|  |
| --- |
| CS 455 |
| Homework #2 |
|  |

|  |
| --- |
| hong kebi  10-19-2015 |

Kebi Hong

CS 455

Homework #2

10/19/15

**Question 1.**

**Problem P9 (15p). Find the total average response time.**

The average object size = 850,000 bits

The average request rate = 16 request / sec = arrival rate = β

The average internet delay = 3 sec

The transmission rate = 15 Mbps (according to the graph)

Average time request to send = ∆ = average object size / transmission rate

= 850,000 / 15,000,000 = 0.0567 sec

Average access delay = ∆ / (1 - ∆β)

= 0.0567 / (1 – 0.0567 \* 16) = 0.6 sec

Average response time = average access delay + average internet delay

= 0.6 + 3 = 3.6 sec

**# Find the total response time**

Traffic intensity = ∆β = 0.0567 \* 16 = 0.907

Because we have a cache that has a miss rate 0.4, then the traffic intensity is only reduced by 60%. Because for 60% of the time, we can find the data in the cache.

Cache hits: response time = ∆ / (1 - ∆β) = 0.0567 / (1 – 0.4 \* 0.907) = 0.089 sec

Cache miss: response time = 0.089 + 3 = 3.089 sec

Total response time = 0.4 \* 3.089 + 0.6 \* 0.089 = 1.29 sec

**Problem P10 (15p).**

Because it is only a 10-meter link, the propagation speed is really small that can be ignored.

**Non-persistent HTTP with parallel connection:**

We need one RTT the set up the TCP connection and another RTT to get the request and one transmission time to get the data.

Original transmission speed = 150 bits/sec

Transmission speed for 10 parallel connection = 150 / 10 = 15 bits/ sec

Size of control message = 200 bits

Size of downloaded object = 100,000 bits

Total time = 200 / 15 + 200 / 15 + 200/ 15 + 100,000/ 15 = 6706.7 sec

**Persistent HTTP with pipelining:**

In this way, the client will send 10 request at same time and get 10 data back separately

Transmission time = 150 bits/sec

Size of control message = 200 bits

Size of downloaded object = 100,000 bits

Total time = 10\*(200/150) + 10\* (100,000 / 150) = 6680 sec

**Conclusion:**

**As we can see, the total time cost for the two strategy is very close. Persistent HTTP with pipelining is not significantly faster than non-persistent case with parallel connection.**

**Problem P22 (10p).**

For client- server, the minimum distribution time = max {NF /us, F/ }

For P2P, the minimum distribution time = max {F/us, F/, NF/ (us +)}

F = 15 Gbit = 15000 Mbits

Us = 30 Mbps

= 2 Mbps

Note: 1 Gbit = 1000 Mbit = 1,000,000 Kbits = 1,000,000,000 bits

Client-Server

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10 | 100 | 1,000 |
| 300 Kbps (0.3) | 7500 | 50000 | 500000 |
| 700 Kbps (0.7) | 7500 | 50000 | 500000 |
| 2 Mbps | 7500 | 50000 | 500000 |

Peer-to-peer

|  |  |  |  |
| --- | --- | --- | --- |
|  | 10 | 100 | 1000 |
| 300 Kbps (0.3) | 7500 | 25000 | 45454.5 |
| 700 Kbps (0.7) | 7500 | 15000 | 20547.9 |
| 2 Mbps | 7500 | 7500 | 7500 |

**Problem P26 (5p).**

Part a. Bob’s claim is possible. Using a peer-to-peer strategy, every client can become a server and serve other computer, and there is no requirement that client must share their resource. As long as there is a peer with completed file in the BitTorrent torrent, Bob can download the file he needs from that peer.

Part b. Bob’s idea is correct. Bob can operate several computers at the same time, and use the same strategy (free-riding) to get the resource at the same time. Each computer can ask for the different part of the data from the server and save the downloading time.

**Problem P28 (5 p).**

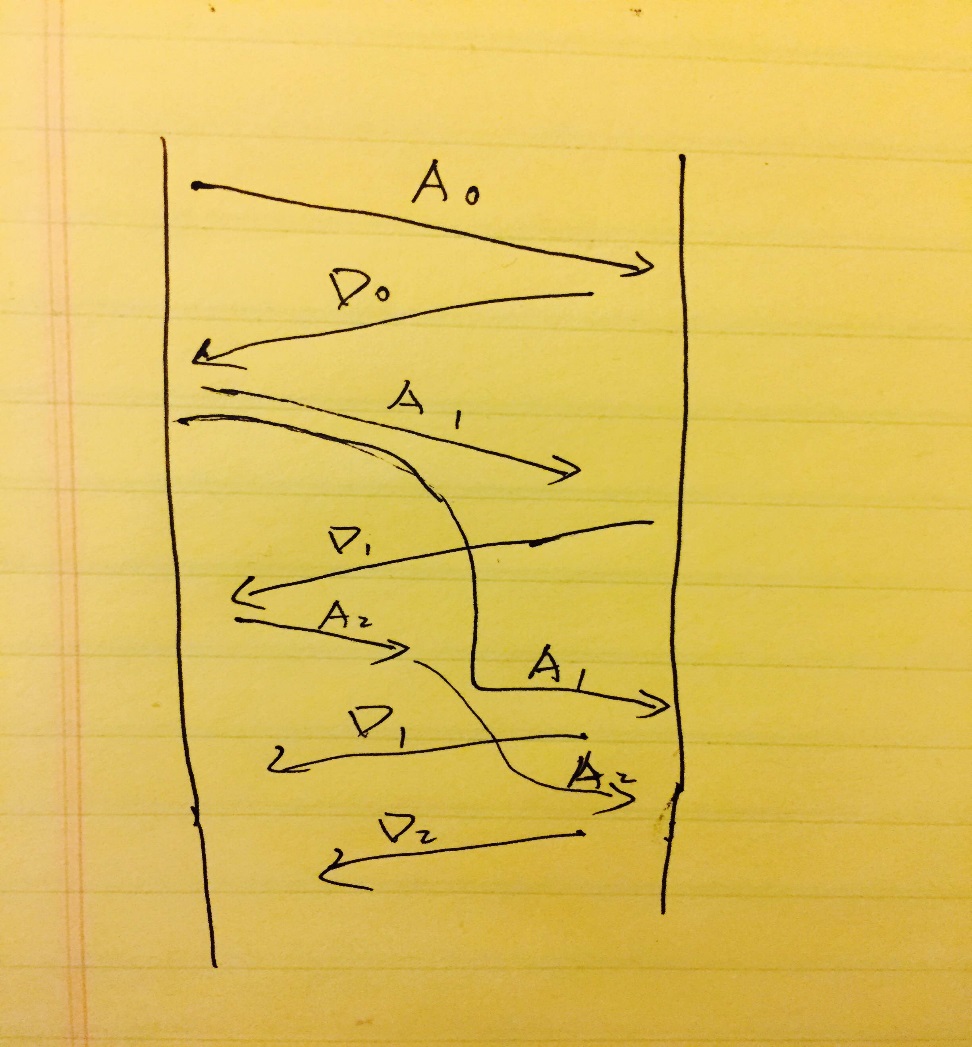
Because peer 6 only knows peer 15, peer 6 will send peer 15 a message that “who are peer 6’s predecessor and successor”. No one knows the answer and the message keeps passing alone until it reaches peer 5. Peer 5 knows that peer 6’s predecessor is peer 5 and successor is peer 8. Peer 5 pass this message to peer 6. Then peer 6 joins the DHT and makes peer 8 his successor. At last, peer 5 changes his successor to peer 6.

**Question 2.**

**Problem P6 (5 p).**

Assume that the sender is on “Wait for call 1 from above” and the receiver is in state “wait for 1 from below.” Then sender sends a message with sequence number 1 and change to state “Wait for ACK or NAK 1.” Receiver receives the message and send back an ACK with sequence 1 and change to state “Wait for 0 from below.” Unfortunately, the ACK message is corrupted. When the sender gets the corrupted ACK message, it resend the whole message with sequence number 1. However the receiver is waiting for the message with sequence number 0, and it will only sends the NAK when it doesn’t get the message with sequence number 0. Sender and receiver are waiting for the message with different sequence number and they will never be able to get it.

**Problem P13 (5p).**



**Problem P19 (15p).**

A (Sender):

* “Wait for B or C ACK.” In this state, A will set a timer separately for both B and C. when the timer expires, A will send another packet with a sequence number N to either B or C. If A receives an ACK from C, then it closes the timer for C and transits to “Wait for B ACK” state. . If A receives an ACK from B, then it closes the timer for B and transits to “Wait for C ACK” state. A will only transit to one of the two states, if A receives ACK at the same time, A will choose one ACK and ignore the other one.
* “Wait for B ACK.” In this state, A will set a timer for B and when the timer expires, A will send another packet with a sequence number N to B. If A receives ACK from B, A will close timer and transit to “Wait for new data” state.
* “Wait for C ACK.” In this state, A will set a timer for C and when the timer expires, A will send another packet with a sequence number N to C. If A receives ACK from C, A will close timer and transit to “Wait for new data” state.
* “Wait for new data.”

If A gets a request that ask A to send packet to B and C, A send B and C a packet with sequence number N, and transit to “Wait for B or C ACK” state.

B (receiver):

* If B receives a packet from A with a sequence number N, and N equals to the sequence number X in this state. B will send the ACK with sequence number X and update the sequence number X to X+1. If B receives a packet from A with a sequence number N, and N does not equal to the sequence number X in this state. B will send the ACK with sequence number X.

C (receiver):

* If C Receives a packet from A with a sequence number N, and N equals to the sequence number X in this state. C will send the ACK with sequence number X and update the sequence number X to X+1. If C receives a packet from A with a sequence number N, and N does not equal to the sequence number X in this state. C will send the ACK with sequence number X.

**Problem P21 (15P).**

FSM description:

A (the request):

* “Wait for request 0 from above.”

A is waiting for a call from above to request the next data from B. When A receives a request from above, it sends a request message R0 to B, sets a timer and makes a transition to the “Wait for D0” state. When A in this state, A ignores anything it receives from B.

* “Wait for D0.”

The timer is set from the last state, and when the timer expires, A will send another R0 message to B, then restart the timer. If A receives D0, A stops timer and make a transition to “Wait for Request 1 from above” state. If A receives a D1, A will ignore the message and keeps stay in this state.

* “Wait for request 1 from above.”

A is waiting for a call from above to request the next data from B. When A receives a request from above, it sends a request message R1 to B, sets a timer and makes a transition to the “wait for D1” state. When A in this state, it ignores anything receives from B.

* “Wait for D1.”

The timer is set from the last state, and when the timer expires, A will send another R1 message to B, then restart the timer. If A receives D1, A stops timer and make a transition to “Wait for Request 0 from above” state. If A receives a D0, A will ignore the message and keeps stay in this state.

B (Data sender):

* “Send D0.”

If B receives R0, it will send D0 to A. if B receives R1, then it knows that D0 has been received correctly, and transits to the “Send D1” state.

* “Send D1.”

If B receives R1, it will send D1 to A. if B receives R0, then it knows that D1 has been received correctly, and transits to the “Send D0” state.

**Question 3.**

Transmission time = 1000 / 1000,000 = 1 msec

The Round-Trip time = Time to send request + Time to receive + 2\* propagation delay

= 1 msec + 0 + 2\* 270 = 541 msec

The utilization for total of K packets sent over 541 mesc = k / 541

a). stop-and-wait

Sender can only send one packet at a time, therefore k = 1.

Utilization = 1/541 = 0.18%

b). go-back-n

Sender send n message at a time and wait for the ACK. For 3-bit sequence number, the max number of packets it can send is 7, therefore k = 7.

Utilization = 7/541 = 1.29%