Xiaoyu Xia homework 5

[xix90@pitt.edu](mailto:xix90@pitt.edu)

1.

(a)

Assume that the window size for both sender and receiver is 4, and the sequence space is less than 2\*4=8(2N), which is 7.

Assume that the sender sends 4 packages successfully and the receiver has already received these four packages, so we can slide the window of the receiver forward and wait for new data. But when the receiver sends ack0 to the sender, it is lost, so the sender has to resend packet 0（timeout）so that the window for the sender can slide forward.

At this time, the receiver's receiving window has been updated, and the receiver cannot distinguish between new data frames and old data frames with sequence number 0, and ambiguity occurs.

图示

描述已自动生成

(b)

Assume that the window size for sender and receiver is 4, and the sequence space is less than 4+1=5(N+1), which is 4.

Assume that the sender sends 4 packages successfully and the receiver has already received these four packages one by one. But when the receiver sends ack0 to the sender, it is lost, so the sender has to resend packet 0（timeout） so that the window for the sender can slide forward.

At this time, the receiver cannot distinguish between new data frames and old data frames with sequence number 0, and ambiguity occurs.

图示, 文本

描述已自动生成

2.

(1) the timeout interval after 10 RTT measurements is 336.884ms.

电脑萤幕的截图

描述已自动生成

(2)

As the picture below shows: When the n is 29, the timeout interval falls below 50ms which is exactly 49.70ms.

电脑萤幕的截图

描述已自动生成

3.

(a)

If the situation is lossless transmission: the total time for receiving an ack is an RTT, and then we can slide its window forward and transmit new data.

So the total time is T 下標 1 等於 R T T 等於 300 m s.

When we lose a segment, first the sender sends this package and receives ack:

same time as T 下標 1 等於 R T T 等於 300 m s.

Secondly, upon receipt of the third duplicate ACK, the fast retransmit algorithm detects the loss and retransmits the lost segment. So after the sender sends segments, it will receive 3 duplicate acks which take 30ms. So T 下標 2 等於 T 下標 1 加 3 乘號 T 下標 s e g m e n t s 結束下標 等於 300 加 30 左小括號 m s 右小括號 等於 330 空格 m s

When the sender detects the loss, it will retransmit the lost segment and receive the ack from the receiver, which costs one RTT.

So the total time T 下標 3 等於 T 下標 2 加 R T T 等於 330 加 300 左小括號 m s 右小括號 等於 630 空格 m s

**Finally, the sender loses 330ms (630-300ms=330ms) compared with the lossless transmission.**

(b).

If we use the timeout principle, so when timeout the sender retransmits the lost segment and receives the ack from the receiver, which costs one RTT.

So the total time under this situation is: T 下標 4 等於 T i m e o u t 加 R T T 等於 400 加 300 左小括號 m s 右小括號 等於 700 空格 m s

We can compare both situations：T 下標 4 減 T 下標 3 等於 700 減 630 左小括號 m s 右小括號 等於 70 空格 m s

**We can save 70ms if we use fast retransmit.**

4.

(a)

If our congestion window at some point in time is W bytes, then we are allowed to send W bytes per RTT.

So max throughput at that time is .

So, t h r o u g h t p u t 等於 100 M b p s 等於 分數 R T T 分之 m a x 空格 w i n d o w 空格 s i z e 乘號 T C P 空格 s e g m e n t 空格 s i z e 空格 乘號 8 結束分數

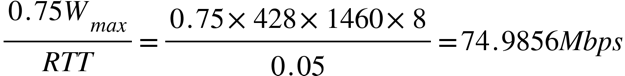
空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 空格 等於 分數 0.05 分之 m a x 空格 w i n d o w 空格 s i z e 空格 乘號 1460 乘號 8 結束分數
m a x 空格 w i n d o w 空格 s i z e 等於 分數 1460 乘號 8 分之 0.05 乘號 100 乘號 10 的 6 次方 結束分數 等於 428.08 几乎等于 428

(b)

Ignore slow start, assume we are always in congestion avoidance:

Window increases linearly from  to W 下標 m a x 結束下標 until loss occurs and process repeats.

So the average window size is (+W 下標 m a x 結束下標)/2=0.75W 下標 m a x 結束下標

Average throughput is 

5.

(a)

Parallel connection is separating TCP connection per object but runs them in parallel.

And the TCP fairness goal is: if K TCP sessions share the same bottleneck link of bandwidth R, each should have an average rate of R/K.

When we have a parallel connection, we can get a larger bandwidth compared to one TCP connection.

(b)

1. Allows the client to open multiple connections and execute multiple HTTP tasks in parallel.

2. Ability to increase page loading speed, overcoming dead time and bandwidth limitations of single-hop connections.