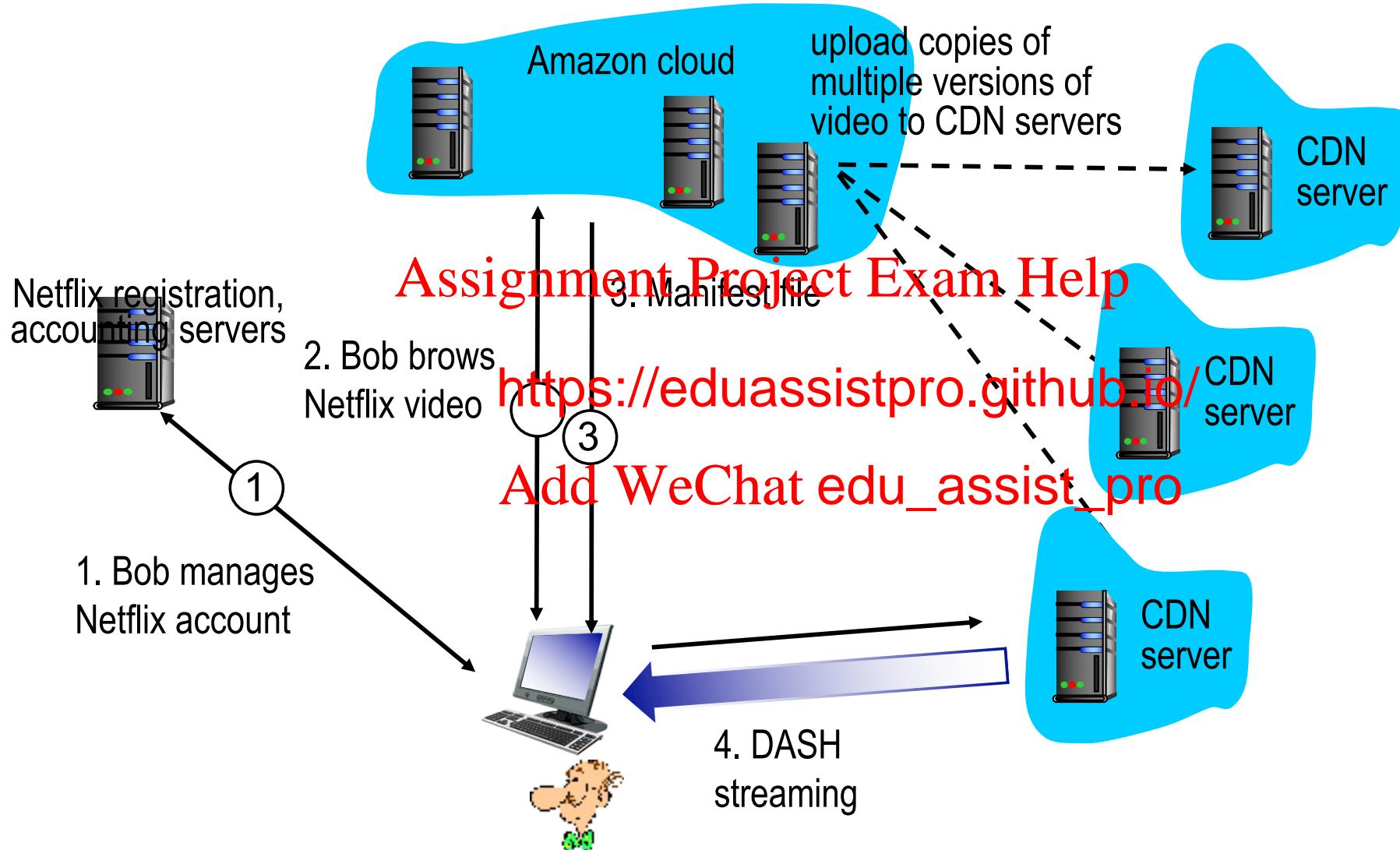
CDN content access: a closer look

Bob (client) requests video <http://netcinema.com/6Y7B23V>

- video stored in CDN at <http://KingCDN.com/NetC6y&B23V>



Case study: Netflix



Uses Push caching (during offpeak)
Preference to "deep inside" followed by "bring home"

NetFlix servers (snap shot from Jan 2018)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

Researchers from Queen Mary University of London (QMUL) traced server names that are sent to a user's computer every time they play content on Netflix to find the location of the 8492 servers (4152 ISP, 4340 IXP). They have been found to be scattered across 578 locations around the world.



Quiz: CDN

- ❖ The role of the CDN provider's authoritative DNS name server in a content distribution network, simply described, is:
 - a) to provide an alias address for each browser access to the “origin” site
 - b) to map the <https://eduassistpro.github.io/> server closest to the request to the CDN
 - c) to provide a mechanism for “proxies” to provide paths for clients (browsers)
 - d) none of the above, CDN networks do not use DNS

Transport Layer

our goals:

- ❖ understand principles behind transport layer services:
 - multiplexing, demultiplexing
 - reliable data transfer
 - flow control
 - congestion control
- ❖ learn about Internet transport layer protocols:
 - UDP: connectionless

<https://eduassistpro.github.io/>

ort
onn -oriented
Add WeChat edu_assist_pro

Transport Layer Outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

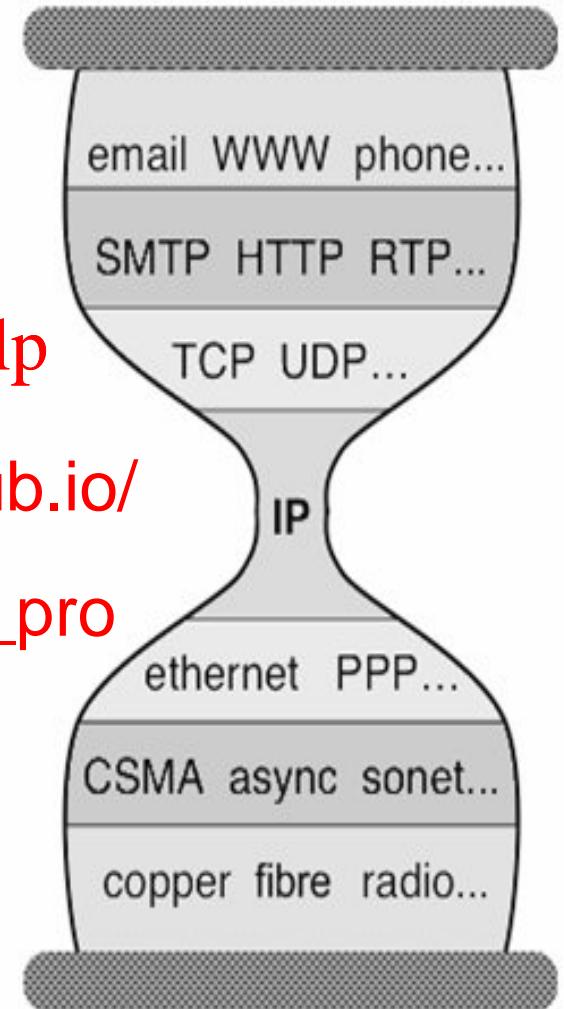
■ segment structure
able data transfer

https://eduassistpro.github.io/
Add WeChat edu_assist_pro
3.6 mechanisms of congestion control

3.7 TCP congestion control

Transport layer

- ❖ Moving “down” a layer
 - ❖ Current perspective:
 - Application is the boss....
 - Usually execute <https://eduassistpro.github.io/>
 - The network I
- Assignment Project Exam Help
d !!
Add WeChat edu_assist_pro

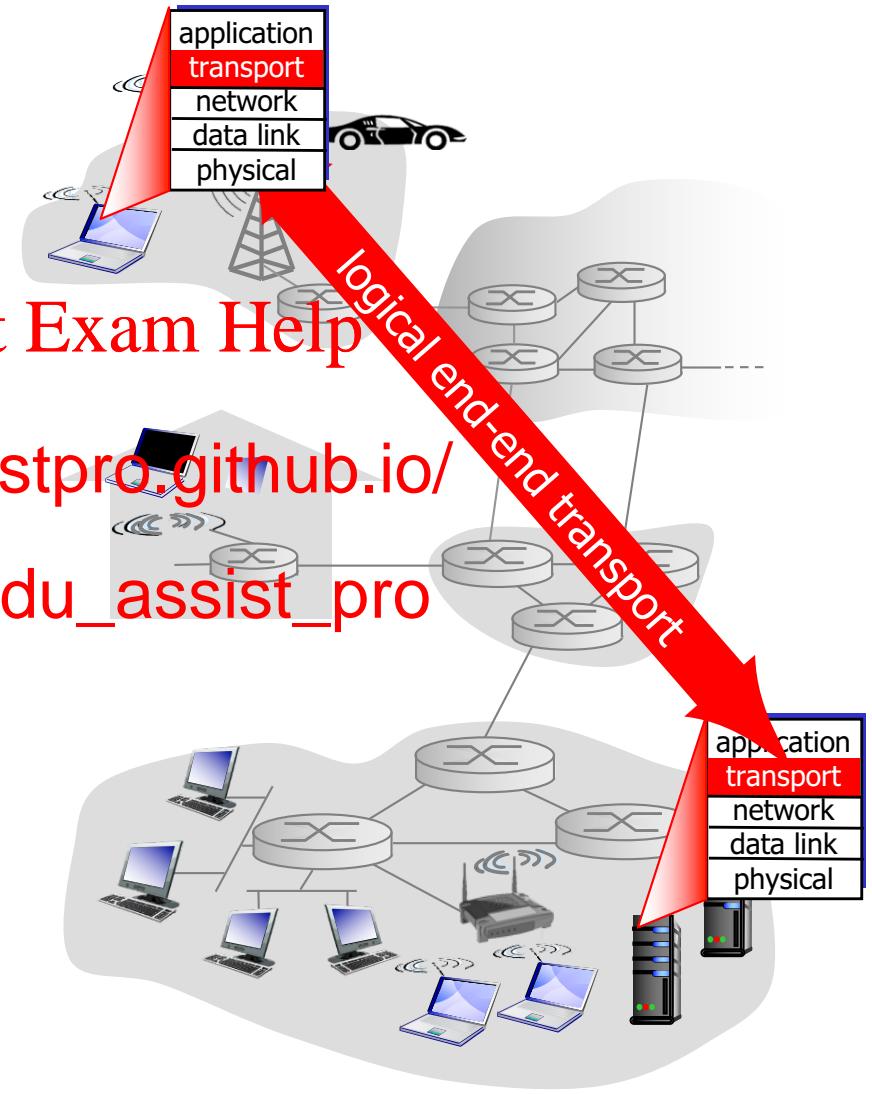


Network layer (context)

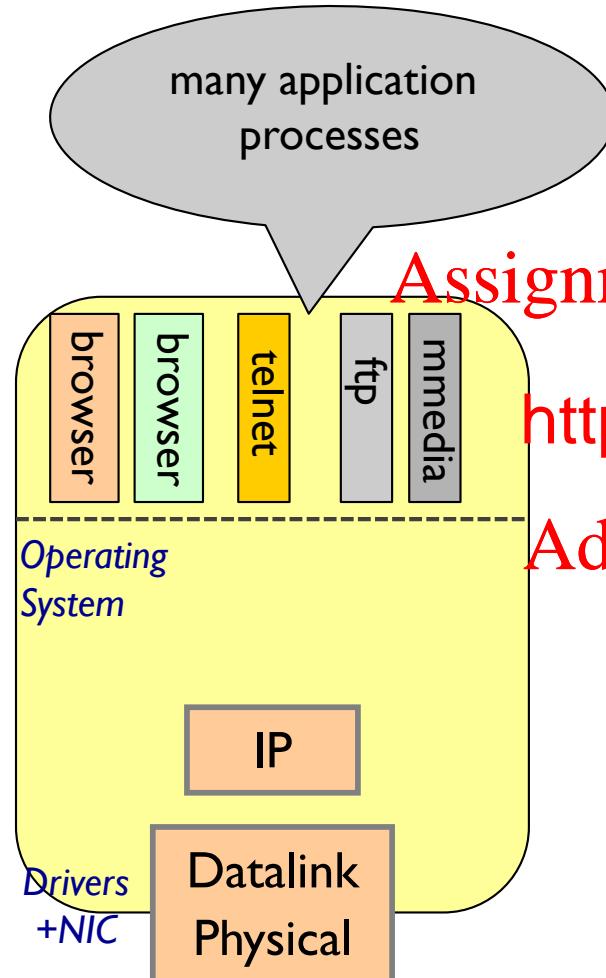
- ❖ What it does: finds paths through network
 - Routing from one end host to another
- ❖ What it doesn't:
 - Reliable transfer <https://eduassistpro.github.io/>
 - Guarantee paths
 - Arbitrate transfer rates
- ❖ For now, think of the network layer as giving us an “API” with one function:
sendtohost(data, host)
 - Promise: the data will go to that (usually!!)

Transport services and protocols

- ❖ provide *logical communication* between app processes running on different hosts
- ❖ transport protocol ~~Assignment Project Exam Help~~ *logical end-end transport* between end systems
 - send side: breaks messages into *segments*, passes to network layer
 - rcv side: reassembles segments into messages, passes to app layer
 - Exports services to application that network layer does not provide



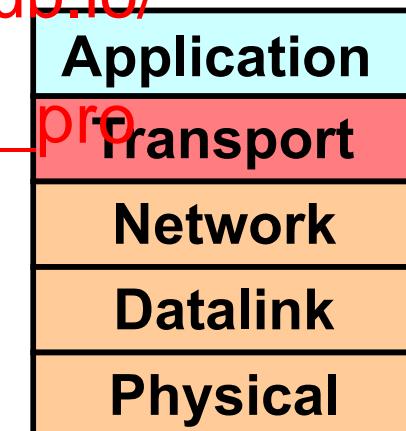
Why a transport layer?



Assignment Project Exam Help

<https://eduassistpro.github.io/>

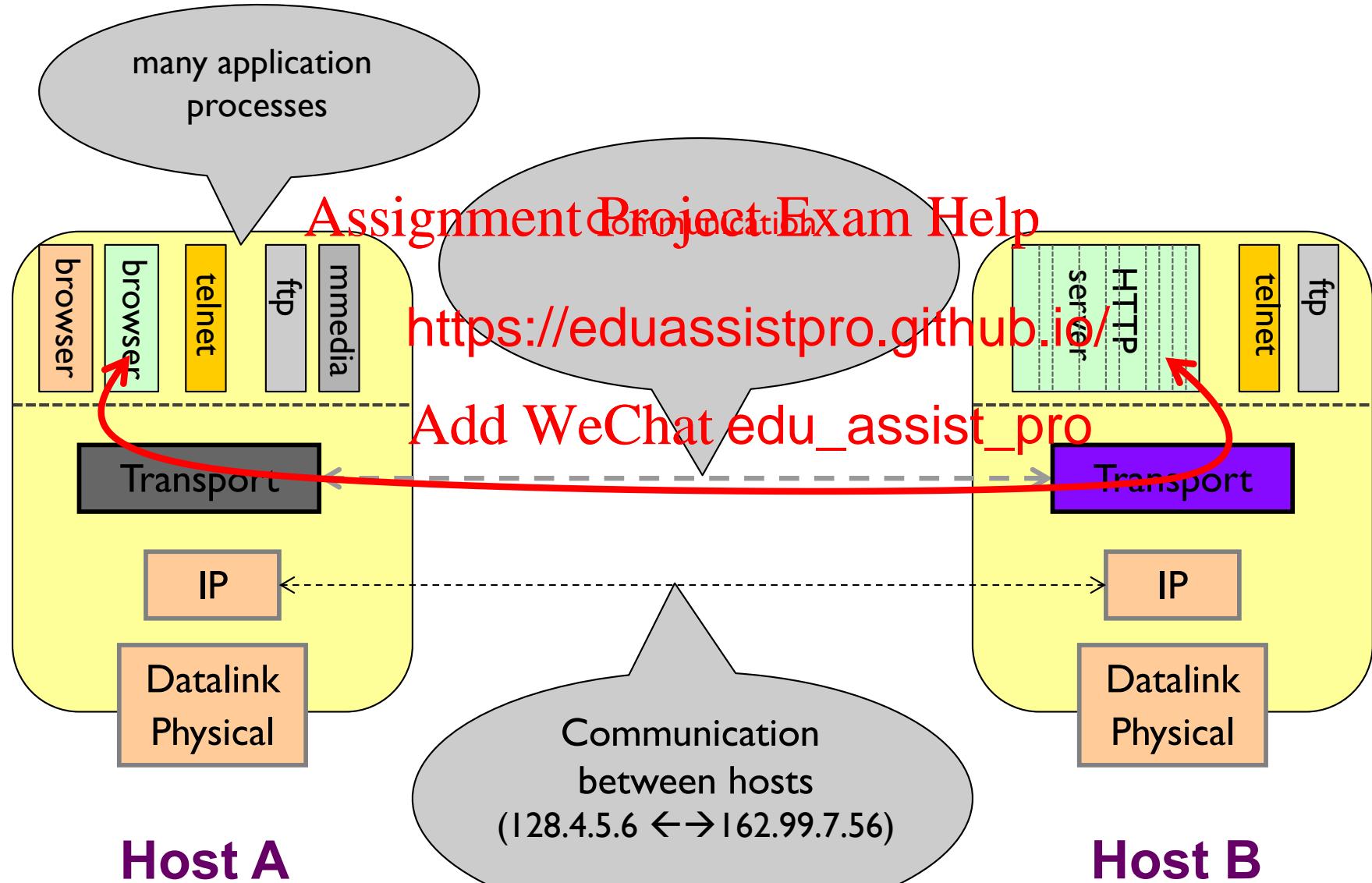
Add WeChat edu_assist_pro



Host A

Host B

Why a transport layer?



Transport Layer Outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

segment structure
able data transfer

connection management
es of congestion control

3.7 TCP congestion control

Multiplexing/demultiplexing

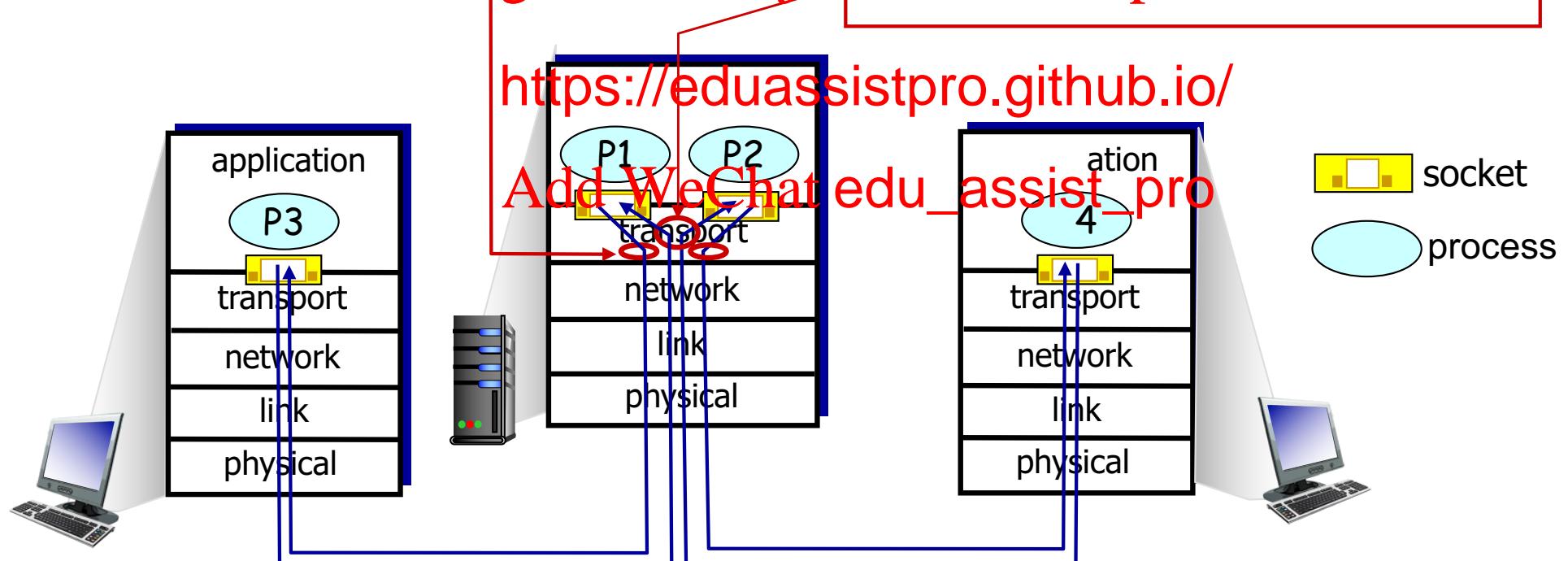
multiplexing at sender:

handle data from multiple sockets, add transport header (later used for demultiplexing)

demultiplexing at receiver:

use header info to deliver received segments to correct socket

Assignment Project Exam Help



Note: The network is a shared resource. It does not care about your applications, sockets, etc.

Connectionless demultiplexing

- ❖ *recall:* created socket has host-local port #:

```
DatagramSocket mySocket1  
= new DatagramSocket(1234);
```

- ❖ *recall:* when creating datagram to send into UDP socket, must specify destination IP address

<https://eduassistpro.github.io/>

- ❖ when host receives UDP segment:

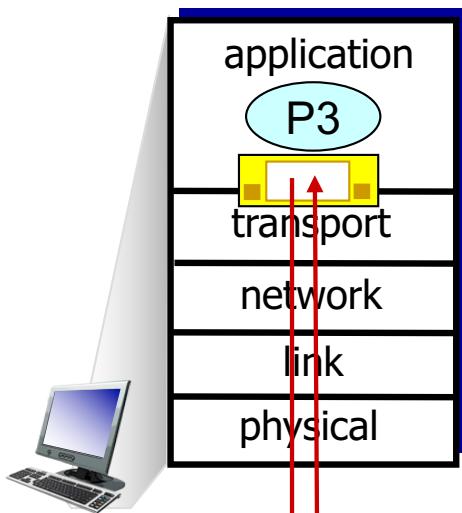
- checks destination port # in segment
- directs UDP segment to socket with that port #



Add WeChat edu_assist_pro
grams with *same*
dest. port #, but different
source IP addresses
and/or source port
numbers will be directed
to *same socket* at dest

Connectionless demux: example

```
DatagramSocket  
mySocket2 = new  
DatagramSocket  
(9157);
```



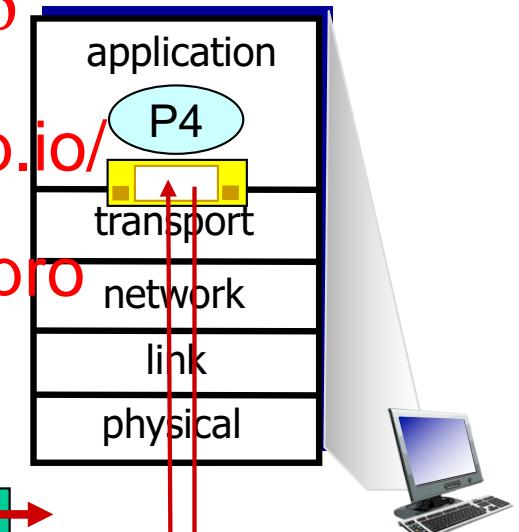
```
DatagramSocket  
serverSocket = new  
DatagramSocket  
(6428);
```

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

```
DatagramSocket  
mySocket1 = new  
DatagramSocket  
(5775);
```



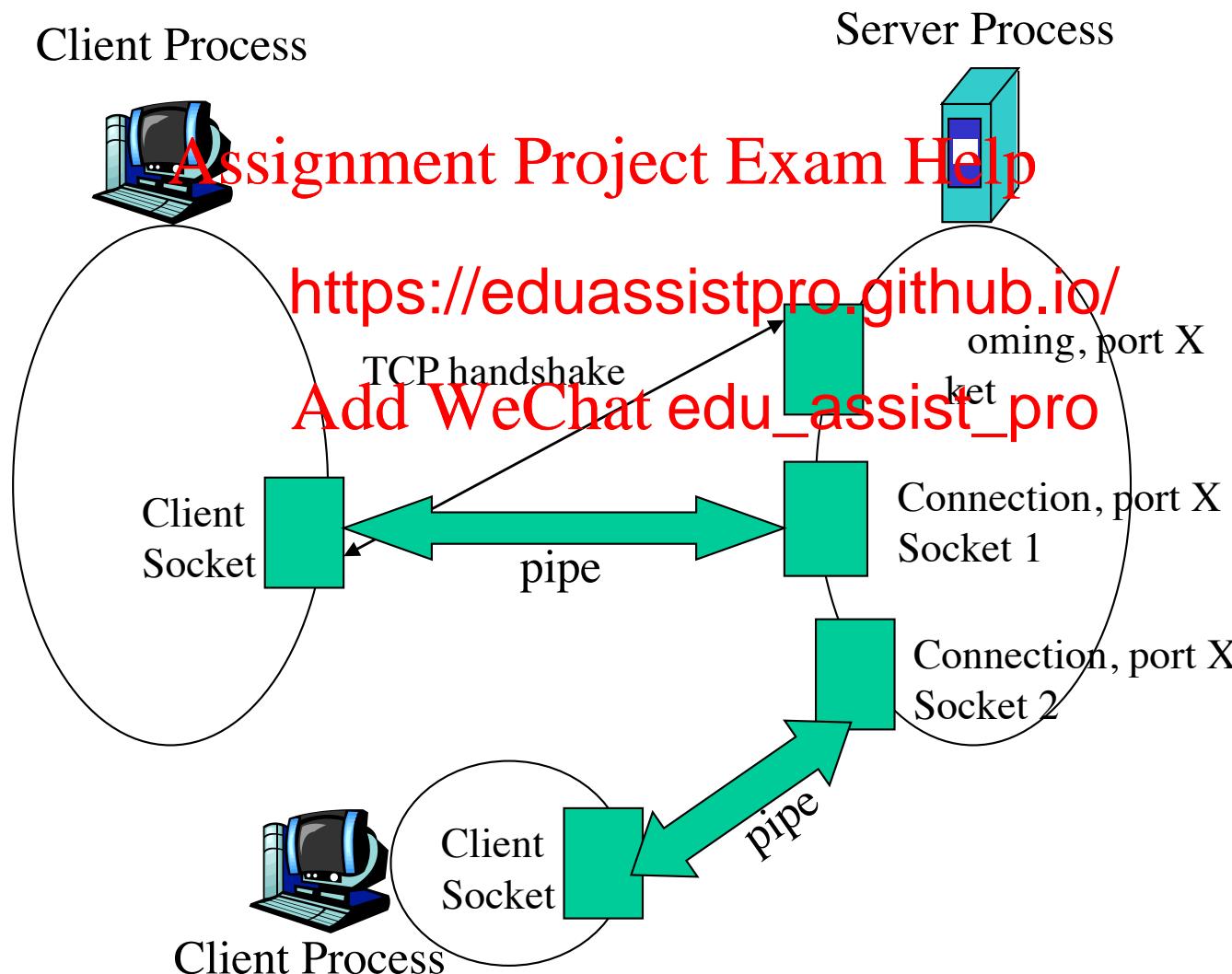
source port: ?
dest port: ?

source port: 9157
dest port: 6428

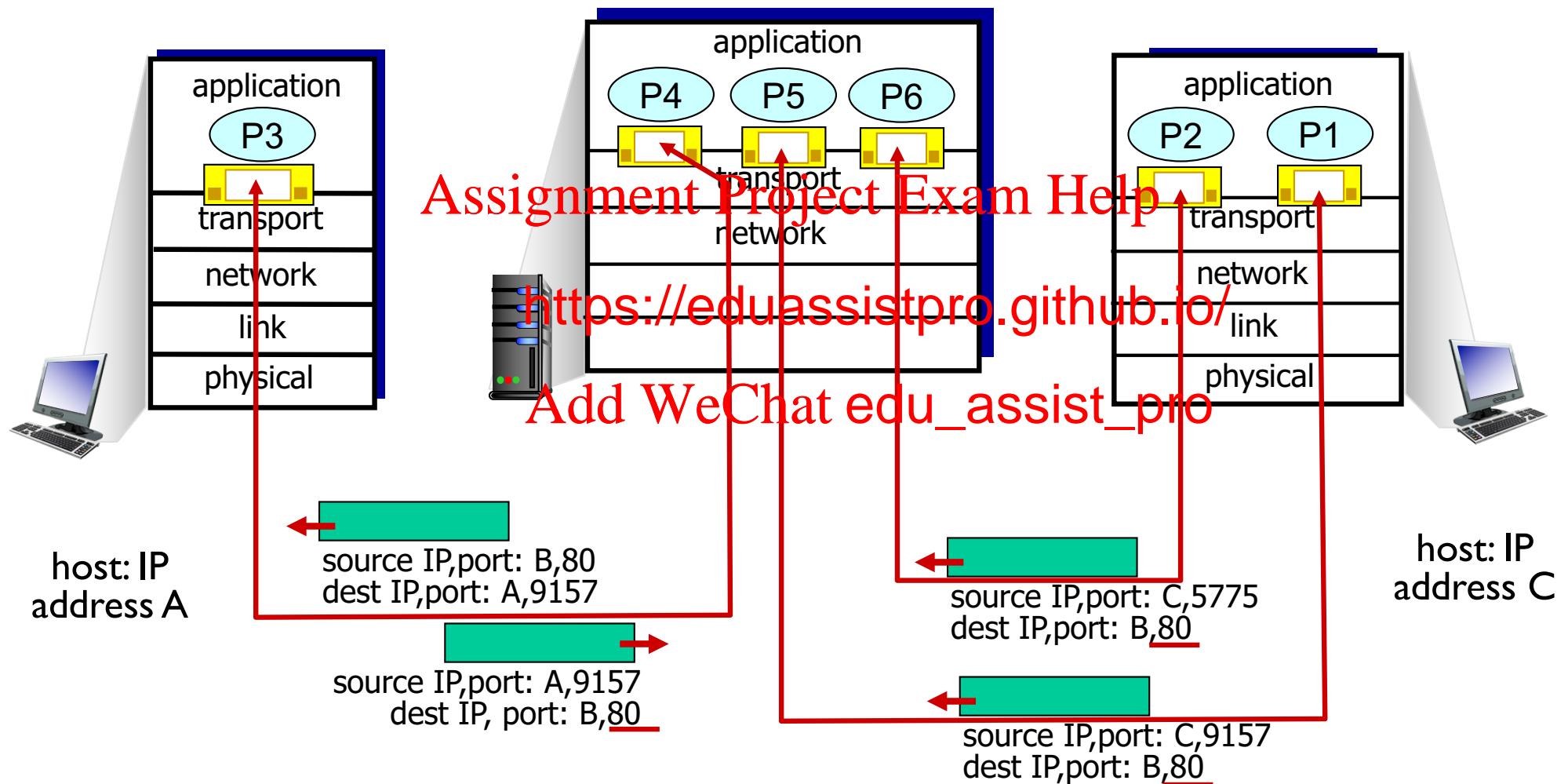
Connection-oriented demux

- ❖ TCP socket identified by 4-tuple:
 - source IP address
 - source port number
 - dest IP address <https://eduassistpro.github.io/>
 - dest port number
- ❖ demux: receiver uses all four values to direct segment to appropriate socket
- ❖ server host may support many simultaneous TCP sockets:
 - each socket identified by a 4-tuple
 - ❖ servers
 - non-persistent HTTP will have different socket for each request

Revisiting TCP Sockets



Connection-oriented demux: example



three segments, all destined to IP address: B,
dest port: 80 are demultiplexed to *different* sockets

Quiz: Sockets



- ❖ Suppose that a Web Server runs in Host C on port 80. Suppose this server uses persistent connections, and is currently receiving requests from two different Hosts, A and B.

<https://eduassistpro.github.io/>

- Are all of the requests being passed through the same socket at host C ?
- If they are being passed through different sockets, do both of the sockets have port 80?

May I scan your ports?

<http://netsecurity.about.com/cs/hackertools/a/aa121303.htm>

- ❖ Servers wait at open ports for client requests
- ❖ Hackers often perform *port scans* to determine open, closed and unreachable ports on candidate victims
- ❖ Several ports are well-known
 - <1024 are reserved <https://eduassistpro.github.io/>
 - Other apps also
 - MS SQL server uses port 1433
 - Sun Network File System (NFS) 2049 (tcp/udp)
- ❖ Hackers can exploit known flaws with these known apps
 - Example: Slammer worm exploited buffer overflow flaw in the SQL server
- ❖ How do you scan ports?
 - Nmap, Superscan, etc

<http://www.auditmypc.com/>

<https://www.grc.com/shieldsup>

Transport Layer Outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

■ segment structure
able data transfer

3.6 principles of congestion control

3.7 TCP congestion control

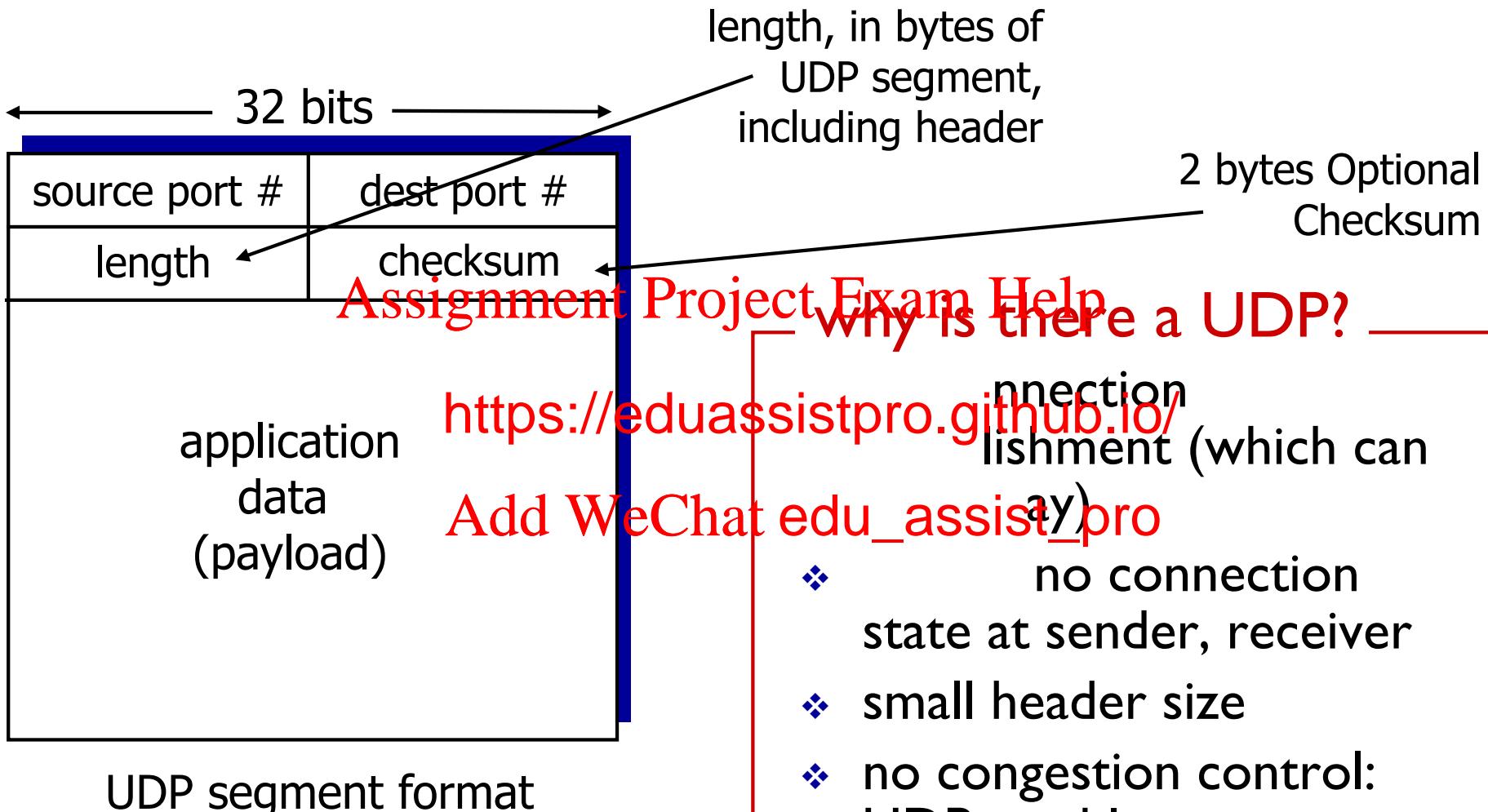
Assignment Project Exam Help

Add WeChat edu_assist_pro

UDP: User Datagram Protocol [RFC 768]

- ❖ “no frills,” “bare bones” Internet transport protocol
- ❖ “best effort” service, UDP segments may be:
 - lost
 - delivered out-of-order to app
- ❖ *connectionless:* <https://eduassistpro.github.io/>
 - no handshaking between UDP ceiver
 - each UDP segment handled i ly of others

UDP: segment header



- ❖ no connection state at sender, receiver
- ❖ small header size
- ❖ no congestion control: UDP can blast away as fast as desired

UDP checksum

- **Goal:** detect “errors” (e.g., flipped bits) in transmitted segment
 - Router memory errors
 - Driver bugs
 - Electromag

sender:

<https://eduassistpro.github.io/>

- ❖ treat segment contents, including header fields, as sequence of 16-bit integers
 - ❖ checksum: addition (one's complement sum) of segment contents
 - ❖ sender puts checksum value into UDP checksum field
- Add WeChat `edu_assist_pro` to receive the received r as 16-bit integers
- ❖ Add that to the checksum
 - ❖ If the result is not 1111 1111 1111, there are errors !

UDP: Checksum

- Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the UDP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.
- Checksum header
- But the header contains the pseudo-header?
- What's IP pseudo-header?

Assignment Project Exam Help

P pseudo-header
<https://eduassistpro.github.io/>

elf?

Add WeChat edu_assist_pro

UDP header

bits	0 – 7	8 – 15	16 – 23	24 – 31
0	Source address			
32	Destination address			
64	Zeros	Protocol	UDP length	
96	Source Port		Destination Port	
128	Length		Checksum	
160+	Data			

{ IP pseduo-header

{ UDP payload
UDP 64

Internet checksum: example

example: add two 16-bit integers

	1	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
<hr/>																
wraparound	1	1	0	https://eduassistpro.github.io/	1	1	1	0	1	1	0	1	1	0	1	1
sum	1	0	1	1	1	0	1	1	1	1	1	0	0	0	1	1
checksum	0	1	0	0	0	1	0	0	0	1	0	0	0	0	1	1

Note: when adding numbers, a carryout from the most significant bit needs to be added to the result

4500 003C 1C46 4000 4006 B1E6 AC10 0A63 AC10 0A0C

4500 -> 0100010100000000

003c -> 0000000000111100

453C -> 0100010100111100

453C -> 0100010100111100

1c46 -> 0001110001000110

6182 -> 0110000110000010

4500 -> 0100010100000000

003C -> 0000000000111100

1C46 -> 0001110001000110

4000 -> 0100000000000000

4006 -> 0100000000000110

0000 -> 0000000000000000

AC10 -> 1010110000010000

0A63 -> 0000101001100011

AC10 -> 1010110000010000

0A0C -> 0000101000001100

6182 -> 0110000110000010

4000 -> 0100000000000000

A182 -> 1010000110000010

97FC -> 100101111111100

AC10 -> 1010110000010000

1440C -> 10100010000001100

1440C -> 10100010000001100

1440C -> 10100010000001101

400D -> 0100010000001101

A0C -> 0000101000001100

4E19 -> 0100111000011001

B1E6 -> 1011000111100110

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add-WeChat edu_assist_pro

E188 -> 1110000110001000

AC10 -> 1010110000010000

18D98 -> 1100011011001100

18D98 -> 1100011011001100

8D99 -> 1000110110011001

8D99 -> 1000110110011001

0A63 -> 0000101001100011

97FC -> 100101111111100

UDP Applications

- ❖ Latency sensitive/time critical
 - ❖ Quick request/response (DNS, DHCP)
 - ❖ Network management (SNMP)
 - ❖ Routing update
 - ❖ Voice/video communication
 - ❖ Gaming (especially FPS)
- ❖ Error correction unnecessary (periodic messages)

Assignment Project Exam Help

Add WeChat edu_assist_pro

Transport Layer Outline

3.1 transport-layer services

3.2 multiplexing and demultiplexing

3.3 connectionless transport: UDP

3.4 principles of reliable data transfer

3.5 connection-oriented transport: TCP

■ segment structure
able data transfer

https://eduassistpro.github.io/
Add WeChat edu_assist_pro
3.6 mechanisms of congestion control

3.7 TCP congestion control

Reliable Transport

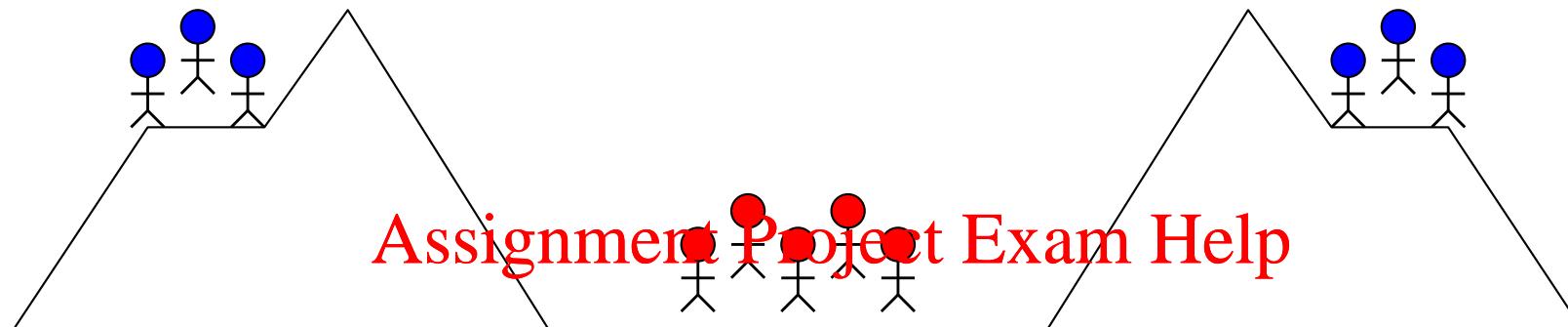
- In a perfect world, reliable transport is easy
- All the bad things best-effort can do
 - a packet is lost
 - a packet is delayed (*why?*)
 - packets are reordered (*why?*)
 - a packet is duplicated (*why?*)

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat
edu_assist_pro

The Two Generals Problem

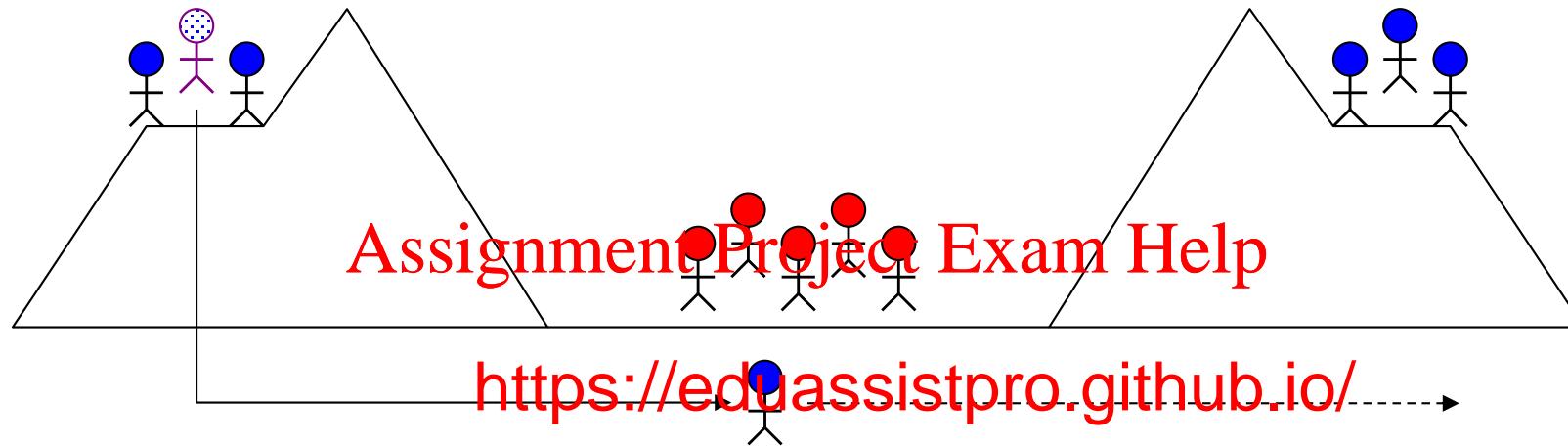


Assignment Project Exam Help

<https://eduassistpro.github.io/>

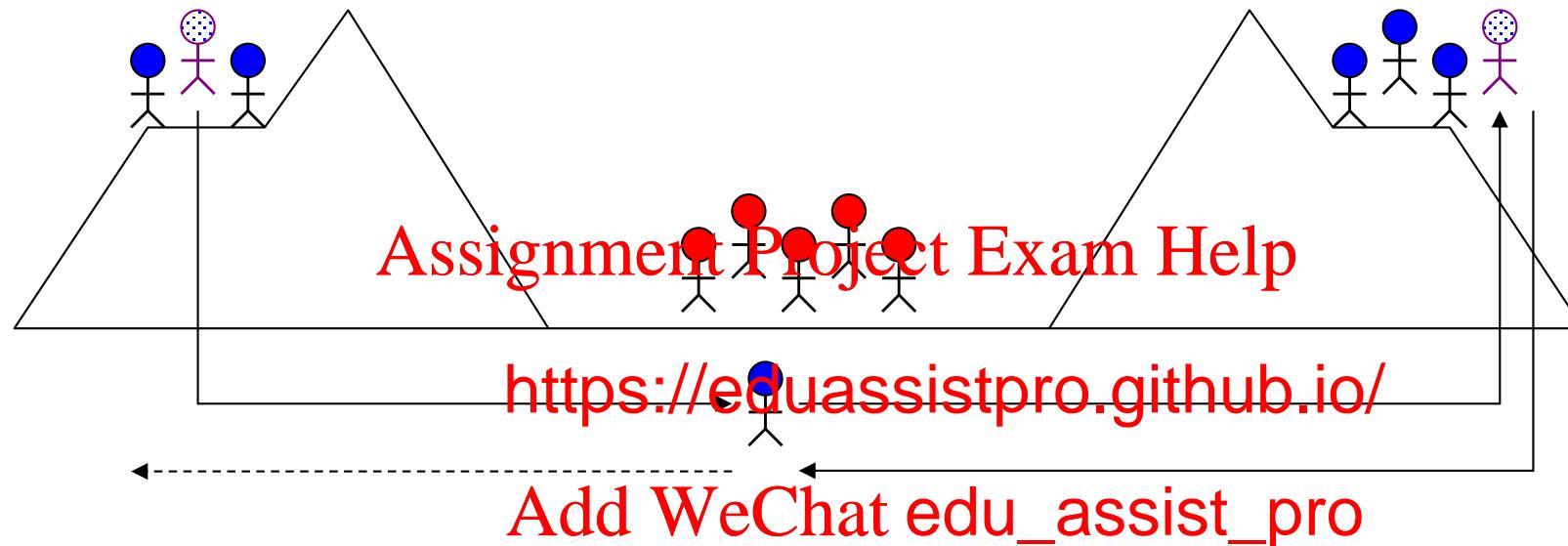
- ❖ Two army divisions
 - Each division led by a general
 - Both must agree when to simultaneously attack
 - If either side attacks alone, defeat
- ❖ Generals can only communicate via messengers
 - Messengers may get captured (unreliable channel)

The Two Generals Problem



- ❖ How to coordinate?
 - Send messenger: “Attack at dawn”
 - What if messenger doesn’t make it?

The Two Generals Problem



- ❖ How to be sure messenger made it?
 - Send acknowledgement: “We received message”

Principles of reliable data transfer

- ❖ important in application, transport, link layers
 - top-10 list of important networking topics!

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat No bits are lost or arrive out-of-order ever `edu_assist_pro`

- ❖ characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principles of reliable data transfer

- ❖ important in application, transport, link layers
 - top-10 list of important networking topics!

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat No bits are lost or arrive out-of-order ever `edu_assist_pro`

- ❖ characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Principles of reliable data transfer

- ❖ important in application, transport, link layers
 - top-10 list of important networking topics!

Assignment Project Exam Help

<https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

- ❖ characteristics of unreliable channel will determine complexity of reliable data transfer protocol (rdt)

Reliable data transfer: getting started

We'll:

- Incrementally develop sender, receiver sides of reliable data transfer protocol (rdt)
- Consider only unidirectional data transfer
 - but control info will flow on both directions!
- Channel will be <https://eduassistpro.github.io/>

Add WeChat edu_assist_pro

rdt1.0: reliable transfer over a reliable channel

- Underlying channel perfectly reliable
 - no bit errors
 - no loss of packets
- Transport Layer Assignment Project Exam Help
 - <https://eduassistpro.github.io/>
 - Add WeChat edu_assist_pro

rdt2.0: channel with bit errors

- ❖ underlying channel may flip bits in packet
 - checksum to detect bit errors
- ❖ *the question: how to recover from errors:*

Assignment Project Exam Help

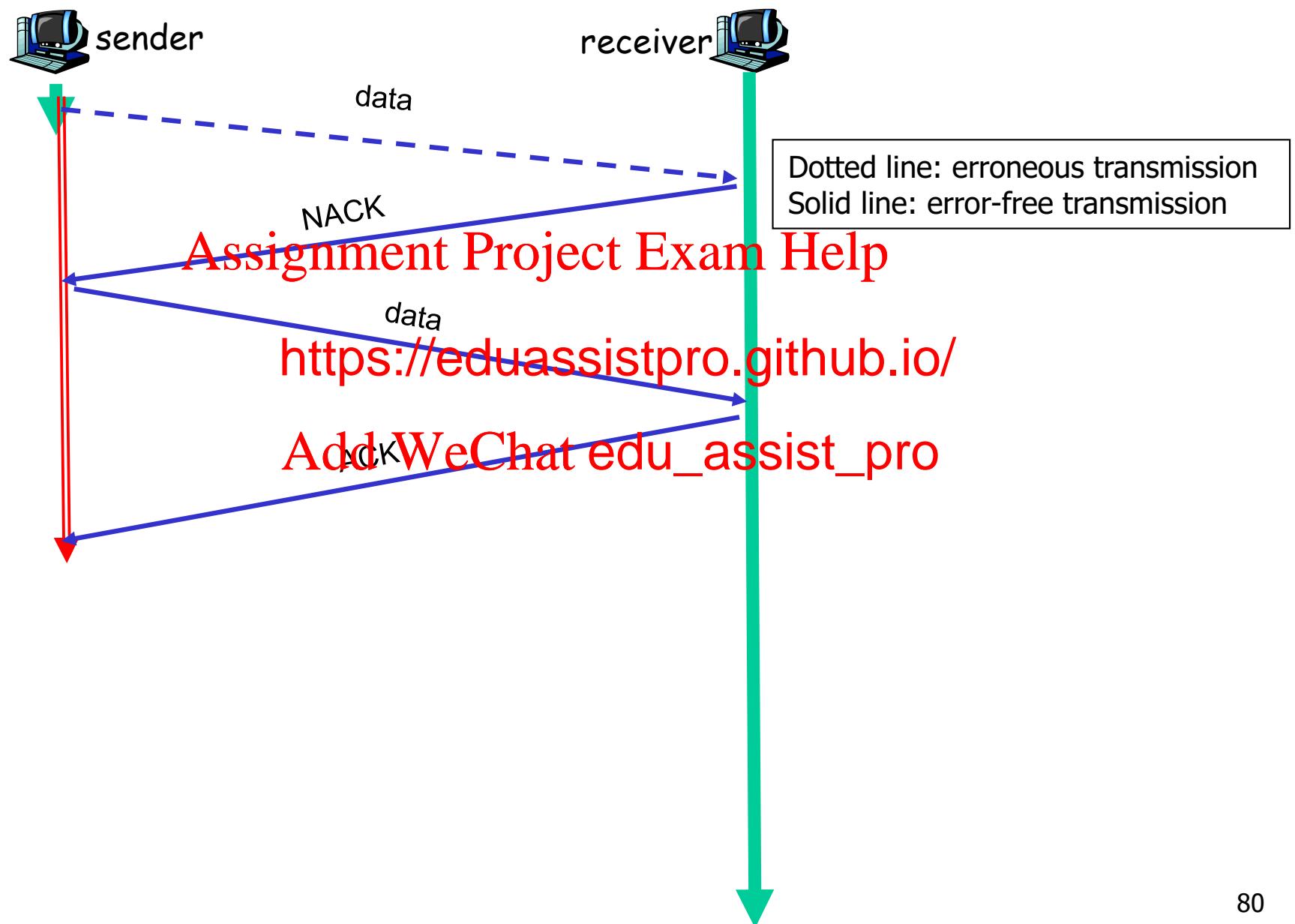
<https://eduassistpro.github.io/>

How do Add WeChat edu_assist_pro humans recover errors during conversation?

rdt2.0: channel with bit errors

- ❖ underlying channel may flip bits in packet
 - checksum to detect bit errors
- ❖ *the question: how to recover from errors:*
 - *acknowledgments ACK* receiver explicitly tells sender that pkt received
 - *negative ackn* receiver explicitly tells sender that pkt had errors
 - sender retransmits pkt on receipt of ACK
- ❖ new mechanisms in rdt2.0 (beyond rdt1.0):
 - error detection
 - feedback: control msgs (ACK,NAK) from receiver to sender
 - retransmission

Global Picture of rdt2.0



Transport Part I: Summary

- ❖ principles behind transport layer services:
 - multiplexing, demultiplexing
 - UDP
 - reliable dat

- ❖ **Next Week:**
 - rdt (continued)
 - Pipelined Protocols for reliable data transfer
 - TCP

<https://eduassistpro.github.io/>
Assignment Project Exam Help
TCP Flow Control
ection
Add WeChat edu_assist_pro