Advanced Network Technologies

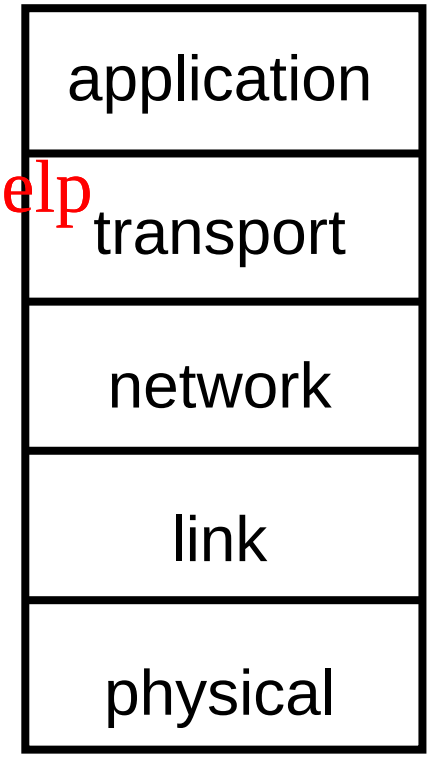
Review

Dr. Wei Bao | Lecturer
School of Computer Science

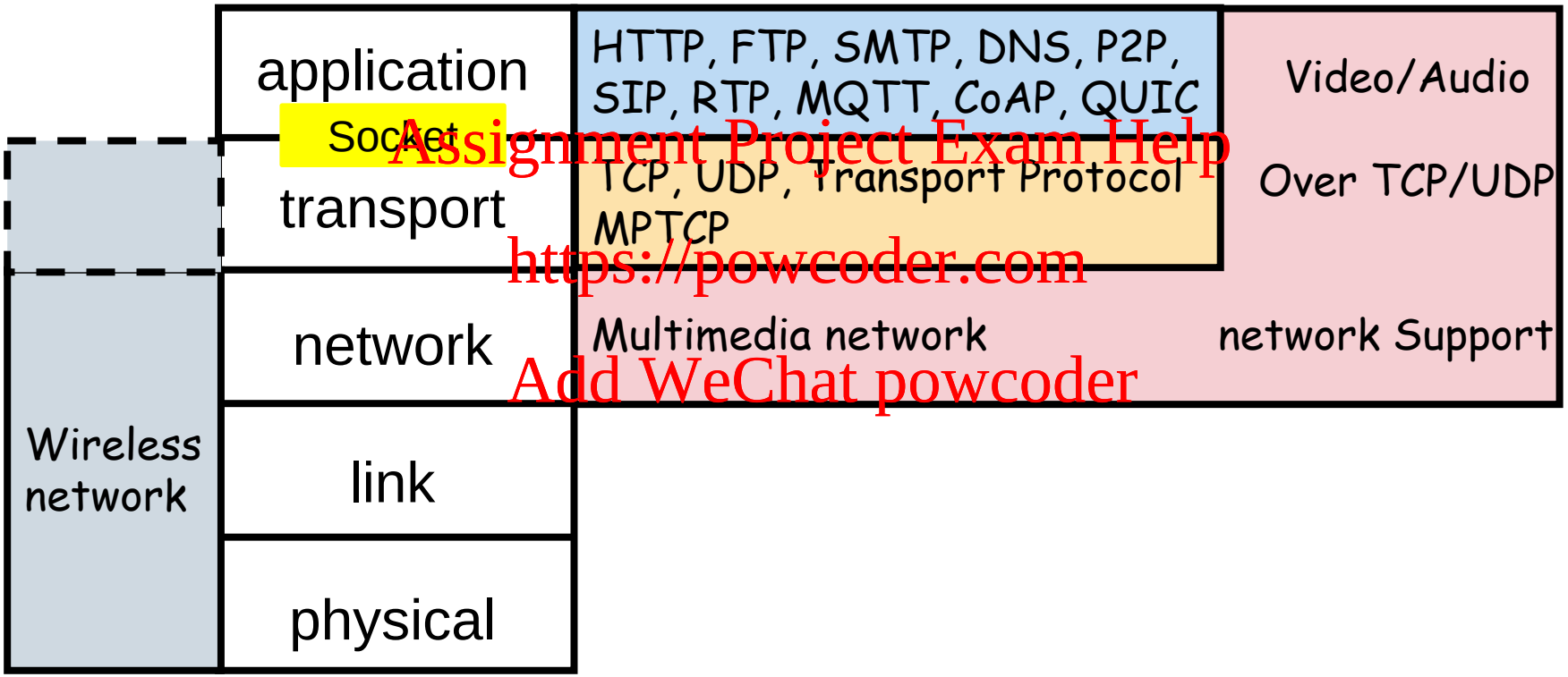
THE UNIVERSITY OF SYDNEY

Protocol Stack

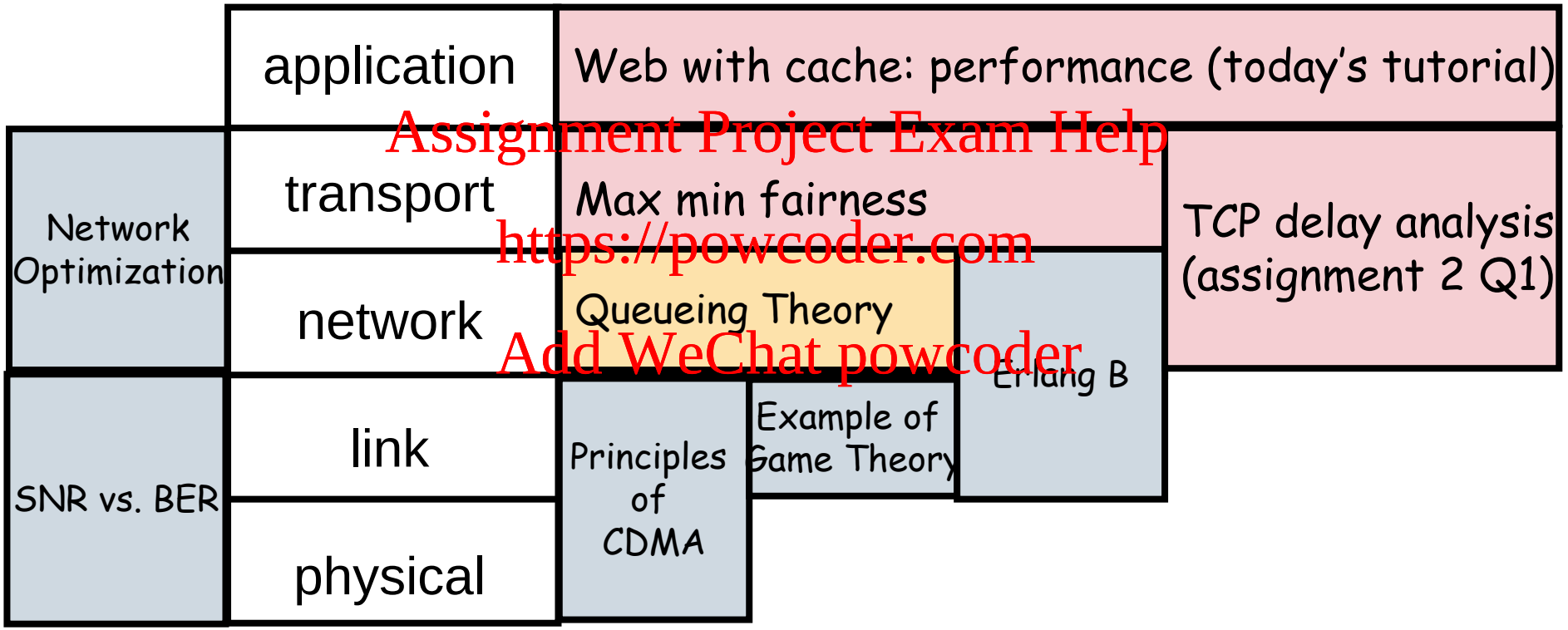
- *application:* supporting network applications
  - FTP, SMTP, HTTP
- *transport:* process-process data transfer
  - TCP, UDP
- *network:* routing of datagrams from source to destination
  - IP, routing protocols
- *link:* data transfer between neighboring network elements
  - Ethernet, 802.11 (WiFi)
- *physical:* bits "on the wire"



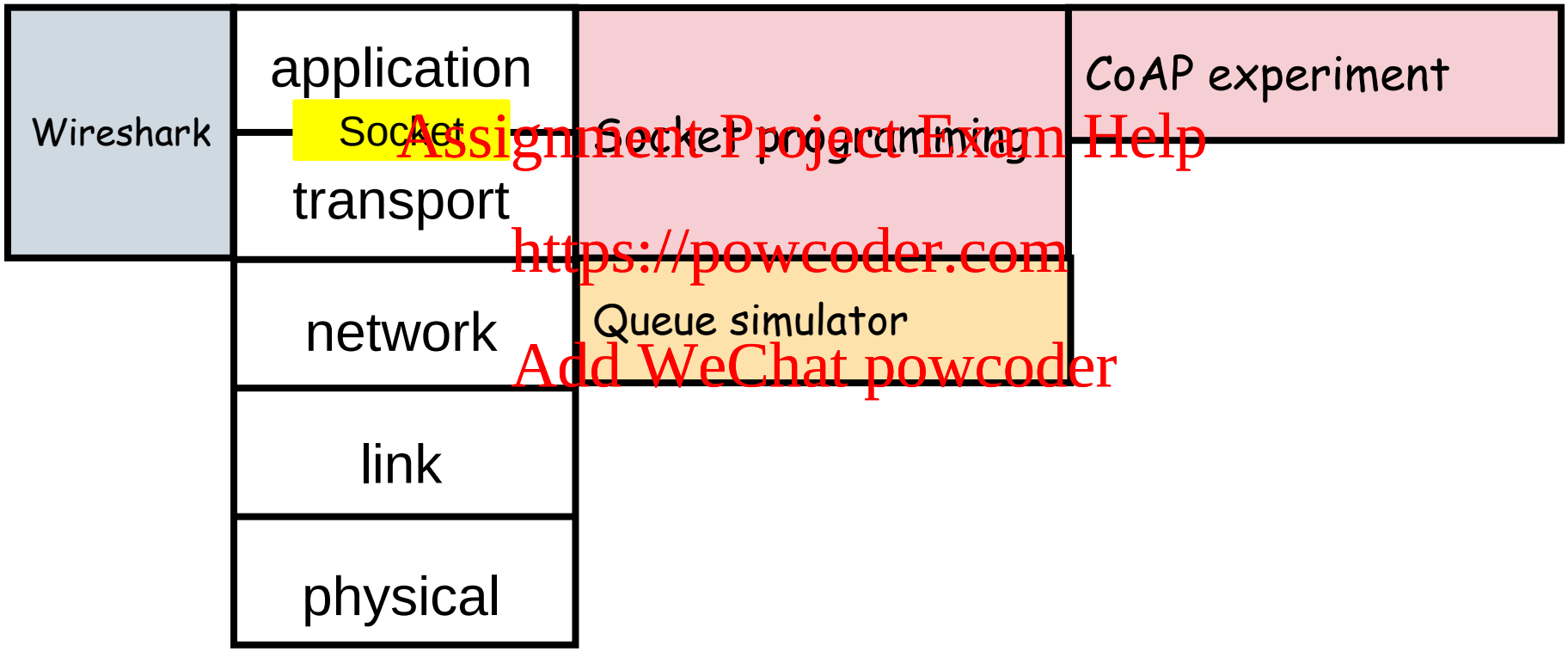
Internet Protocol Stack: Practice



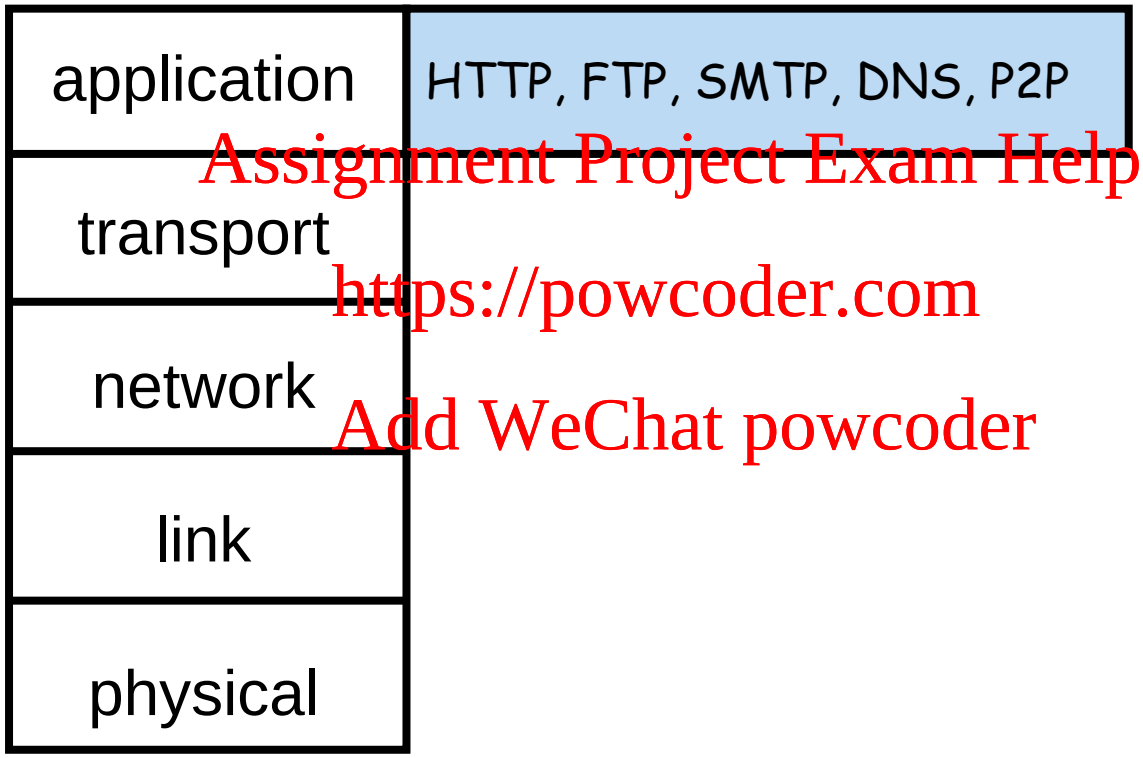
Internet Protocol Stack: Theory



Internet Protocol Stack: Programming/Experiment



Internet Protocol Stack



HTTP: hypertext transfer protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests

HTTP connections

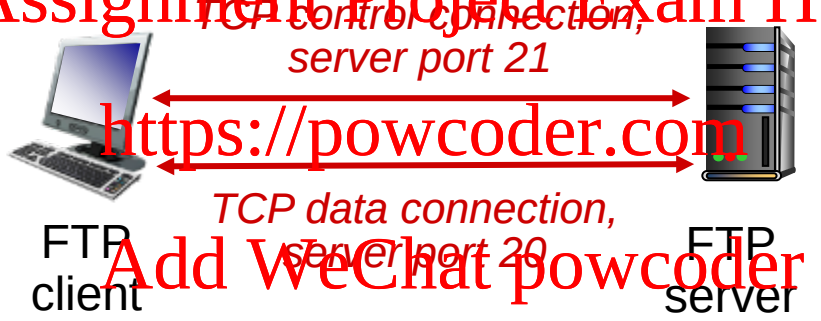
non-persistent HTTP

- at most one object sent over TCP connection
  - connection then closed
- downloading multiple objects required multiple connections

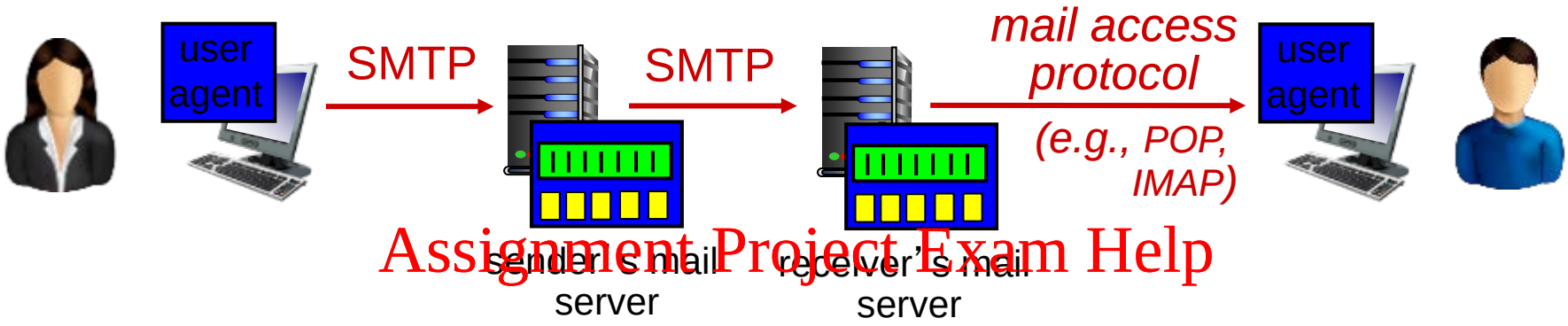
persistent HTTP

- multiple objects can be sent over single TCP connection between client, server

FTP: the file transfer protocol



Email

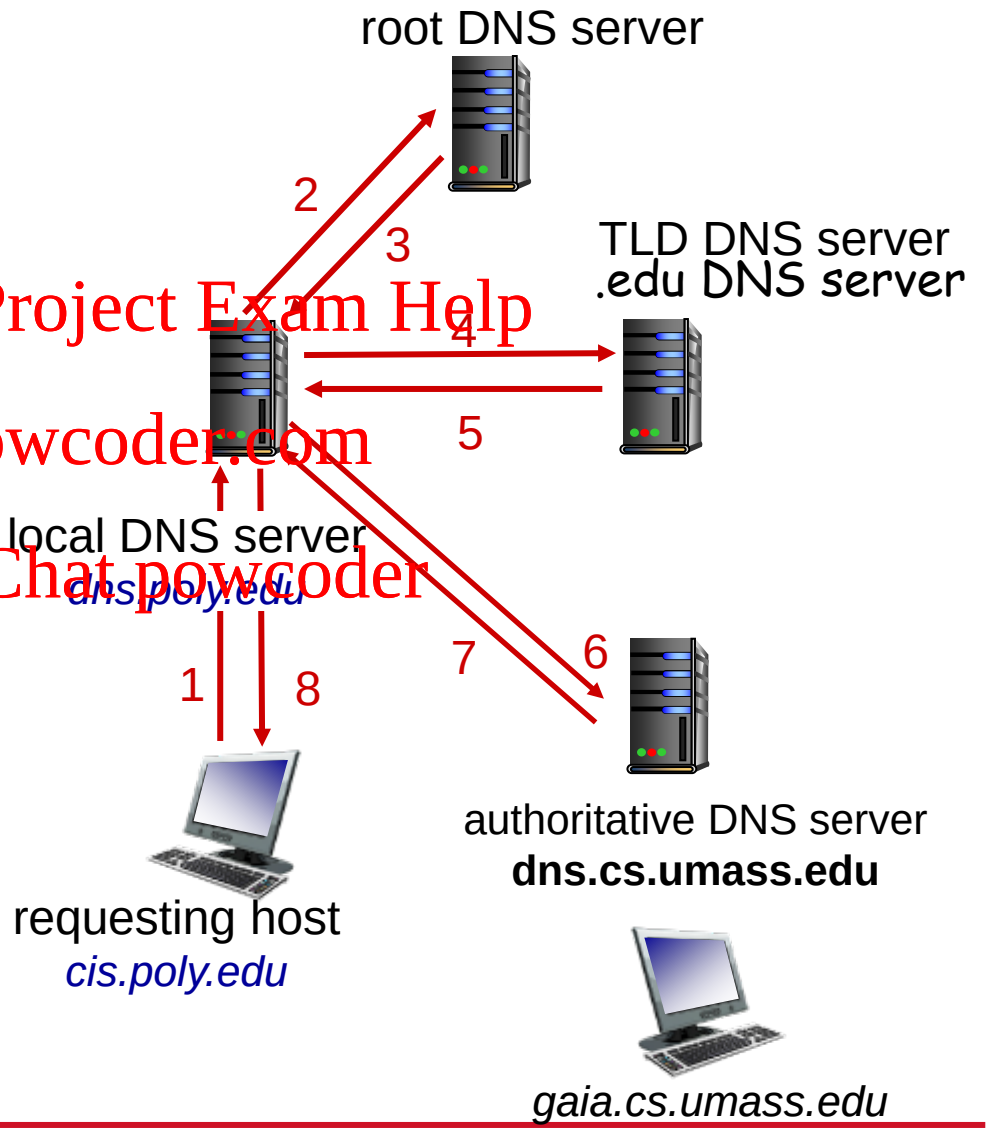


- SMTP: delivery/storage to receiver's server
- mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored msgs on server
  - HTTP: Using a browser to access a webmail `https://webmail.sydney.edu.au`

- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

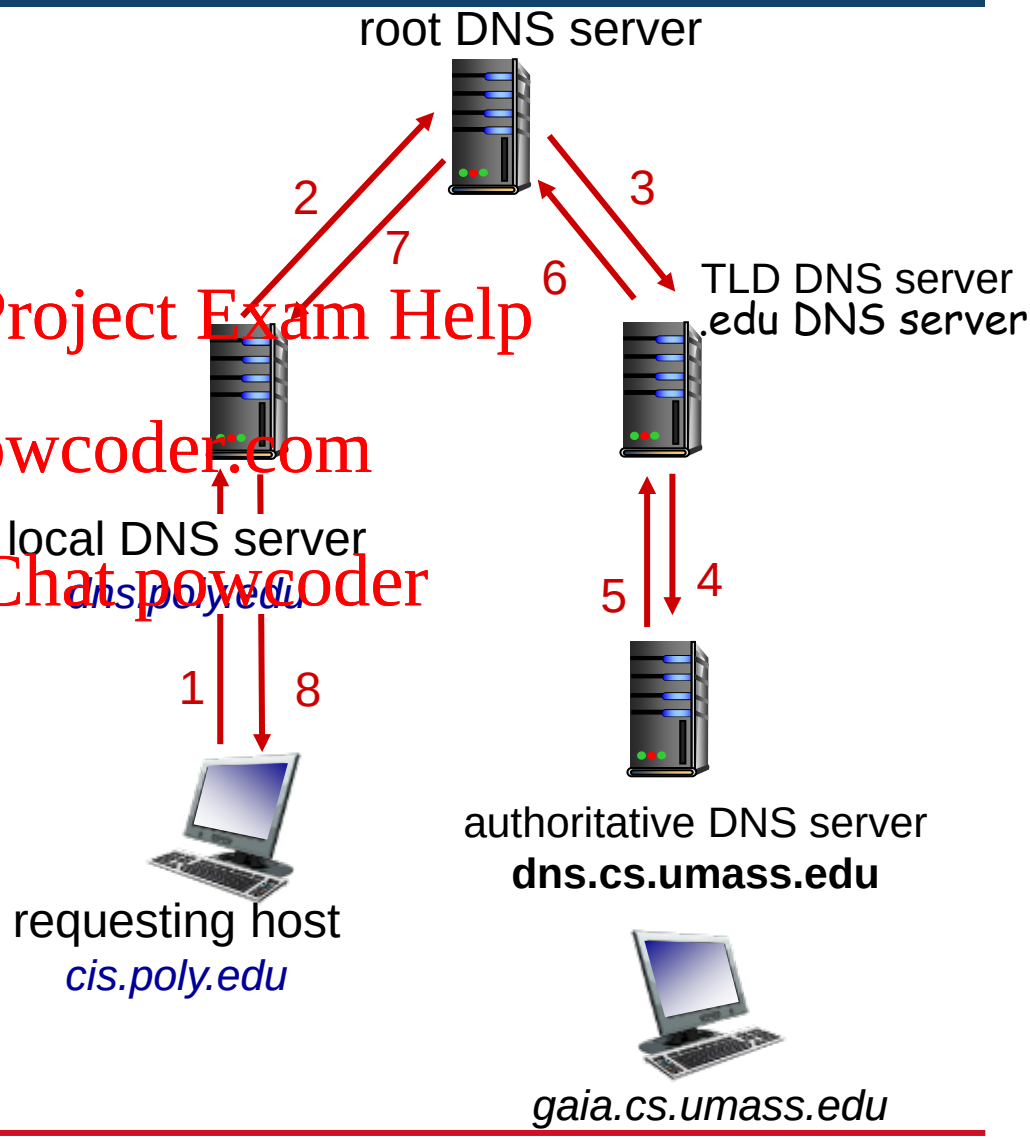
iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"

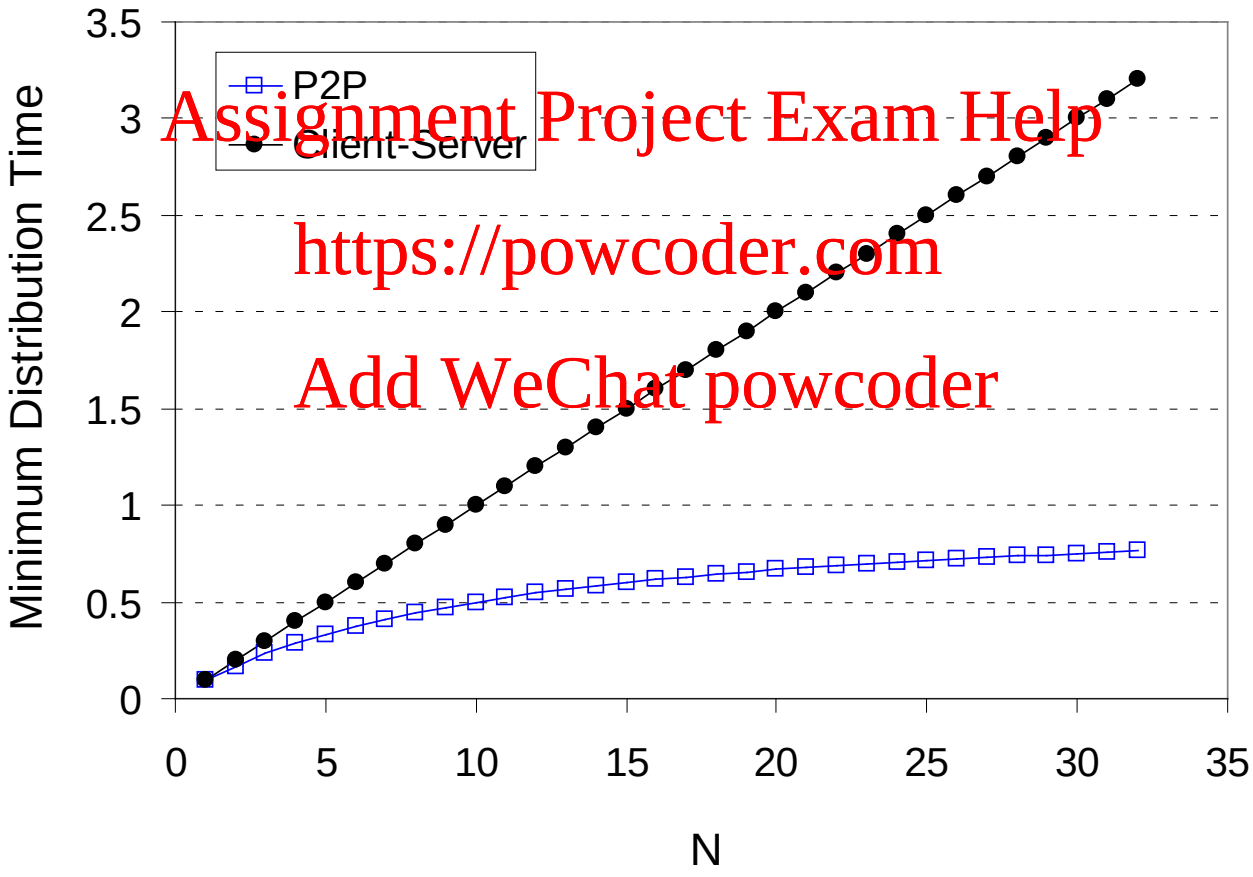


recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



P2P Principle



BitTorrent: tit-for-tat

(1) Alice sends chunks to those four peers currently sending her chunks at highest rate

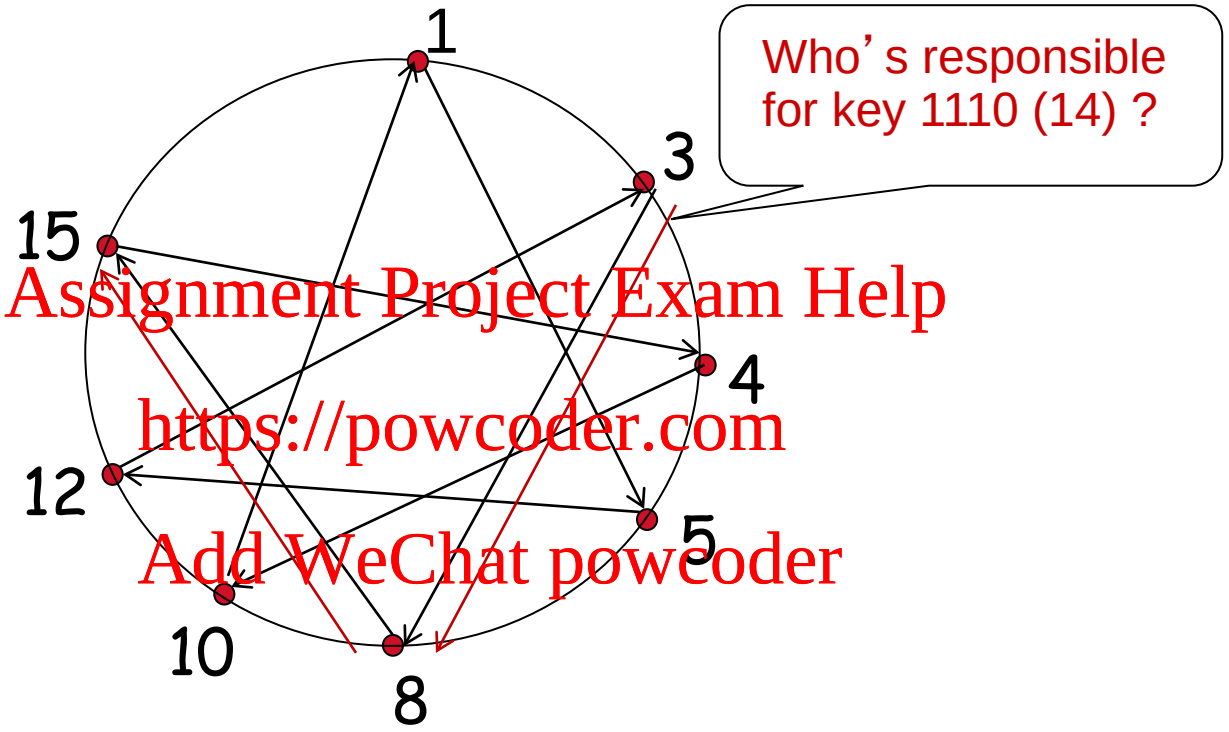
(2) Alice randomly unchokes Bob

(3) Alice becomes one of Bob's top-four providers;

(4) Bob becomes one of Alice's top-four providers

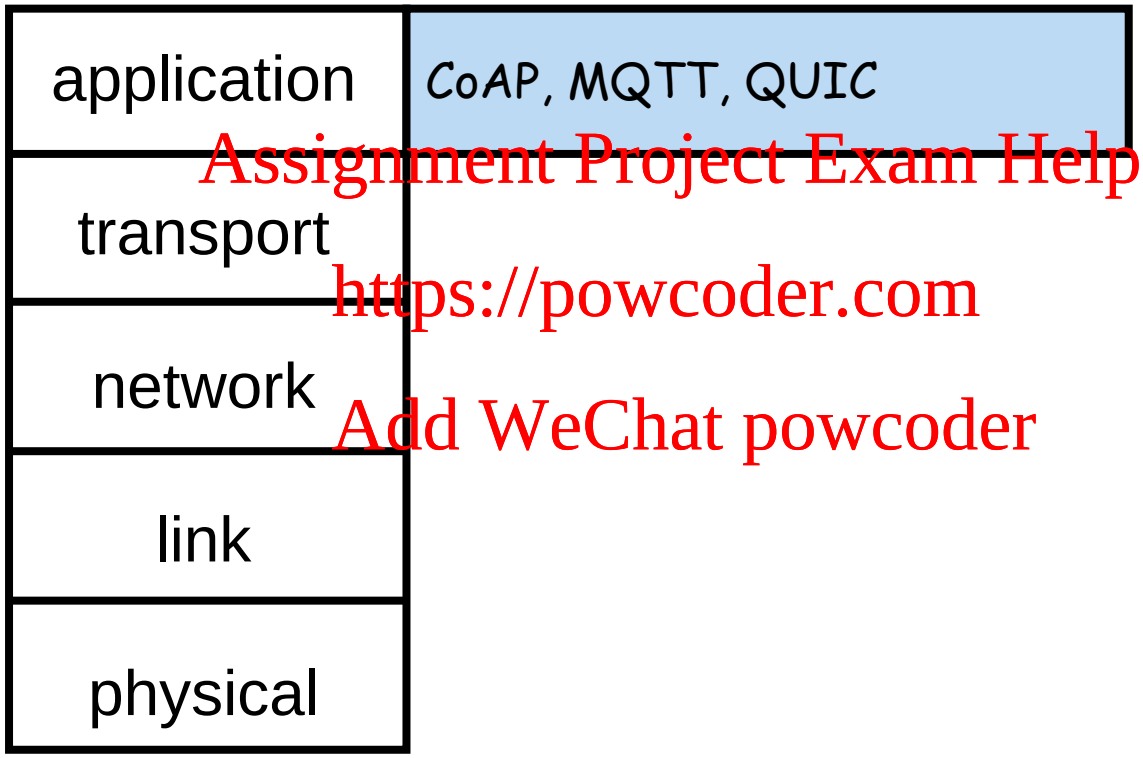
higher upload rate: find better trading partners, get file faster !

Circular DHT with shortcuts



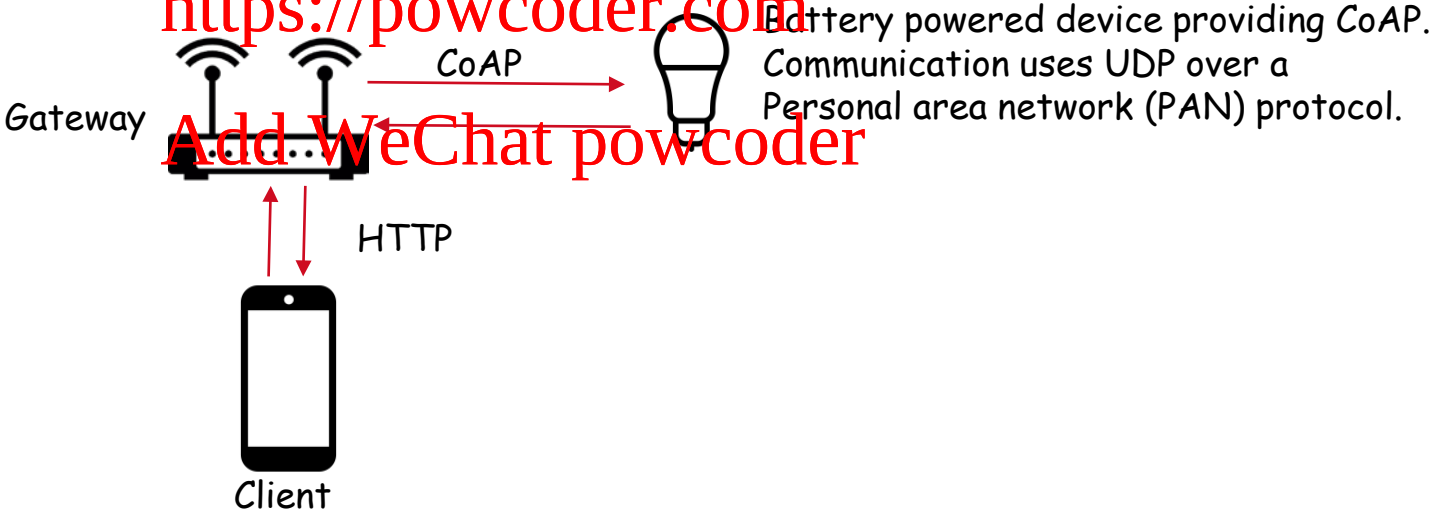
› each peer keeps track of predecessor, successor, short cuts.

Internet Protocol Stack



CoAP

- CoAP provides a request/response interaction like HTTP.
- Over UDP.
- GET, PUT, observe

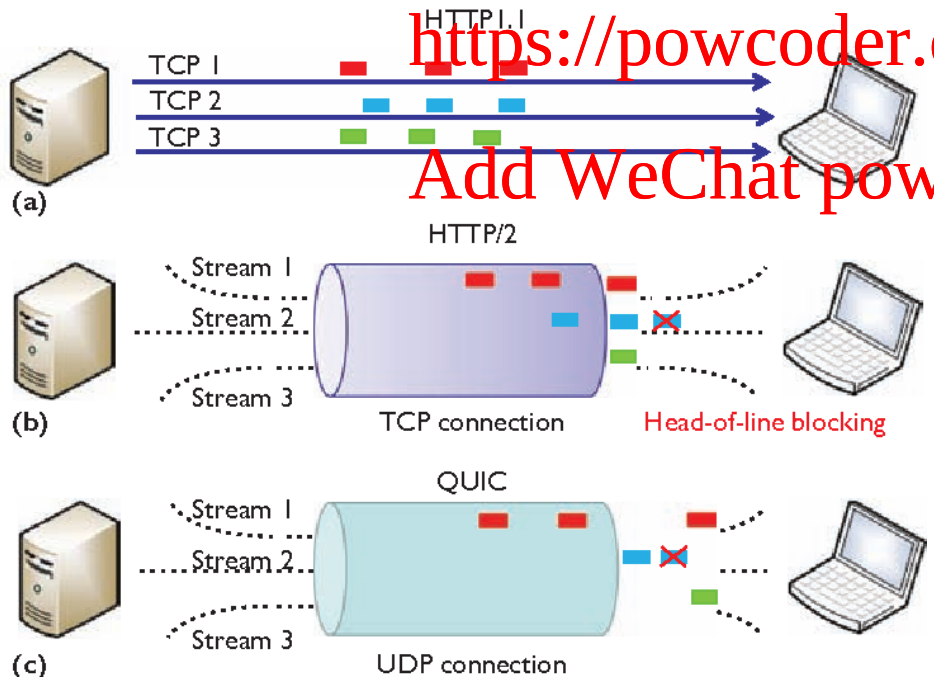


MQTT

- MQTT: Lightweight, publish-subscribe network protocol that transports messages between devices.
- Runs over TCP
- Two types of entities:
  - Broker: server receives and forwards messages.
  - Client: device connected to broker

https://mqtt.org/

Over UDP
Avoid head-of-line blocking.

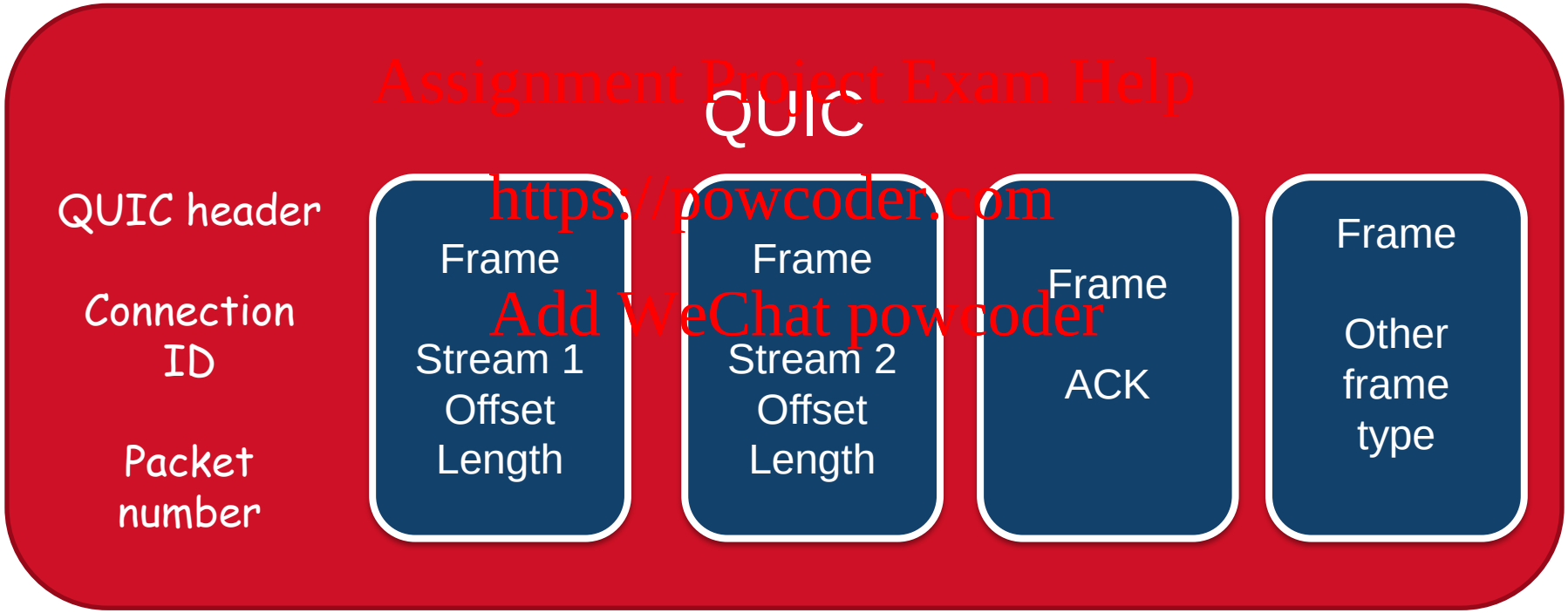


UDP does not care about ordering of packet and if packet get lost.

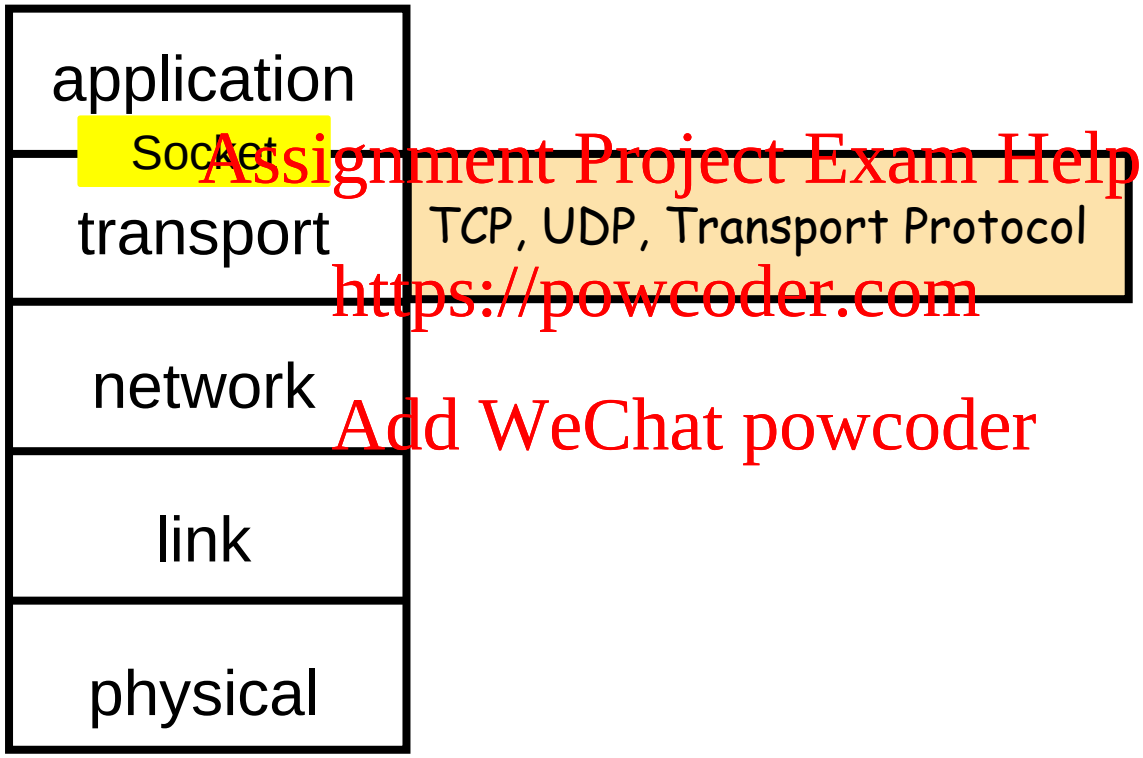
QUIC is solving this issue and it will take care of packet lost in particular stream.

https://medium.com/faun/http-2-spdy-and-http-3-quic-bae7d9a3d484

Connection ID



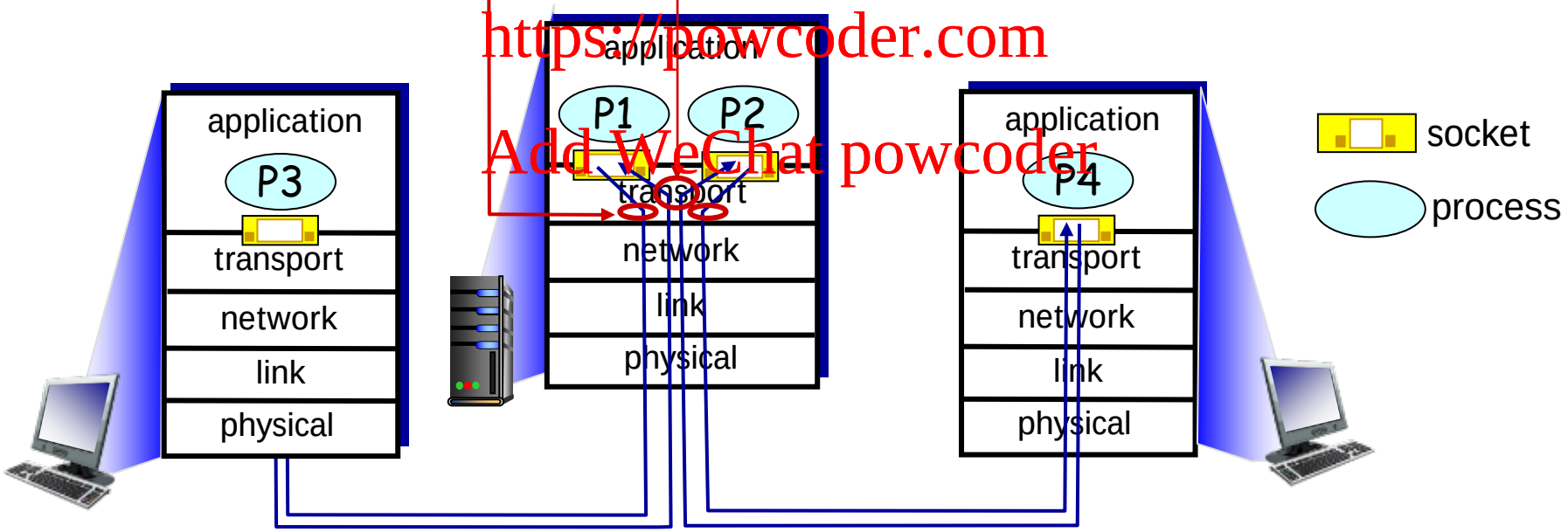
Protocol Stack



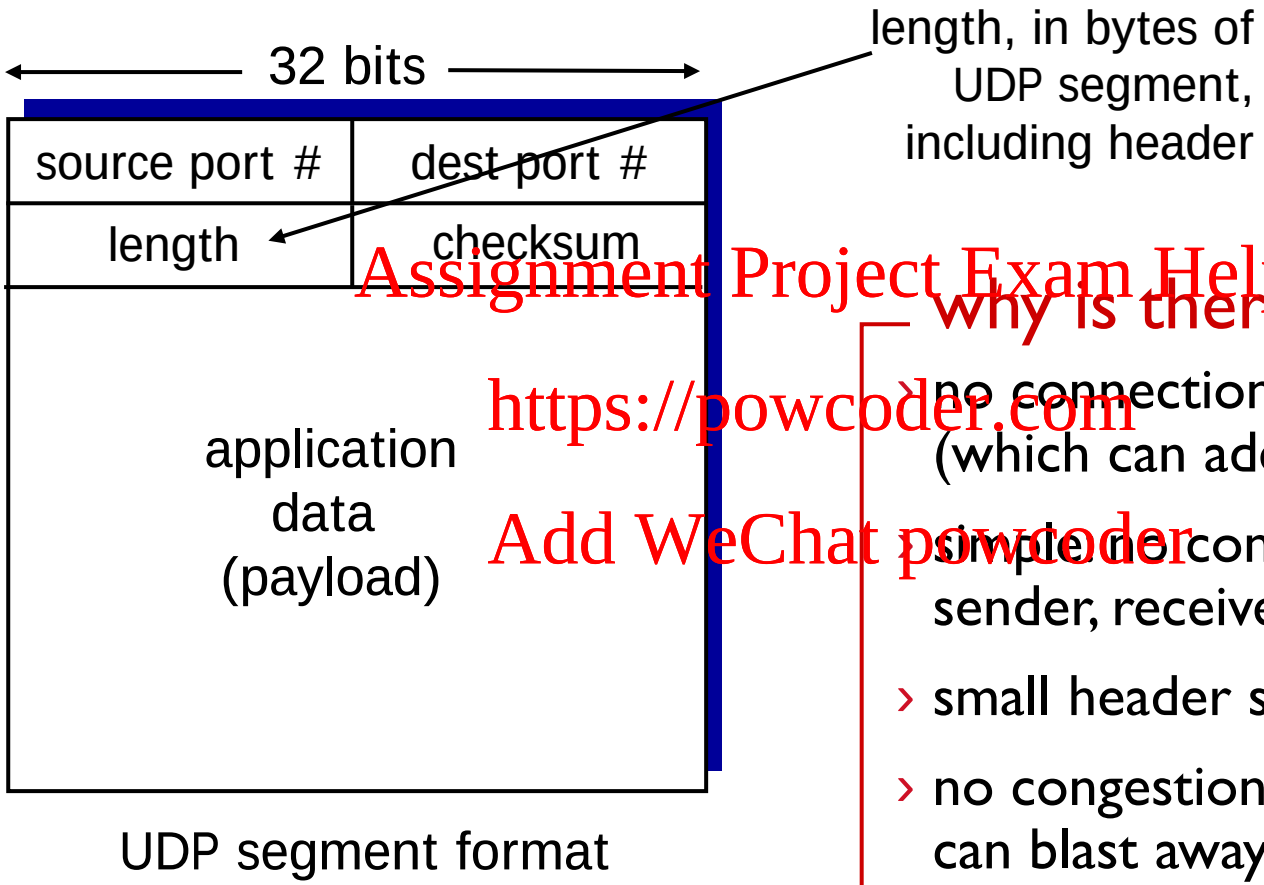
Multiplexing/demultiplexing

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



UDP

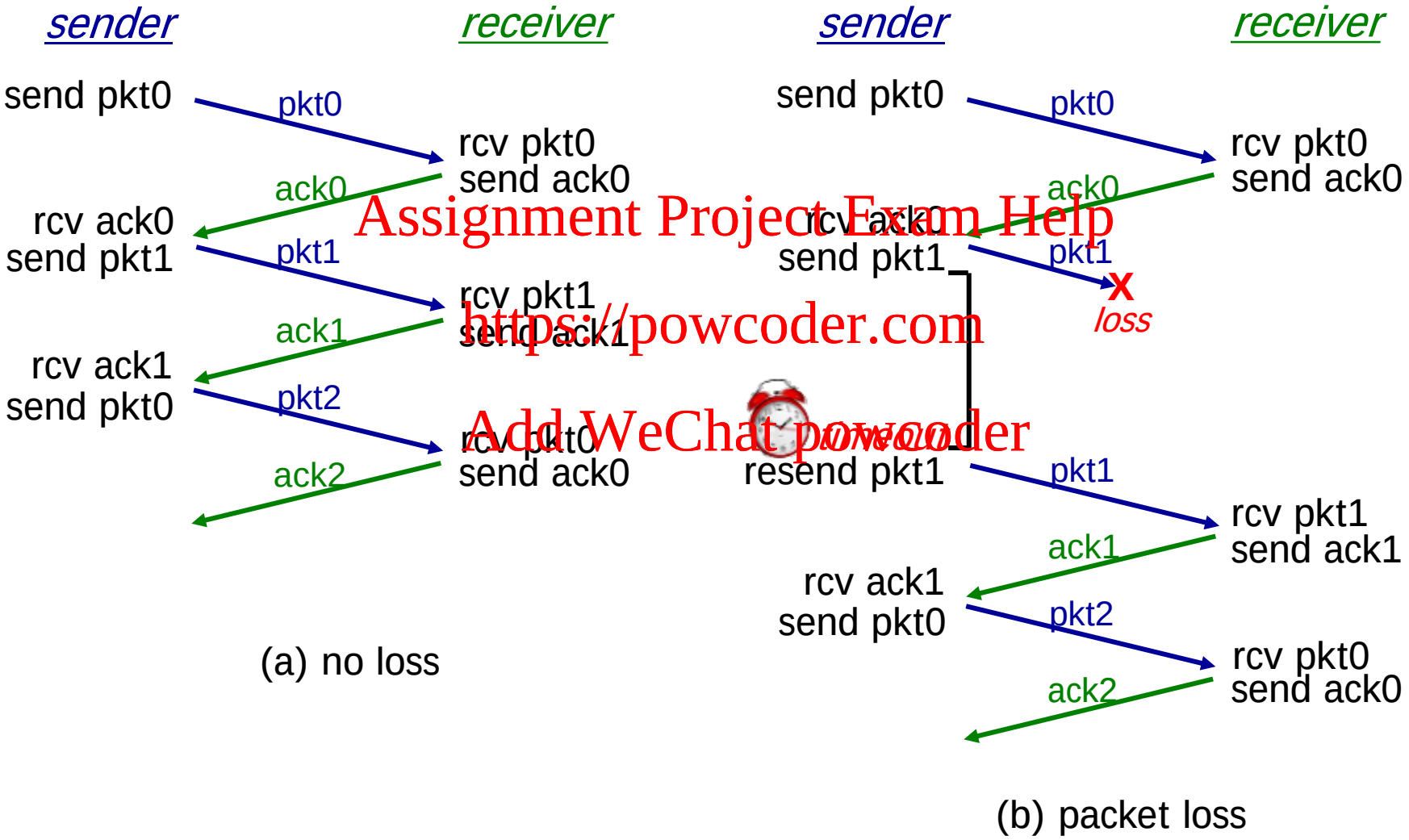


UDP segment format

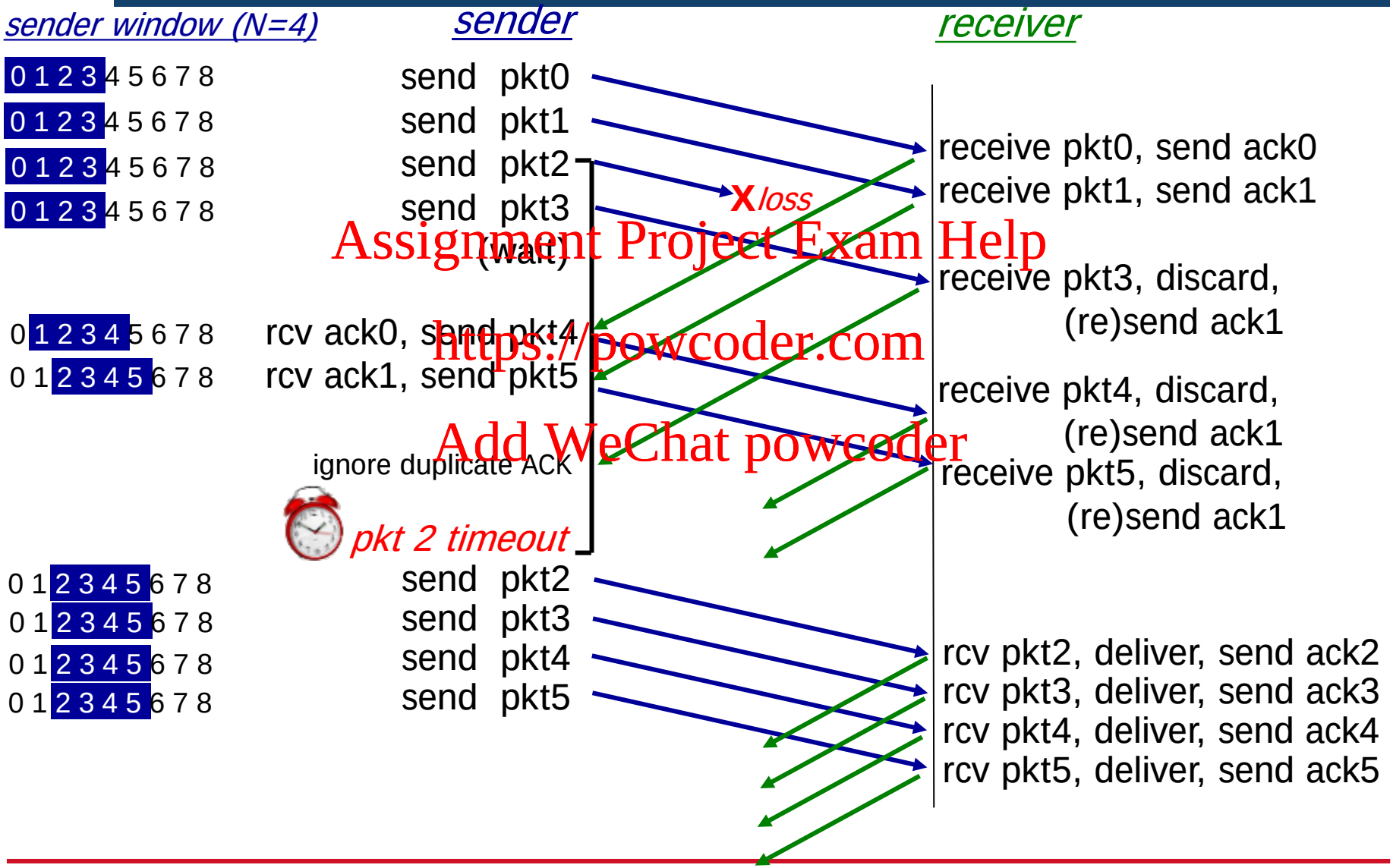
why is there a UDP?

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control: UDP can blast away as fast as desired

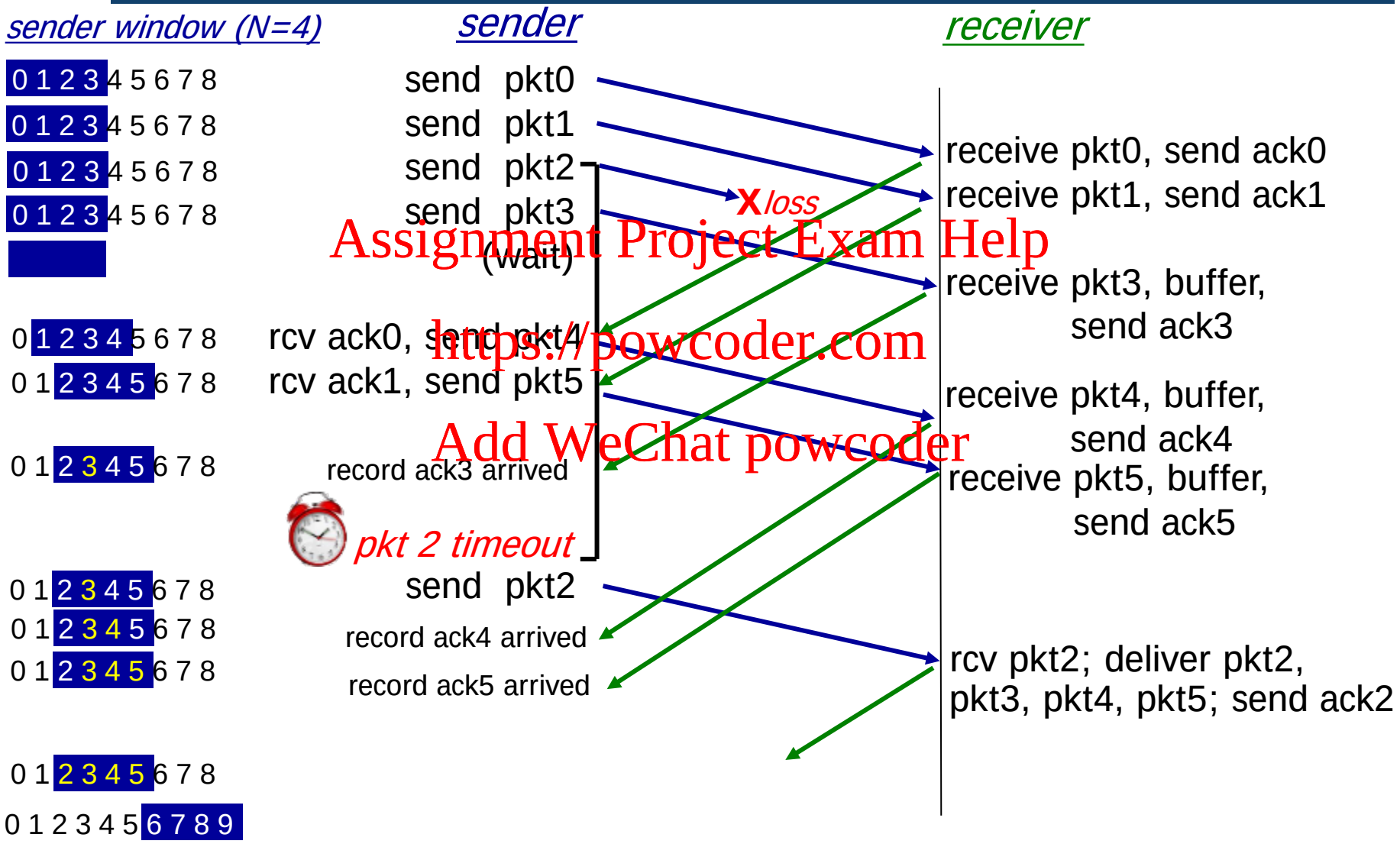
Stop and wait



GBN

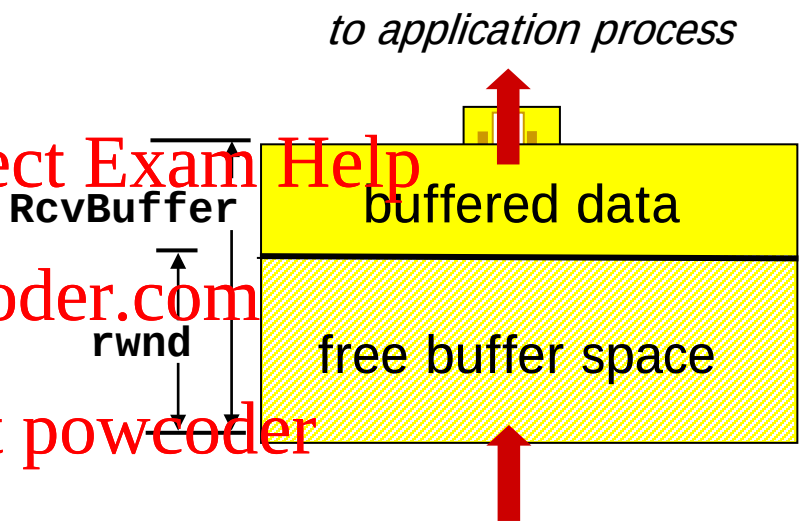


Selective repeat



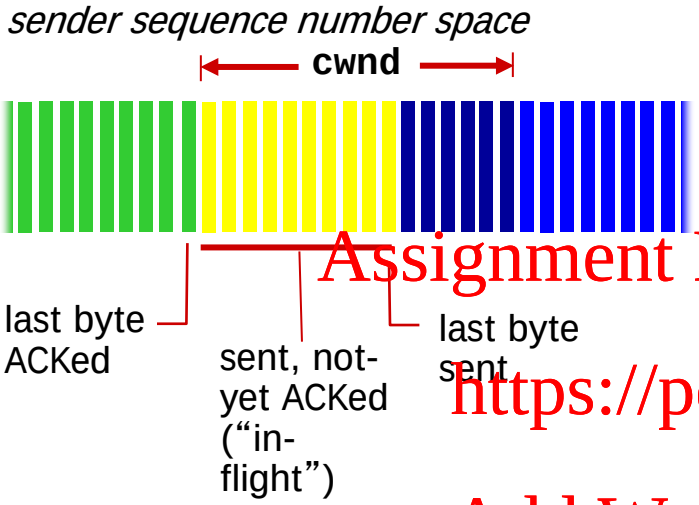
TCP flow control

- receiver "advertises" free buffer space by including **`rwnd`** value in TCP header of receiver-to-sender segments
  - **`RcvBuffer`** size set via socket options (typical default is 4096 bytes)
  - many operating systems autoadjust **`RcvBuffer`**
- sender limits amount of unacked ("in-flight") data to receiver's **`rwnd`** value
- guarantees receive buffer will not overflow



receiver-side buffering

TCP Congestion Control



TCP sending rate:

› *roughly:* send cwnd bytes, wait RTT for ACKS, then send more bytes

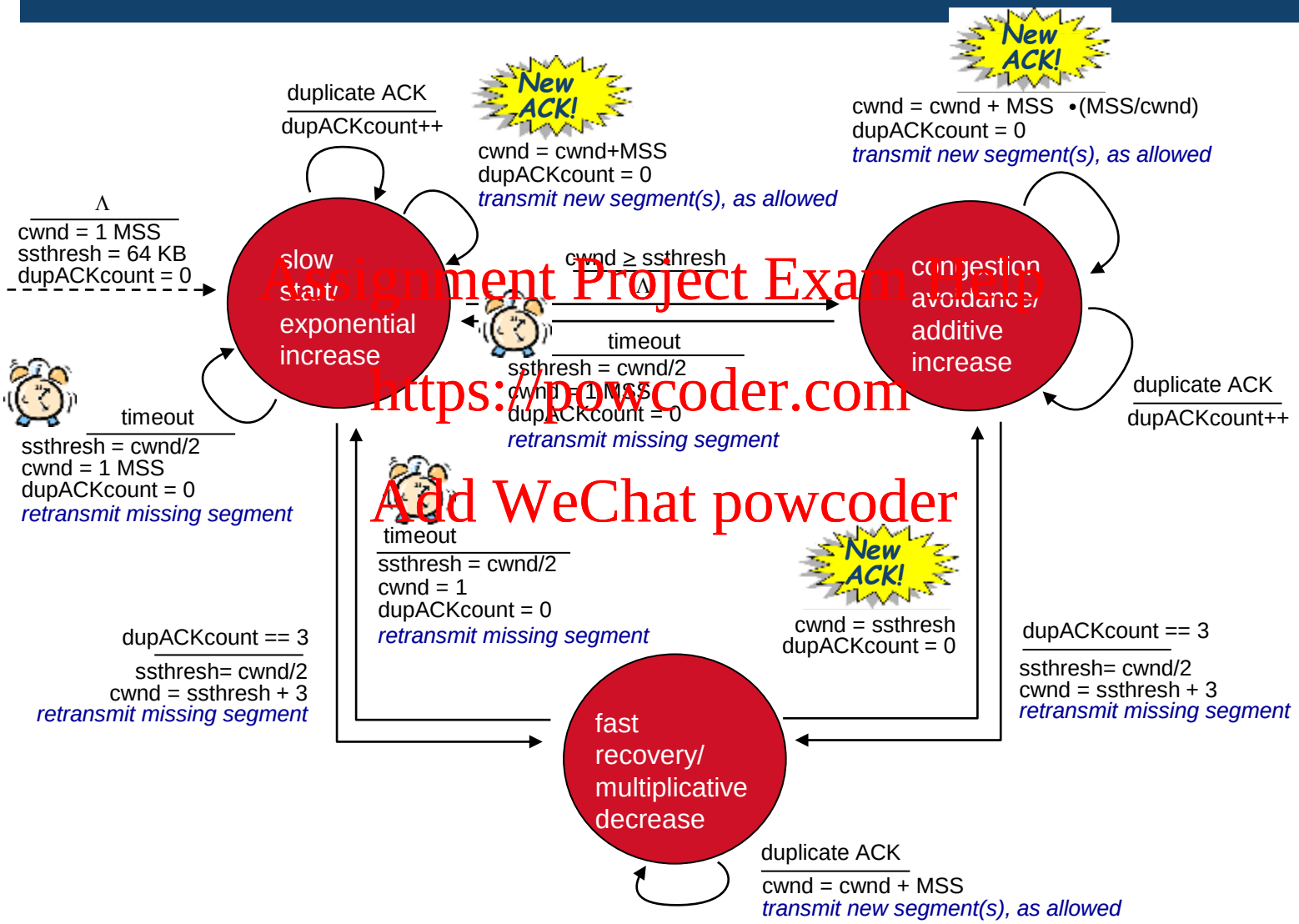
› sender limits transmission:

$$\texttt{LastByteSent} - \texttt{LastByteAcked} \leq \texttt{cwnd}$$

› **cwnd** is dynamic, function of perceived network congestion

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

TCP Congestion Control



TCP round trip time, timeout

› timeout interval: **EstimatedRTT** plus "safety margin"


```
EstimatedRTT = (1- α)*EstimatedRTT + α*SampleRTT
```

typical value: α = 0.125


```
DevRTT = (1-β)*DevRTT +
         β*|SampleRTT-EstimatedRTT|
```

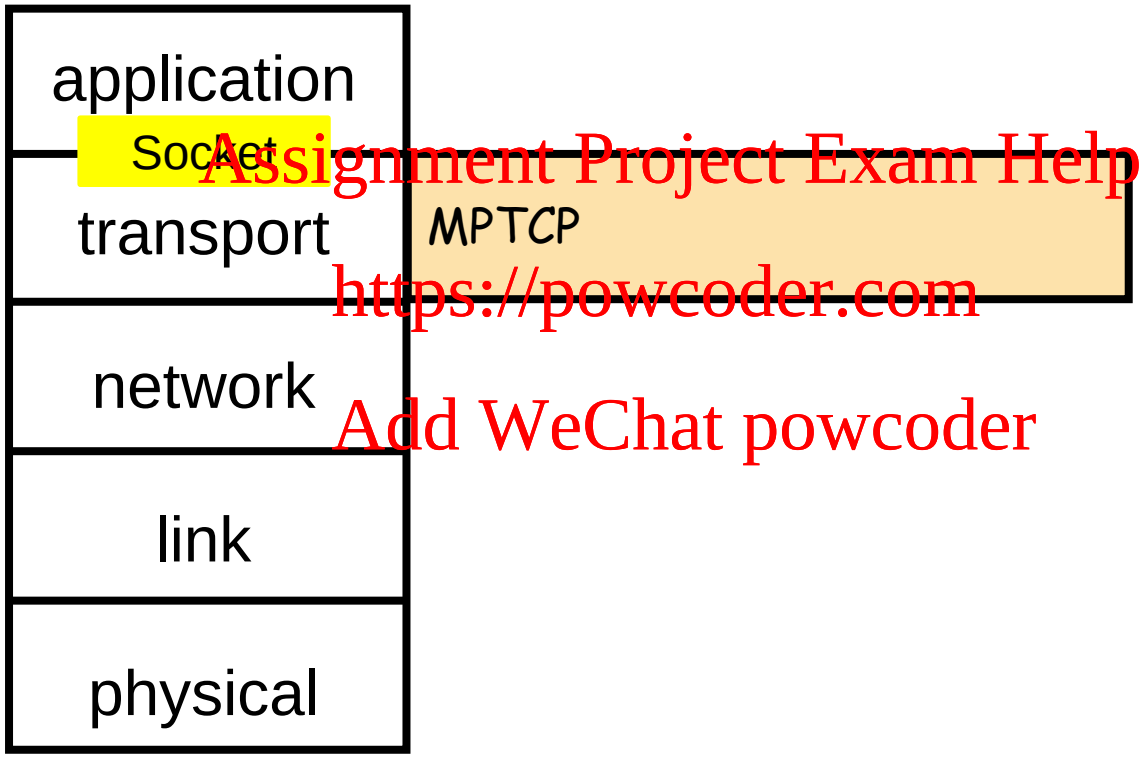

(typically, β = 0.25)

```
TimeoutInterval = EstimatedRTT + 4*DevRTT
```

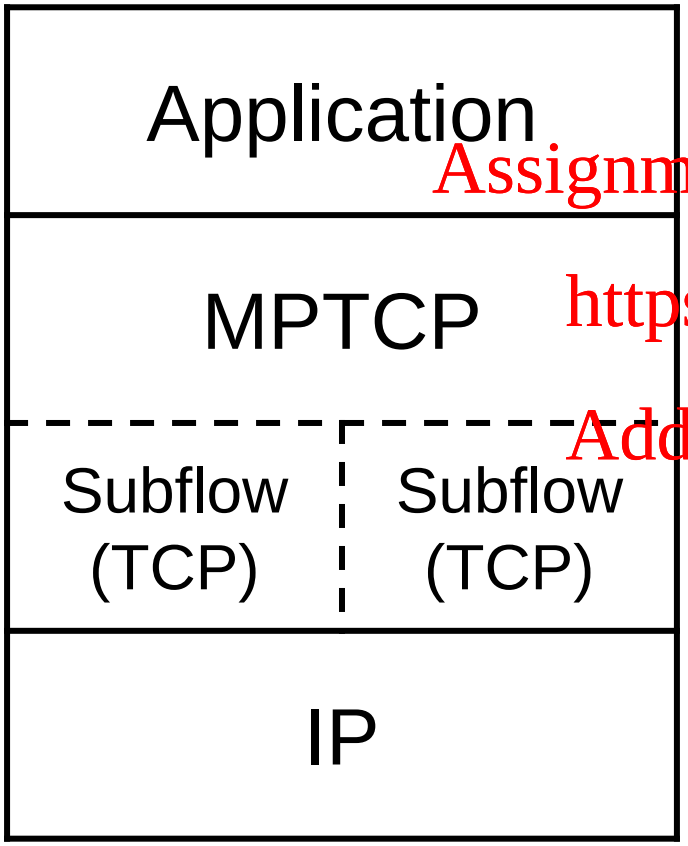
estimated RTT

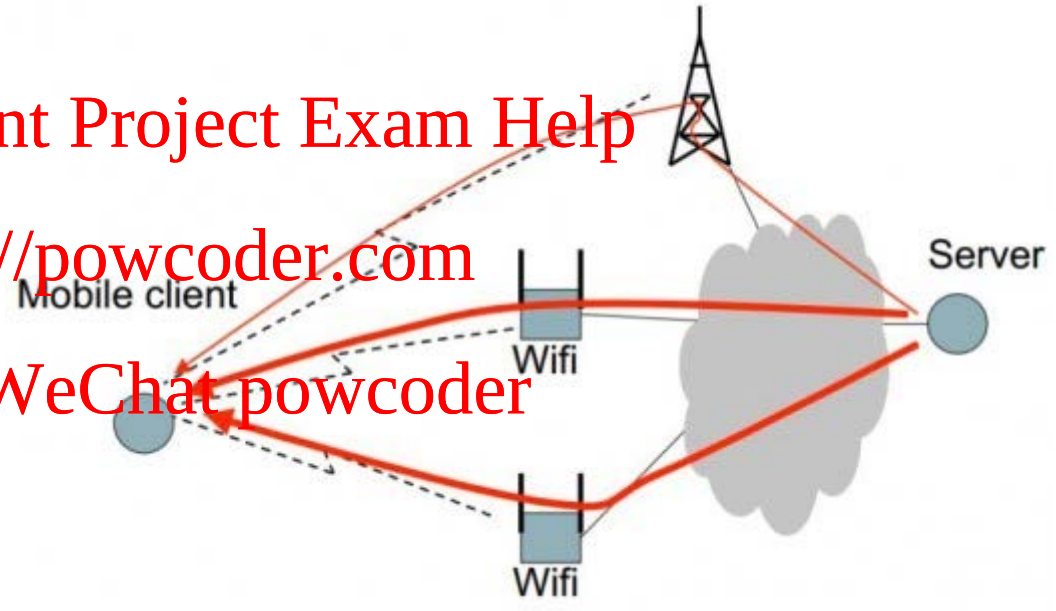
"safety margin"

Protocol Stack



MPTCP in Stack



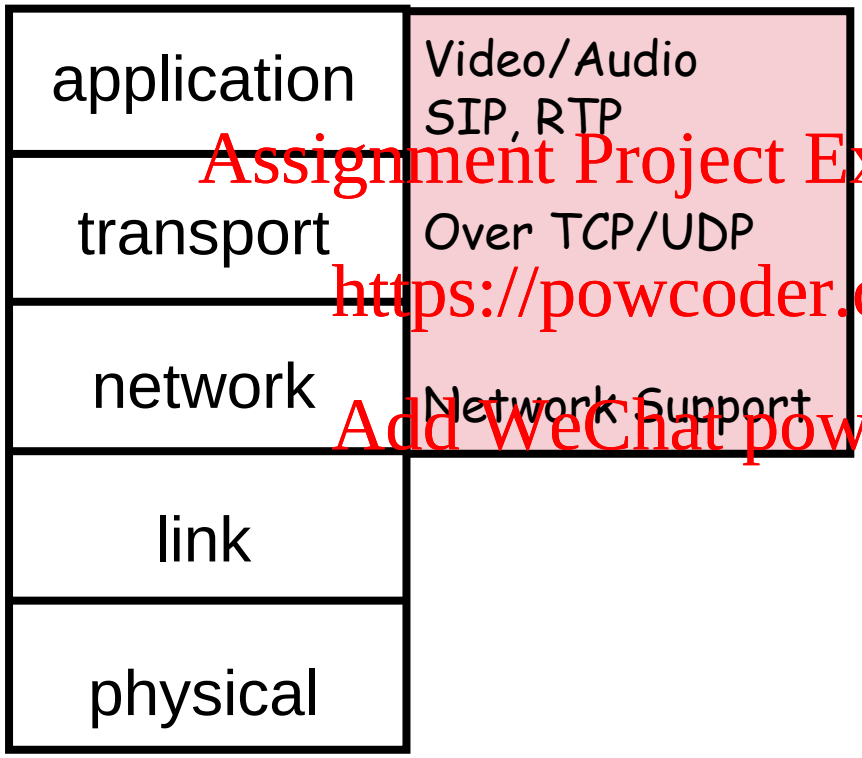


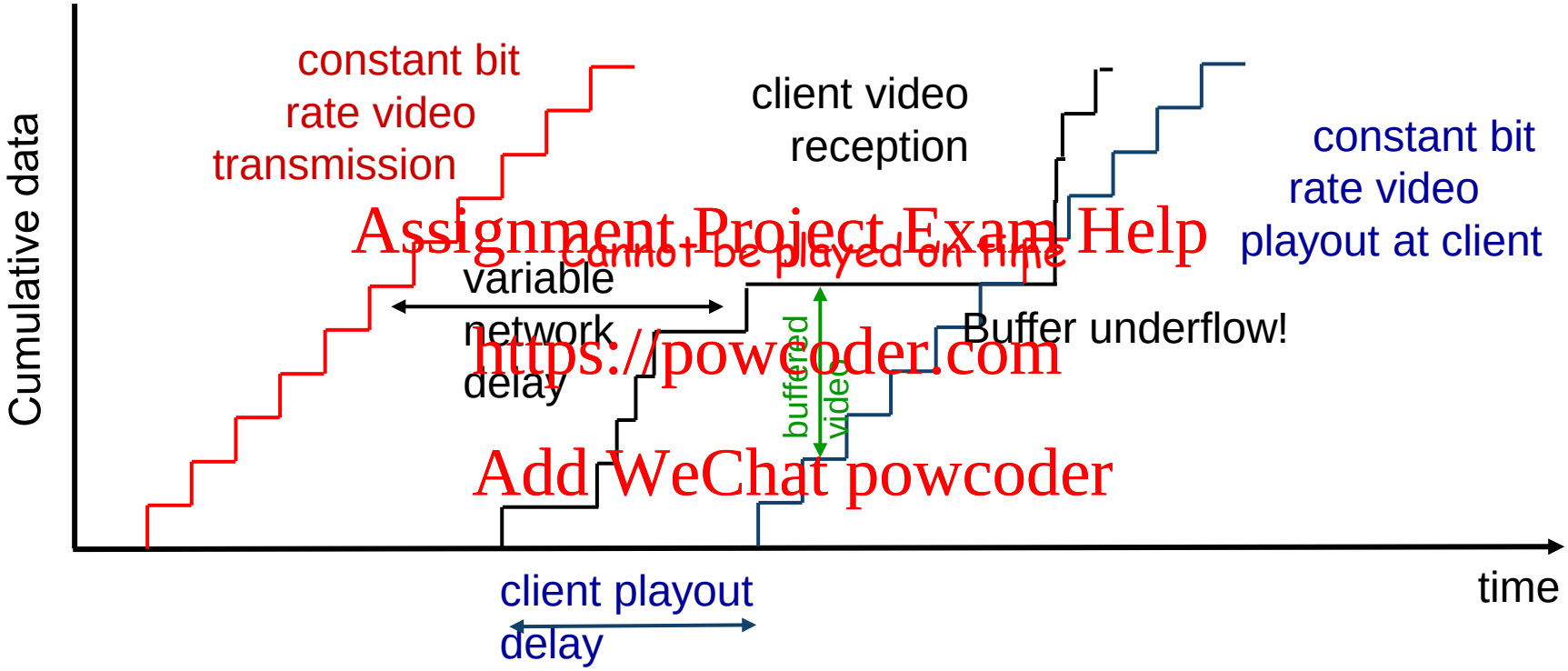
https://pocketnow.com/multipath-tcp

MPTCP in Stack

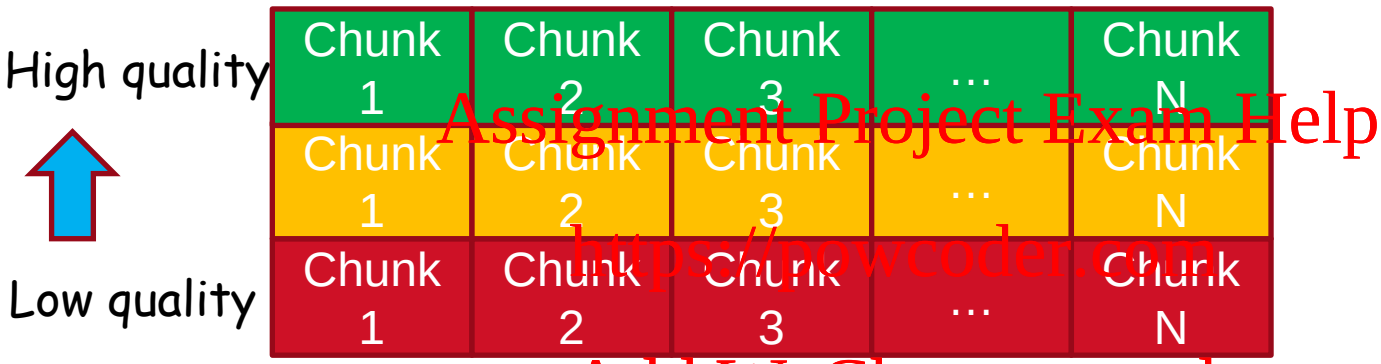
- *Initialization: MP_CAPABLE, JOIN, Token*
- *Sequence number: Subflow sequence number + data sequence number*
- *Flow control: Receive window size is for all subflows.*
- *Congestion control: Updated AIMD for fairness.*

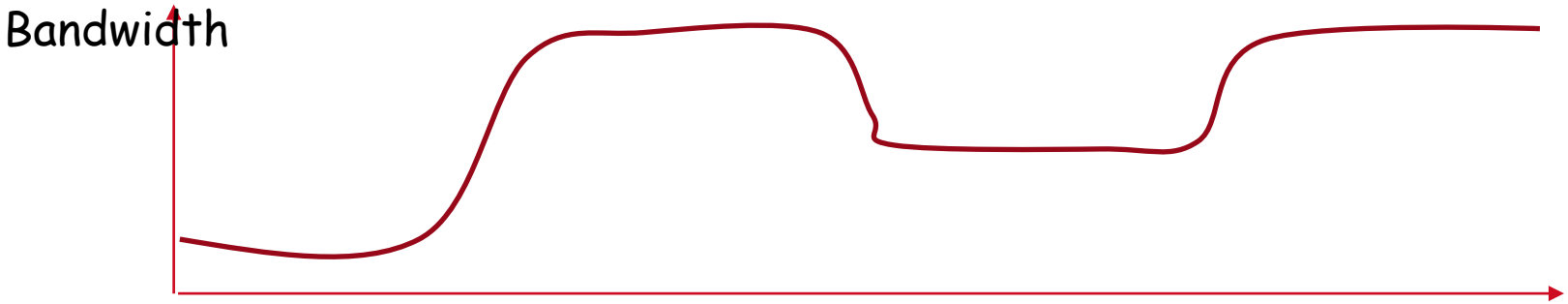
Internet Protocol Stack

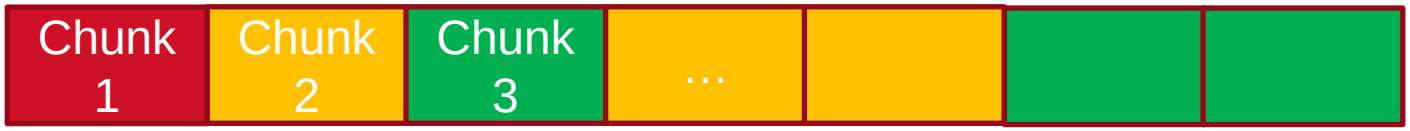



Cumulative data
constant bit rate video transmission
client video reception
constant bit rate video playout at client
Assignment Project Exam Help
cannot be played on time
variable network delay
Buffer underflow!
https://powcoder.com
buffered video
Add WeChat powcoder
client playout delay
time

Streaming multimedia: DASH







Adaptive playout delay

- *goal:* low playout delay, low delay loss rate
- *approach:* adaptive playout delay adjustment:
  - estimate network delay, adjust playout delay at beginning of each talk spurt
  - silent periods compressed and elongated
- adaptively estimate packet delay: (EWMA - exponentially weighted moving average):

$$d_i = (1-\alpha)d_{i-1} + \alpha\,(r_i - t_i)$$

delay estimate after ith packet

small constant,

time received - time sent (timestamp)

measured delay of ith packet

Adaptive playout delay

- also useful to estimate average deviation of delay, v_i

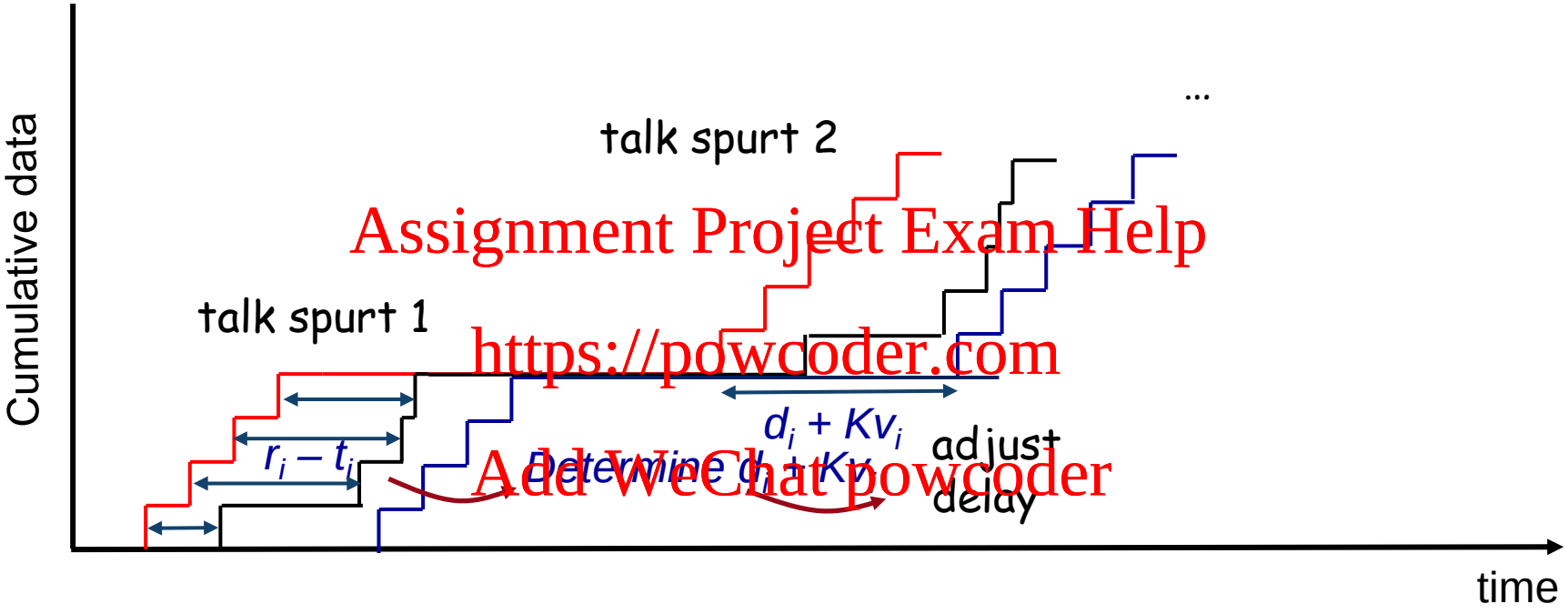
$$v_i = (1-\beta)v_{i-1} + \beta\,|r_i - t_i - d_i|$$

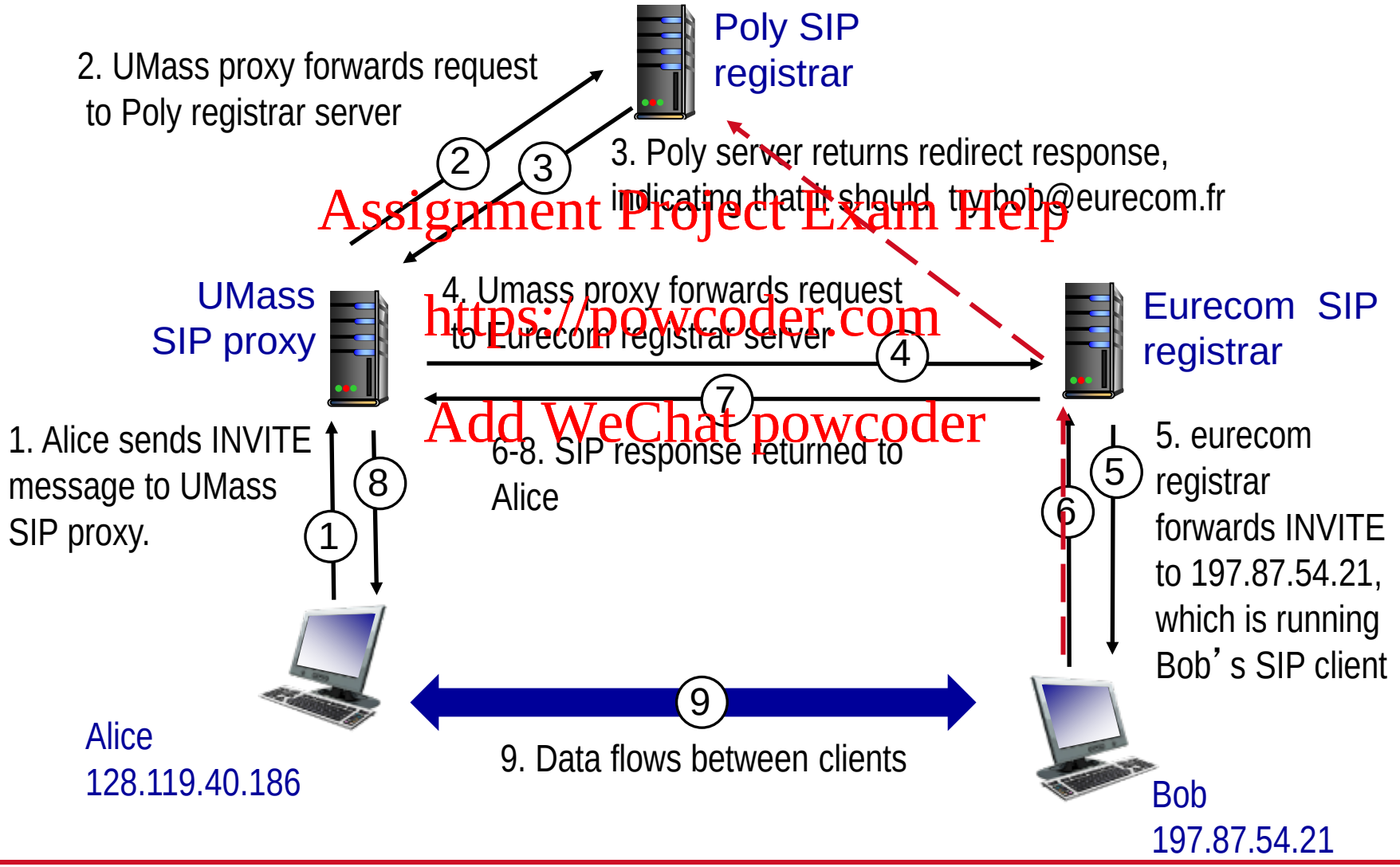
  - estimates d_i, v_i calculated for every received packet, but used only at start of talk spurt

  - for first packet in talk spurt, playout time is:

$$playout\text{-}time_i = t_i + d_i + Kv_i$$

Delay jitter





RTP

payload type	sequence number	time stamp	Synchronization Source ID (SSRC)	Miscellaneous fields

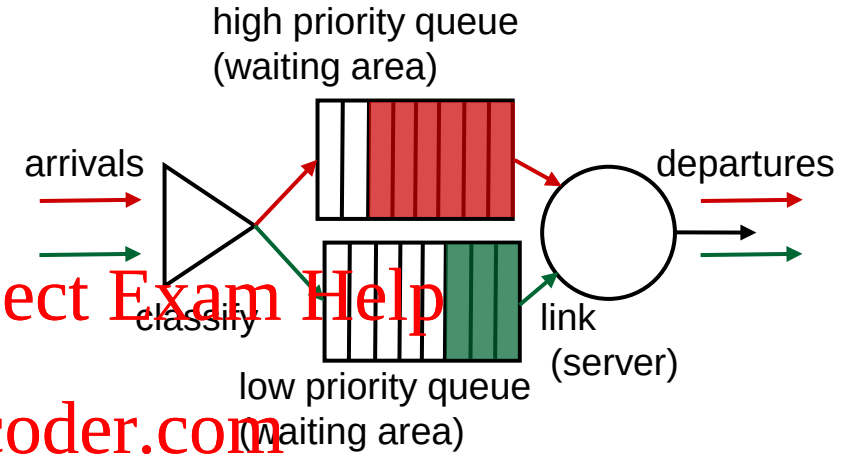
- payload type (7 bits): indicates type of encoding currently being used.
- sequence # (16 bits): increment by one for each RTP packet sent
- *timestamp field (32 bits long):* sampling instant of first byte in this RTP data packet

- Sequence + timestamp：packet loss or new talk spurt.

Scheduling policies: priority

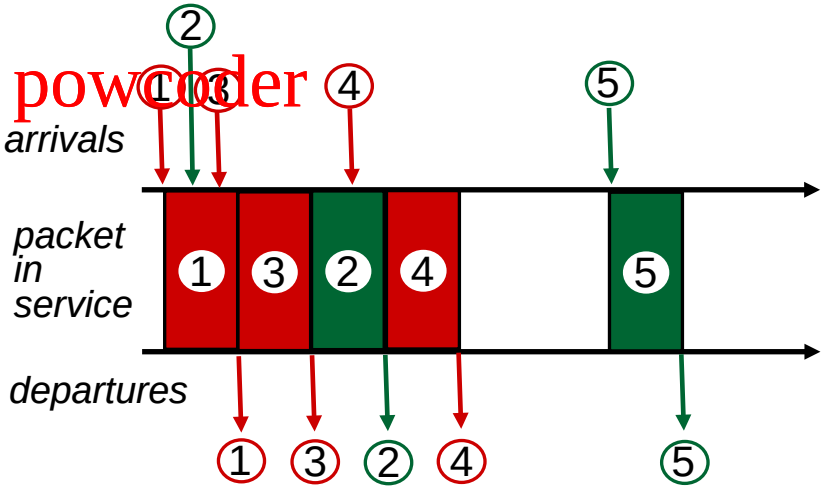
priority scheduling: send highest priority queued packet

non-preemptive



› multiple *classes*, with different priorities

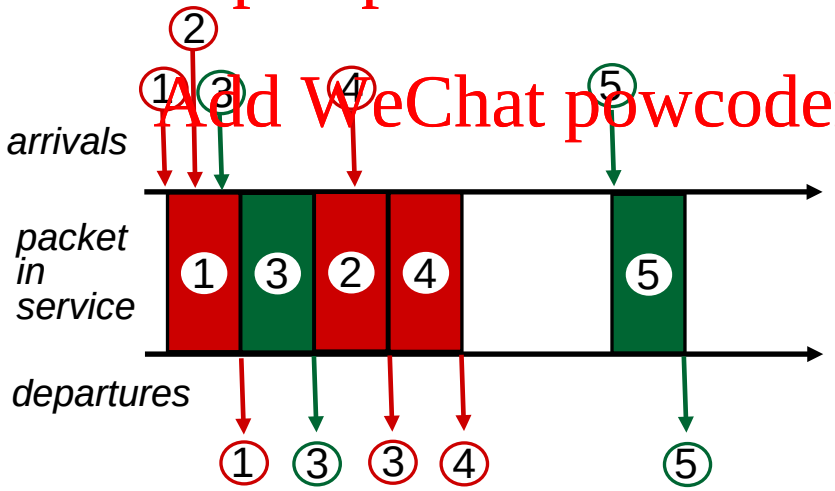
- class may depend on marking or other header info, e.g. IP source/dest, port numbers, etc.
- real world example?



Scheduling policies (con't)

Round Robin (RR) scheduling:

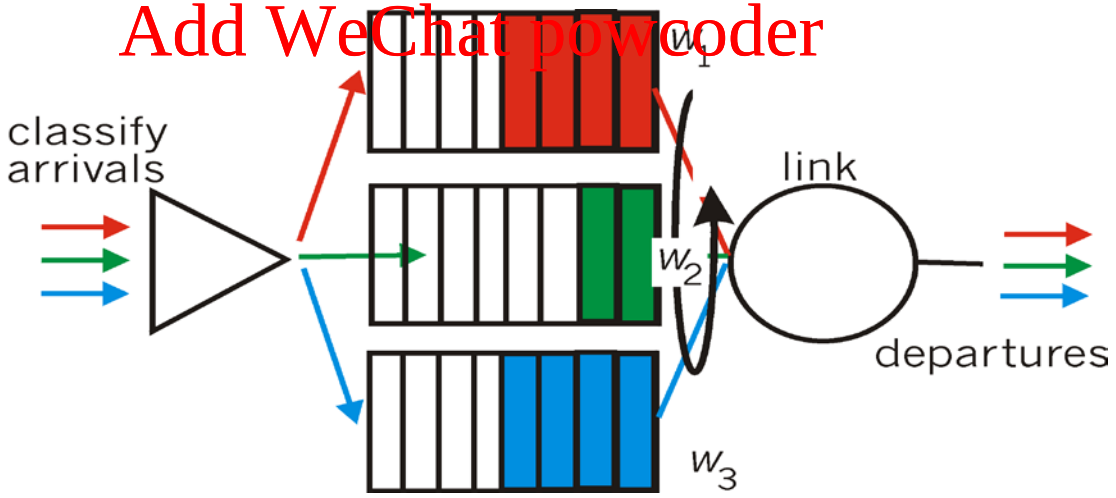
- multiple classes, with equal priority
- cyclically scan class queues, sending one complete packet from each class (if available)



Scheduling policies (con't)

Weighted Fair Queuing (WFQ):

- Each class *i* is assigned a weight w_i
- *Guarantee*: if there are class *i* packets to send (during some interval) then class *i* receives a fraction of service which is $w_i / (\sum w_j)$
- On a link with transmission rate *R*, class *i* achieves throughput $Rw_i / (\sum w_j)$

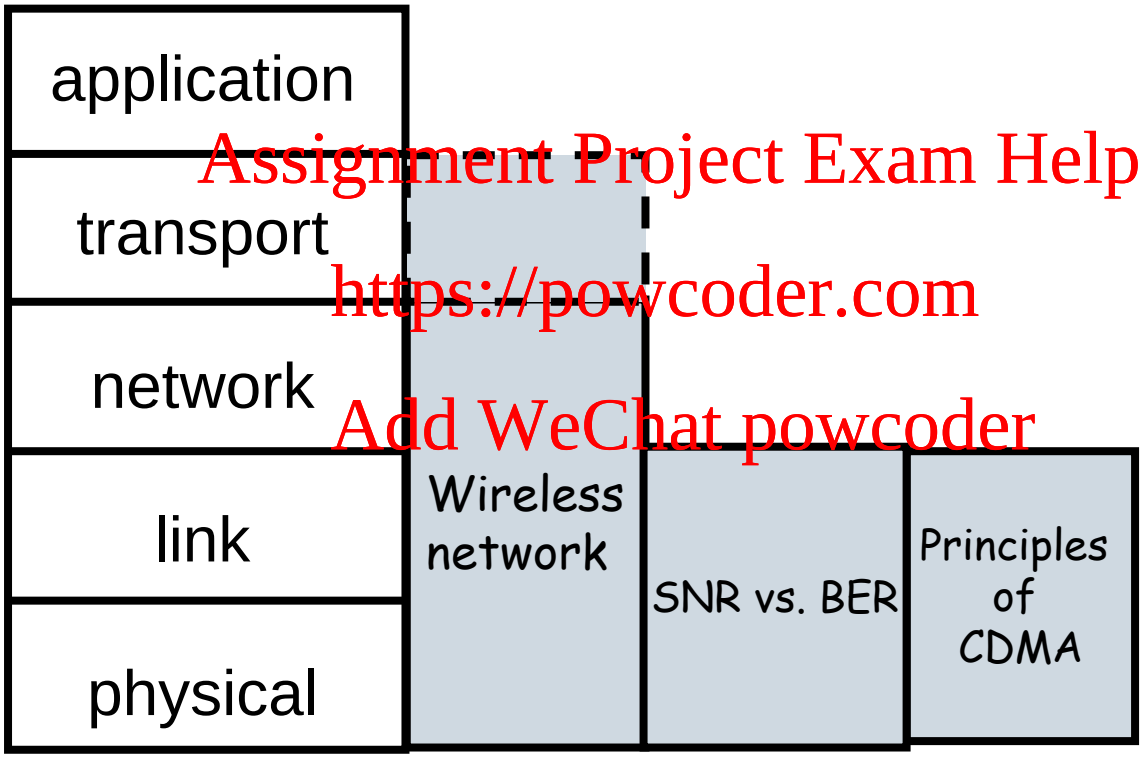


Policing mechanisms: implementation

token bucket: limit input to specified *burst size* and *average rate* (useful to police the flow)

- bucket can hold b tokens
- a packet must remove a token from bucket to be transmitted into the network
- tokens generated at rate *r token/sec* unless bucket full (token ignored)
- *over interval of length t: number of packets admitted less than or equal to (rt + b)*
- *Token-generation rate r limits the rate at which packets enter the network*

Internet Protocol Stack



Wireless Physical layer

› *SNR versus BER tradeoffs*

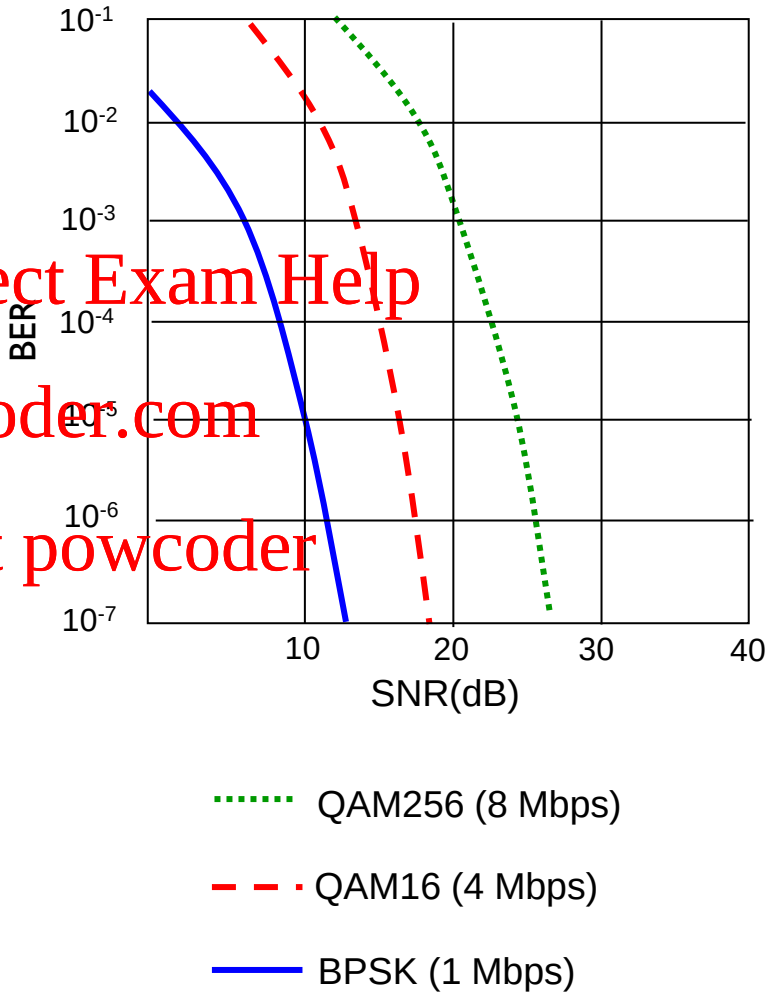
- *Different physical layer modulation:*

What is the meaning of dB?

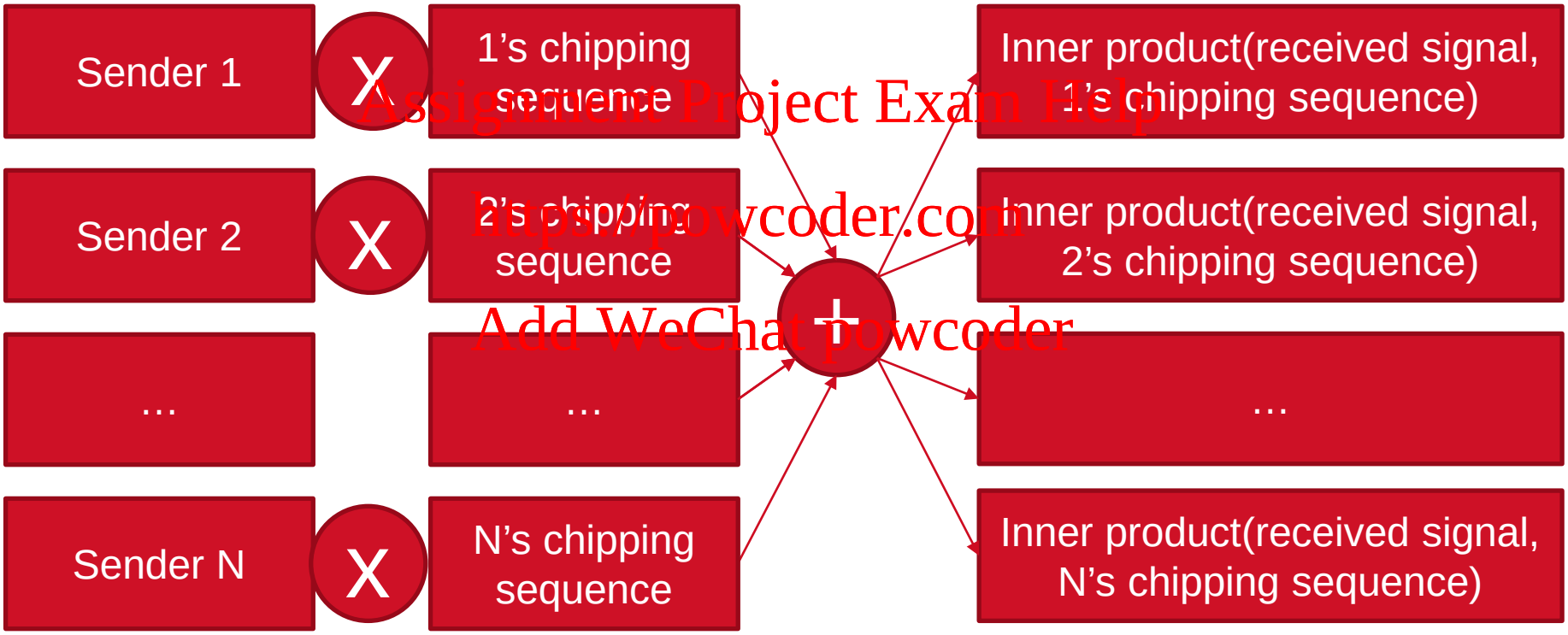
Normal/Gaussian distribution, Q fucntion

Week 9, Assignment 2 Q4

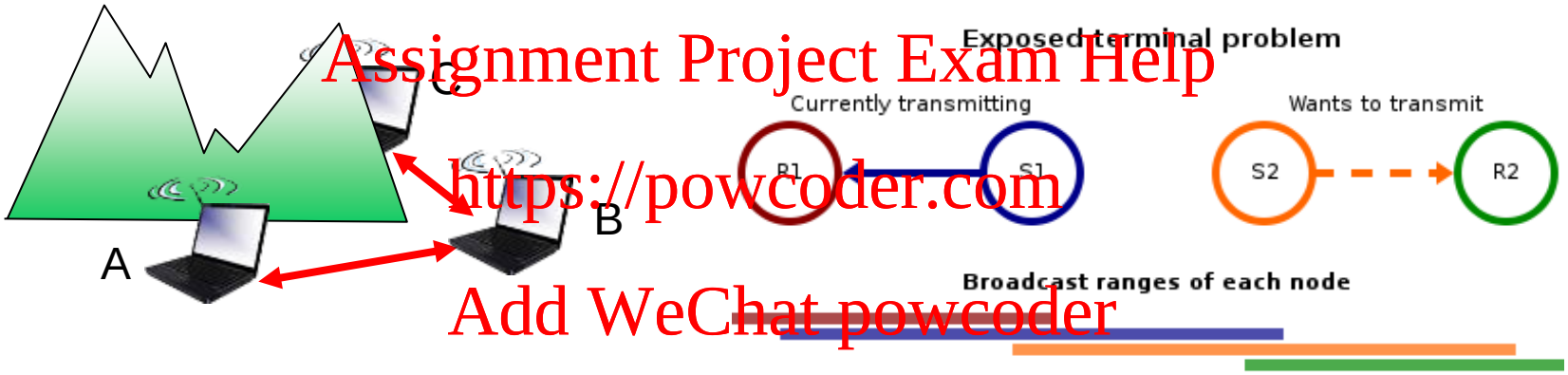
In final exam.



CDMA



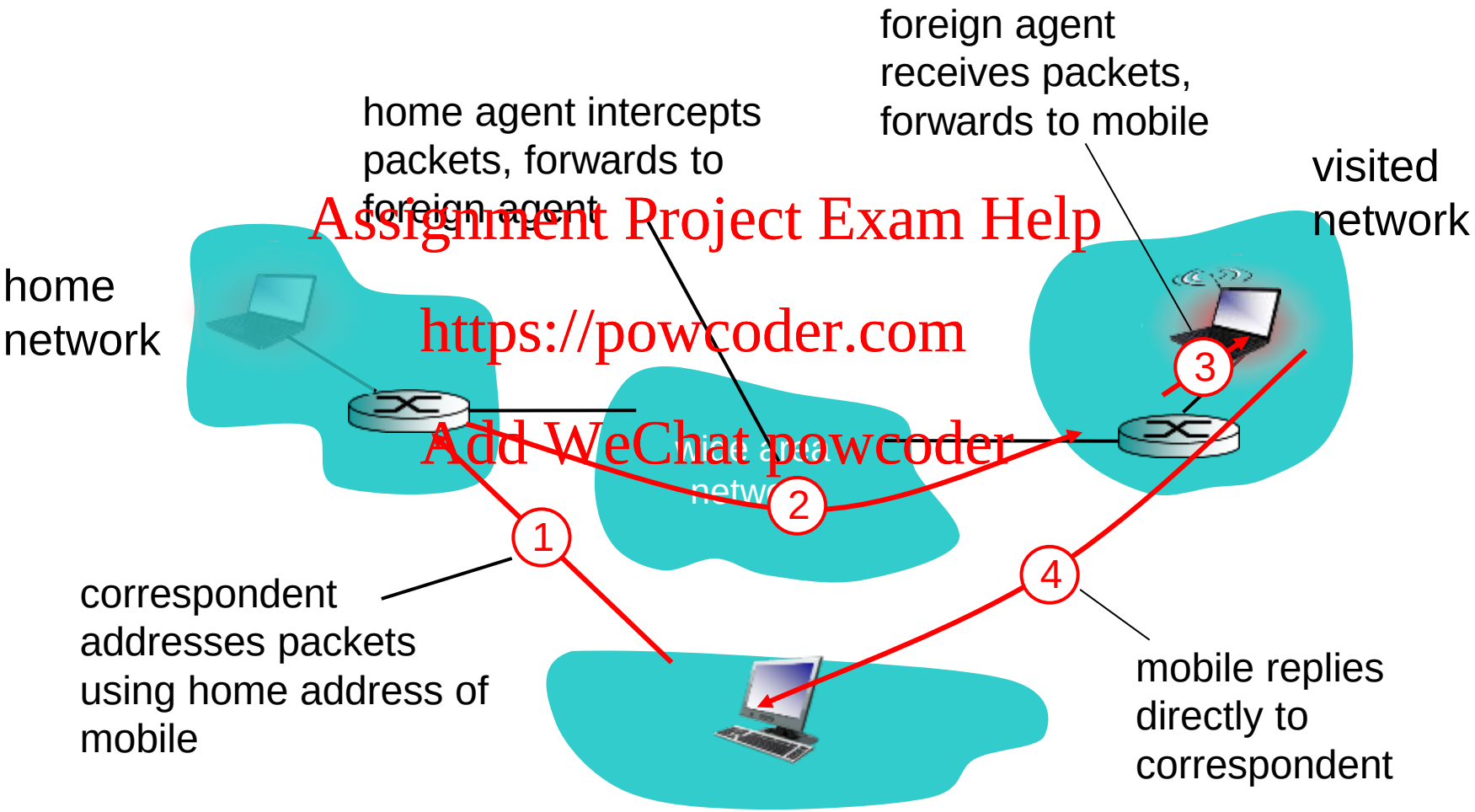
Hidden terminal and exposed terminal



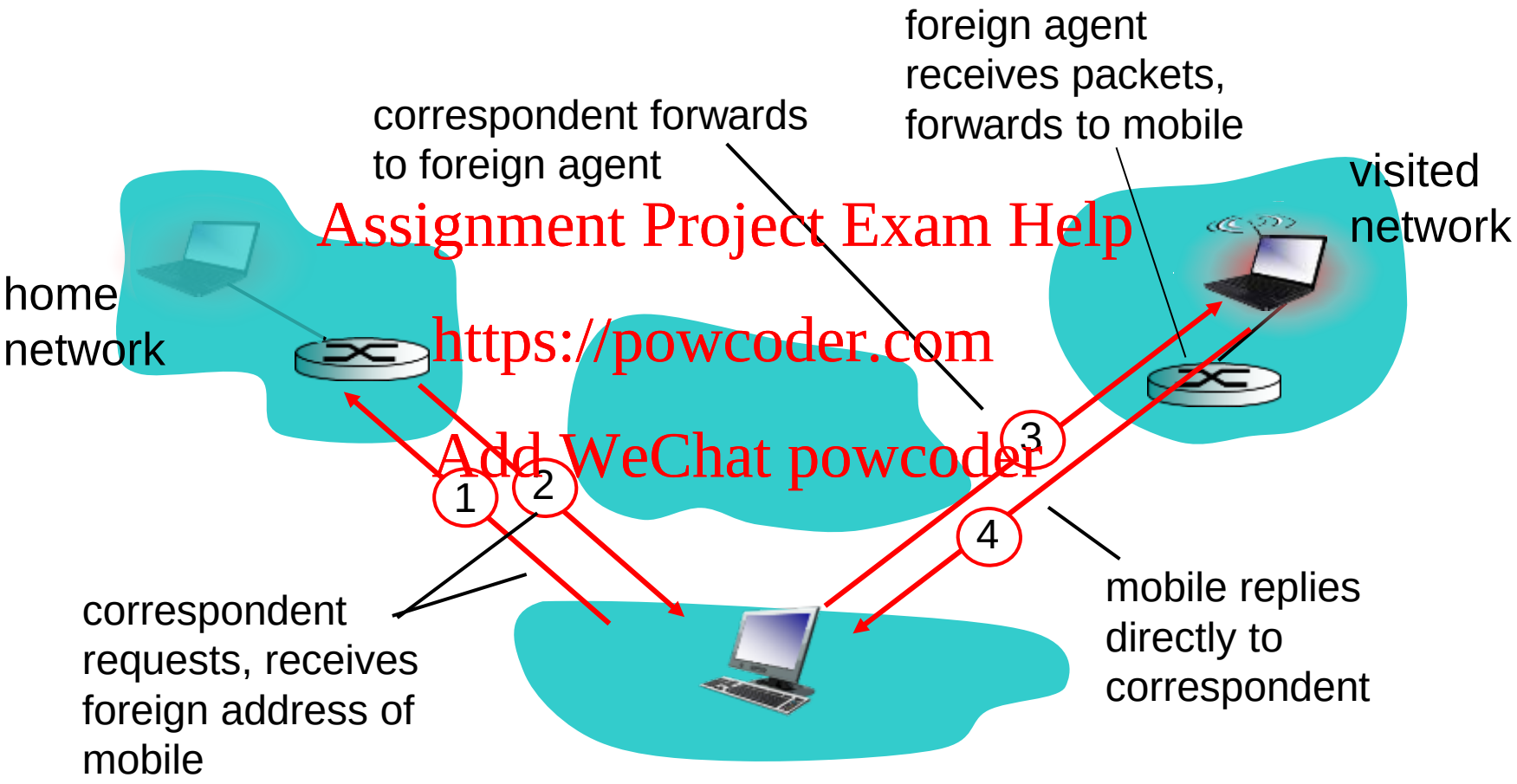
Hidden terminal problem

Exposed terminal problem

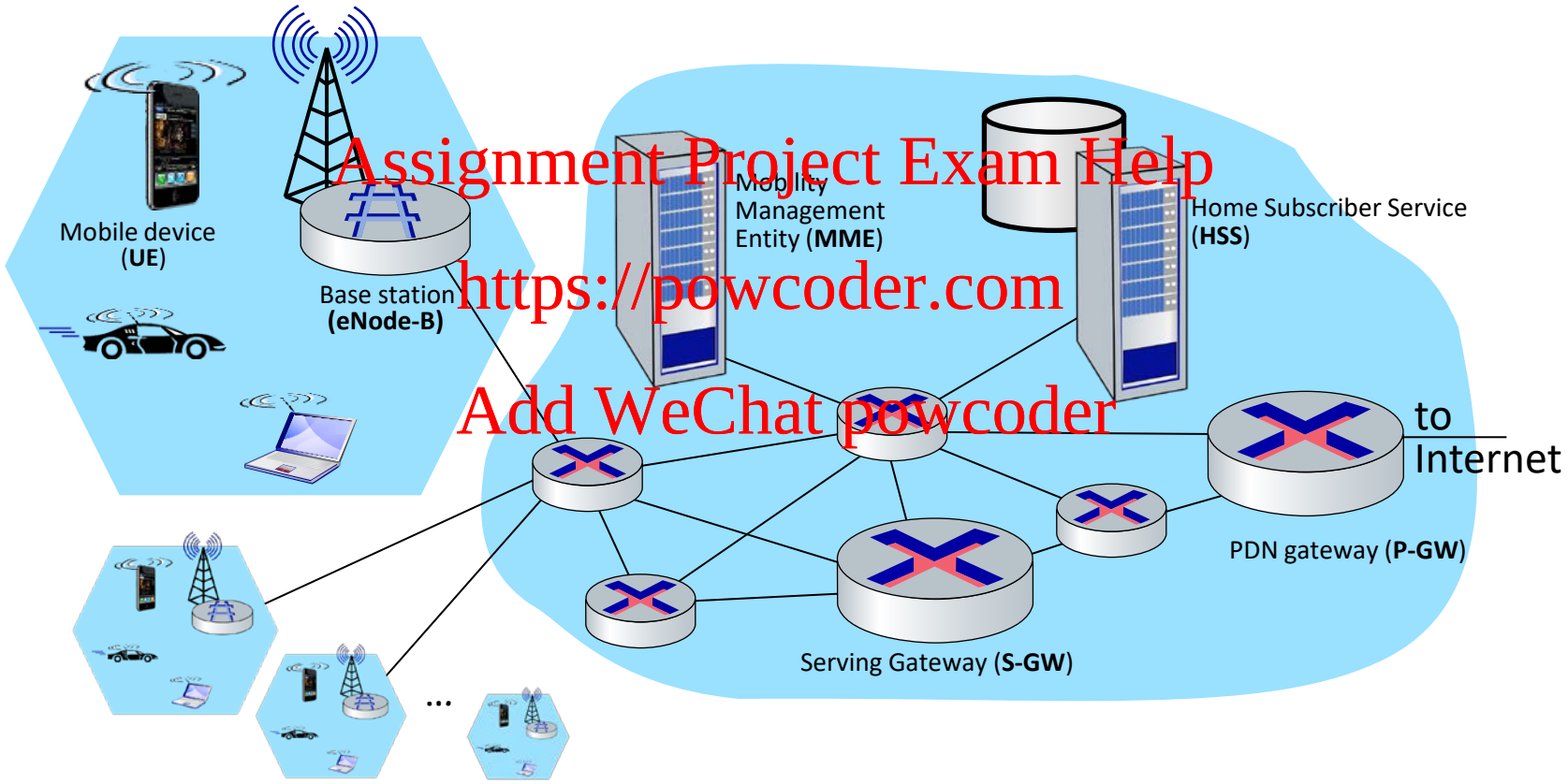
Mobility via indirect routing



Mobility via direct routing



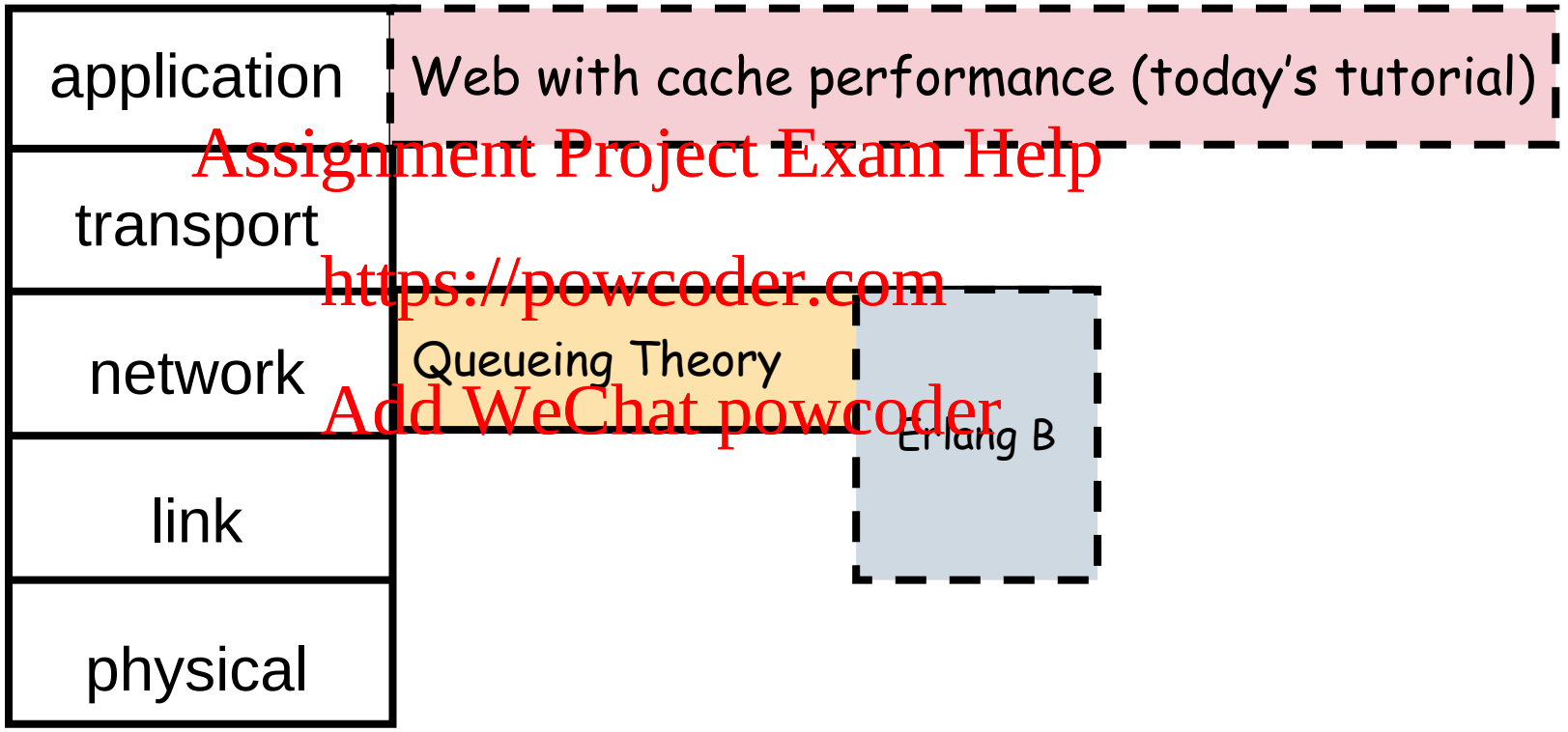
4G LTE



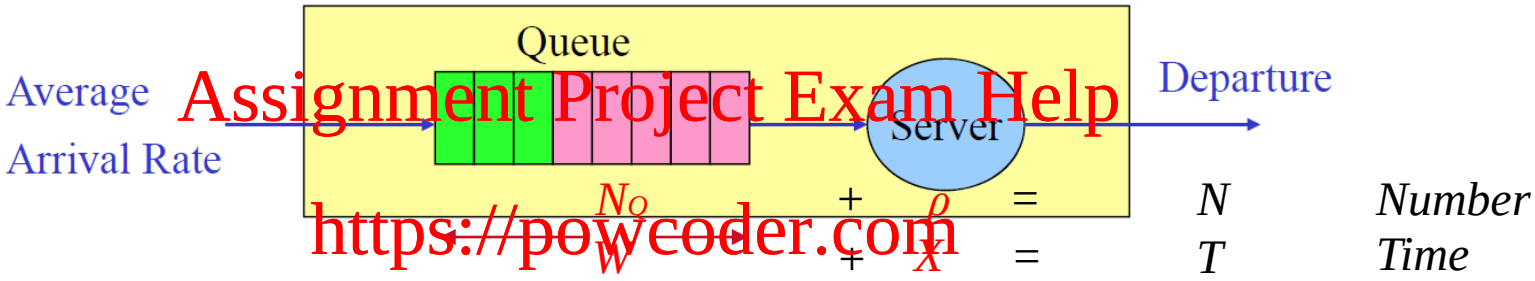
Wireless, mobility: impact on higher layer protocols

- logically, impact *should* be minimal …
  - best effort service model remains unchanged
  - TCP and UDP can (and do) run over wireless, mobile
- … but performance-wise:
  - packet loss/delay due to bit-errors (discarded packets, delays for link-layer retransmissions), and handoff
  - TCP interprets loss as congestion, will decrease congestion window un-necessarily

Internet Protocol Stack: Theory



Queueing System



- W: average waiting time in queue
- X: average service time

- T: average time spent in system ($T = W + X$)
- N_Q = average number of customers in queue
- ρ = utilization = average number of customers in service
- N = average number of customer in system ($N = N_Q + \rho$)
- Want to show later: $N = \lambda T$ (Little's theorem)
- λ Average arrival rate

Stationary Distribution Derivation

Transition diagram and balanced equations

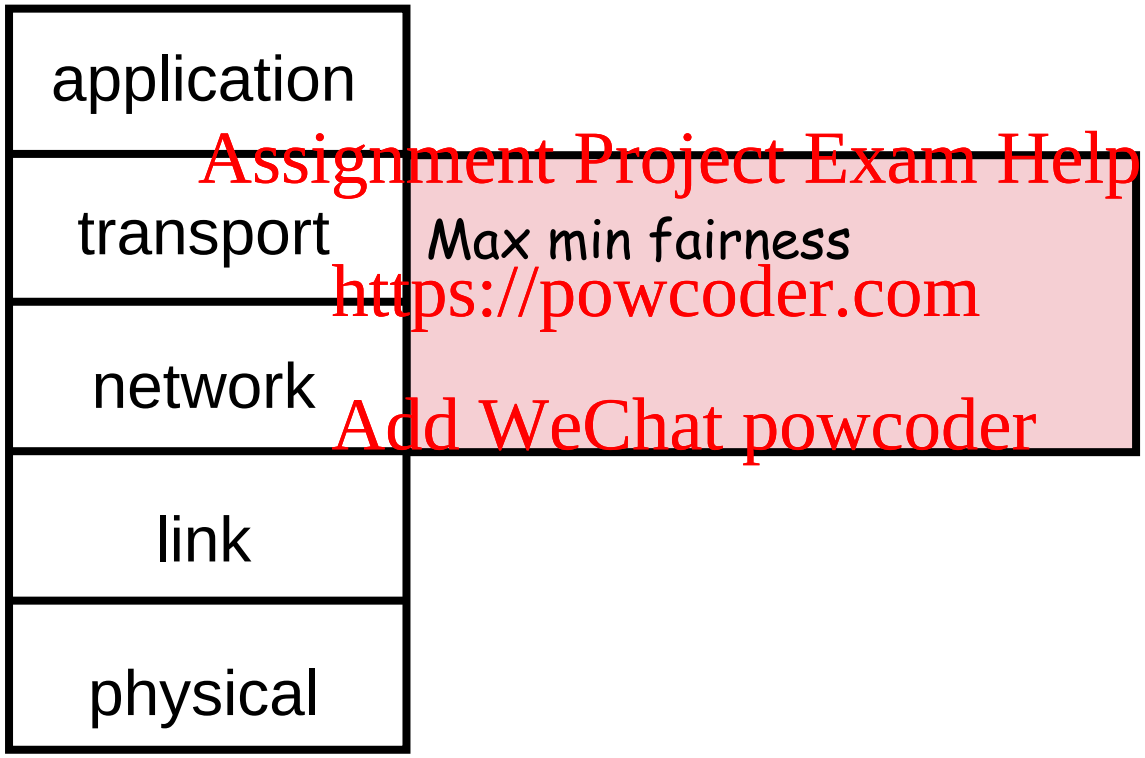
Stationary distribution

Average # of users

Average waiting time

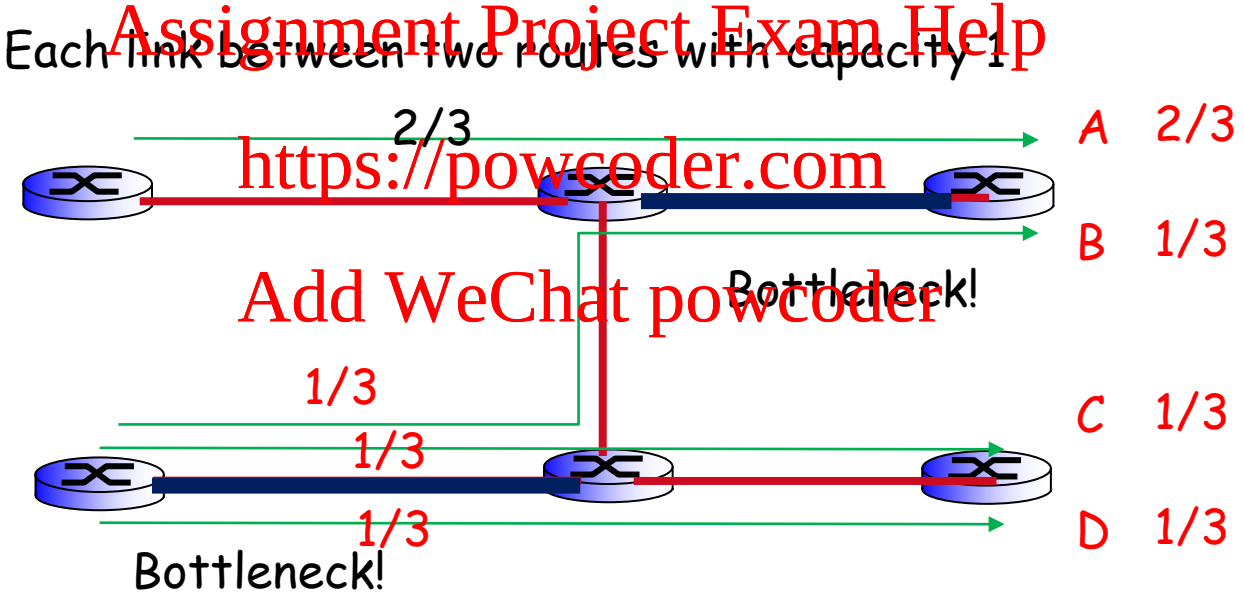
In final exam.

Internet Protocol Stack



Max-min fairness

How to judge if max-min fairness is satisfied.
How to find max-min fairness: Bottleneck approach

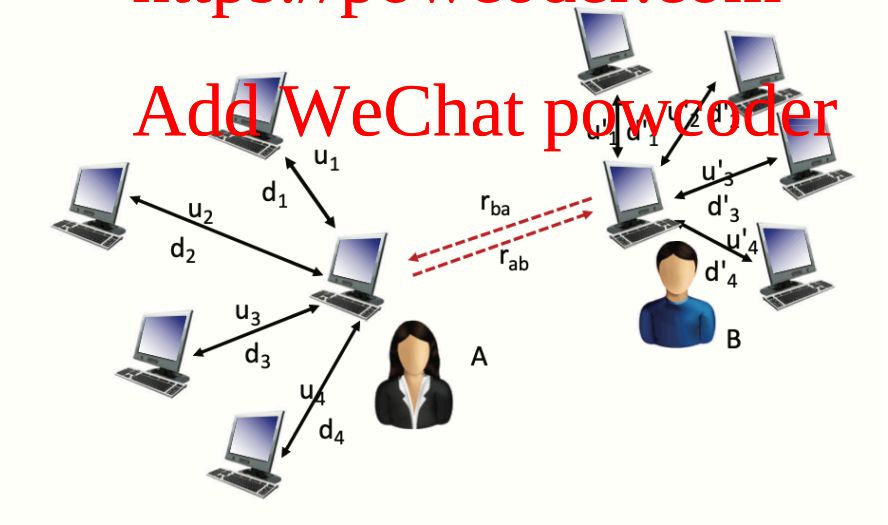


Assignment 1 common mistake

Question 4 (P2P Tit-for-Tat, 18%)**.** As shown in the figure below, A and B are communicating with their top-4 partners in a BitTorrent system. A's uploading and downloading data rates of the ith partner are u_i and d_i respectively; B's uploading and downloading data rates of the ith partner are u'_i and d'_i respective. For $i = 1, 2, 3, 4$, u_i, u'_i, d_i, d'_i are randomly distributed. They are independent random variables, following uniform distribution in $[0, 1]$ Mbps.

Now A optimistically unchoked B, with a sending data rate of r_{ab}. If A becomes a top-4 sender of B, B will start to serve A with a sending data rate of r_{ba}. r_{ab}, r_{ba} are independent random variables, following uniform distribution in $[0, 1]$ Mbps.

What is the probability that both A and B find each other a top-4 sender? Show your mathematical derivations.



Correct solution

Let $D = \min(d_1, d_2, d_3, d_4)$. We first need to find the cdf of D. Let $x \in [0, 1]$

$$
\begin{aligned}
& cdf_D(x) && (1)\\
=& P(D \leq x) && (2)\\
=& 1 - P(\min(d_1, d_2, d_3, d_4) > x) && (3)\\
=& 1 - P(d_1 > x)P(d_2 > x)P(d_3 > x)P(d_4 > x) && (4)\\
=& 1 - (1 - x)^4 && (5)
\end{aligned}
$$

where the last step is because $P(d_i > x) = 1 - x$ as d_i is uniformly distributed in $[0, 1]$

Therefore, pdf of D is

$$pdf_D(x) = 4(1 - x)^3, x \in [0, 1] \quad (6)$$

Now, we compare r_{ba} and D, to get probability $P(r_{ba} > D)$.

$$
\begin{aligned}
& P(r_{ba} > D) && (7)\\
=& \int_0^1 \int_0^1 pdf_D(x) pdf_{r_{ba}}(y) \mathbf{1}(x < y) dy dx && (8)\\
=& \int_0^1 \int_0^y pdf_D(x) pdf_{r_{ba}}(y) dx dy && (9)\\
=& \int_0^1 \int_0^y 4(1 - x)^3 dx dy && (10)\\
=& \frac{4}{5} && (11)
\end{aligned}
$$

The probability that r_{ba} is larger than the min of the current four providers of A is $P(r_{ba} > \min(d_1, d_2, d_3, d_4)) = \frac{4}{5}$. For the same reason, the probability that r_{ab} is larger than the min of the current four providers of B is $P(r_{ab} > \min(d'_1, d'_2, d'_3, d'_4)) = \frac{4}{5}$. Since the above two events are independent, the probability that $P(r_{ba} > \min(d_1, d_2, d_3, d_4), r_{ab} > \min(d'_1, d'_2, d'_3, d'_4)) = \frac{16}{25}$.

Common mistake

$$P(r_{ab} \geq \min(d'_1, d'_2, d'_3, d'_4) \text{ and } r_{ba} \geq \min(d_1, d_2, d_3, d_4))$$
$$= [1 - P(d'_1 > r_{ab})P(d'_2 > r_{ab})P(d'_3 > r_{ab})P(d'_4 > r_{ab})]$$
$$* [1 - P(d_1 > r_{ba})P(d_2 > r_{ba})P(d_3 > r_{ba})P(d_4 > r_{ba})]$$

$$P(r_{ab} \geq \min(d'_1, d'_2, d'_3, d'_4) and r_{ba} \geq \min(d_1, d_2, d_3, d_4))$$
$$= [1 - 0.5^4]^2$$

Why is it wrong?

Common mistake

$$P(r_{ab} \geq \min(d_1', d_2', d_3', d_4') \text{ and } r_{ba} \geq \min(d_1, d_2, d_3, d_4))$$
$$= [1 - P(d_1' > r_{ab})P(d_2' > r_{ab})P(d_3' > r_{ab})P(d_4' > r_{ab})]$$
$$* [1 - P(d_1 > r_{ba})P(d_2 > r_{ba})P(d_3 > r_{ba})P(d_4 > r_{ba})]$$

These are not independent events!

$$P(r_{ab} \geq \min(d_1', d_2', d_3', d_4') and r_{ba} \geq \min(d_1, d_2, d_3, d_4))$$
$$= [1 - 0.5^4]^2$$

P(A and B) = P(A) P(B) is true for independent events.

$d_1' > r_{ab}$ is happens, -> more likely r_{ab} is small -> more likely $d_2' > r_{ab}$ is also true.

Q: Could you give an example when the above approach is correct?
A: r_{ab} is a constant. d_1', d_2', d_3', d_4' are independent and thus $d_1' > r_{ab}$, $d_1' > r_{ab}$, $d_3' > r_{ab}$, $d_4' > r_{ab}$ are independent!

Another correct way

Because r_{ab}, d_1', d_2', d_3', d_4' are continuous random variable, and **independent** and they follow the **same distribution**, (i.i.d. independent and identically distributed), so that they have the same probability, i.e., 1/5, to be the smallest one.

Therefore $P(r_{ab} > \min(d_1', d_2', d_3', d_4')) = 4/5$

Simple way to verify 4/5

verifyq4.py - /Users/weibao/Dropbox (Sydney Uni)/2 Teachings COMP/0 2020/COMP5416/ass...

```
import random
T=100000
count=0
for l in range(T):
    r=[]
    for i in range(4):
        a=random.uniform(0, 1)
        r.append(a)


    R=min(r)
    d=random.uniform(0, 1)
    if d>R:
        count=count+1

print(count/T)
```

Ln: 17 Col: 0

Python 3.7.4 Shell

```
Python 3.7.4 (v3.7.4:e09359112e, Jul  8 2019, 14:36:03)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
 RESTART: /Users/weibao/Dropbox (Sydney Uni)/2 Teachings COMP/0 2020/COMP5416/as
signment1 prepare/solutions prepare/verifyq4.py
0.7997
>>>
```

Ln: 7 Col: 4

Final exam

- The marks of final exam sum up to 100 and it is worth **60%** of your overall mark.
- Online, open book, (type C)
- 130 minutes + buffer time + upload time
- Double-pass policy.
- 7 questions in total.
  - Calculation, short answer and extended response
- Type your answers in the blank below, or write down, scan/photograph, and upload in the end.
- Spend time wisely. Question 1 doesn't mean easiest.

Final exam

- No programming questions
- No Wireshark questions

Office hour

- By appointment
  - wei.bao@sydney.edu.au
  - zhengjie.yang@sydney.edu.au
  - zwan5430@uni.sydney.edu.au
- Assignment 2 common mistakes and Q&A session
  - 4-Dec-2020 (Fri), 3pm (tentative), Zoom
  - Non-compulsory, no recording
- Last-chance office hour
  - 7-Dec-2020 (Mon), 3pm (tentative), Zoom
  - Non-compulsory, no recording

Survey

- **Unit of Study Surveys (USS) for Semester 2 are now open!**
- Login to the University's Student Survey System now to complete a survey:
- **https://student-surveys.sydney.edu.au/students**
- Survey completed will give them an entry into a prize draw to win a range of Apple products including a 64gb Apple iPad Air, an Apple Watch and JB HiFi Gift Cards