



# CSE 322

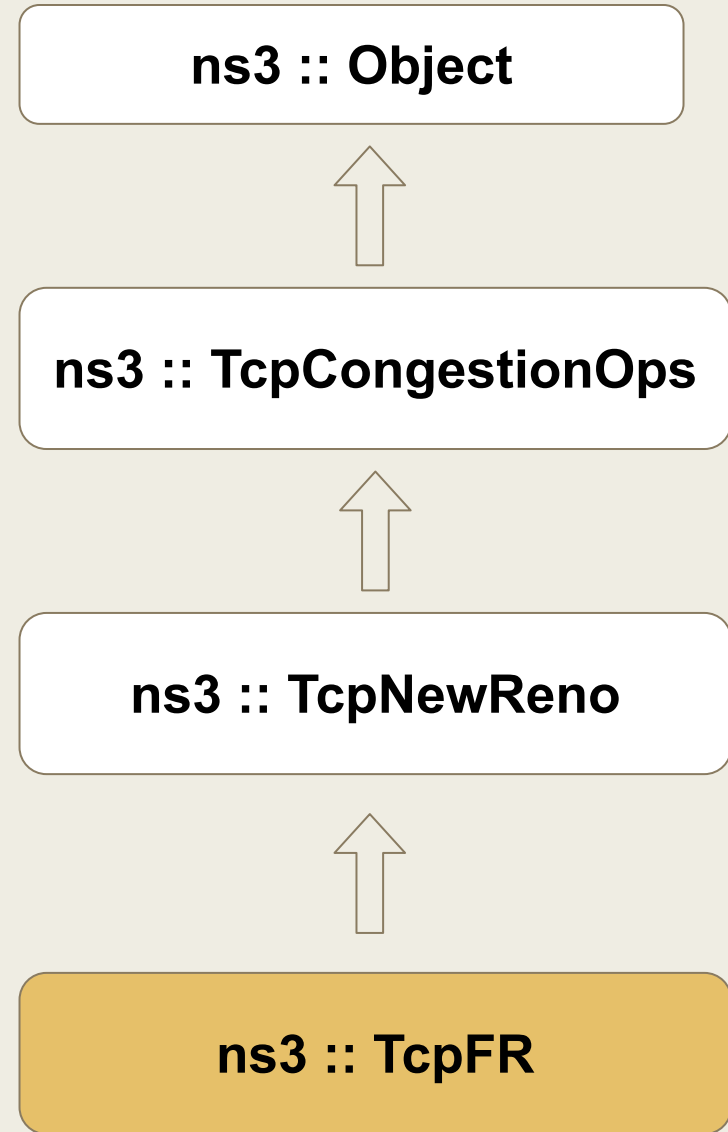
(TCP with Faster Recovery)

1705036

Ayan Antik Khan



# Class Hierarchy



MODIFICATIONS




# FR-TCP ( Bandwidth Estimation )

## Pseudocode

```
if (ACK is received) {  
    sample_BWE = pkt_size*8/(now - lastacktime);  
    BWE = BWE*alpha + sample_BWE*(1 - alpha);  
}
```

- **PktsAacked()** is called every time an ACK is received



```
1 void  
2 TcpCongestionOps::PktsAacked (Ptr<TcpSocketState> tcb, uint32_t segmentsAacked,  
3                               const Time& rtt)  
4 {  
5     NS_LOG_FUNCTION (this << tcb << segmentsAacked << rtt);  
6 }
```

# FR-TCP ( Bandwidth Estimation )

## Pseudocode

```
if (ACK is received) {  
    sample_BWE = pkt_size*8/(now - lastacktime);  
    BWE = BWE*alpha + sample_BWE*(1 - alpha);  
}
```

- **BW Estimation** is similar to that of **TCPWestwood**

```
1 void  
2 TcpWestwood::EstimateBW (const Time &rtt, Ptr<TcpSocketState> tcb)  
3 {  
4     NS_LOG_FUNCTION (this);  
5  
6     NS_ASSERT (!rtt.IsZero ());  
7  
8     m_currentBW = m_ackedSegments * tcb->m_segmentSize / rtt.GetSeconds ();  
9  
10    if (m_pType == TcpWestwood::WESTWOOD)  
11    {  
12        Time currentAck = Simulator::Now ();  
13        m_currentBW = m_ackedSegments * tcb->m_segmentSize / (currentAck - m_lastAck).GetSeconds ();  
14        m_lastAck = currentAck;  
15    }
```

# FR-TCP ( Set *Cwin* & *ssthresh* )

## Pseudocode

- triple duplicate ACKS:  
 $ssthresh = (BWE * RTT_{min}) / a$   
 $CWIN = ssthresh$
- coarse timeout expiration:  
 $ssthresh = (BWE * RTT_{min}) / a$   
 $CWIN = 1$

- **BWE** available from before
- **Rtt<sub>min</sub>** can be calculated
- **Get** “triple dupack” or “timeout” from **m\_congstate()** & **m\_ecnState()**

```
1 void
2 TcpNewReno::IncreaseWindow (Ptr<TcpSocketState> tcb, uint32_t segmentsAacked)
3 {
4     NS_LOG_FUNCTION (this << tcb << segmentsAacked);
5
6     if (tcb->m_cWnd < tcb->m_ssThresh)
7     {
8         segmentsAacked = SlowStart (tcb, segmentsAacked);
9     }
10
11     if (tcb->m_cWnd >= tcb->m_ssThresh)
12     {
13         CongestionAvoidance (tcb, segmentsAacked);
14     }
```

# GFR-TCP ( Congestion Avoidance )

## Pseudocode

```
If (CWIN > ssthresh) AND (CWIN < BWE*RTT_min)
then
    ssthresh += (BWE*RTT_min-ssthresh)/2;
```

- Check condition every **500ms** (Standard TCP Value)
- Recompute *ssthresh* on **True**

```
1 void
2 TcpNewReno::CongestionAvoidance (Ptr<TcpSocketState> tcb, uint32_t segmentsAacked)
3 {
4     NS_LOG_FUNCTION (this << tcb << segmentsAacked);
5
6     if (segmentsAacked > 0)
7     {
8         double adder = static_cast<double> (tcb->m_segmentSize * tcb->m_segmentSize) / tcb->m_cWnd.Get ();
9         adder = std::max (1.0, adder);
10        tcb->m_cWnd += static_cast<uint32_t> (adder);
11        NS_LOG_INFO ("In CongAvoid, updated to cwnd " << tcb->m_cWnd <<
12                     " ssthresh " << tcb->m_ssThresh);
13    }
14 }
```

THANK YOU!