Forward Explicit Congestion Notification (FECN) for Datacenter Networks
Pseudocode Version 1.0

March 22, 2007

**Per-Port Switch Variables:**
Nominal Capacity of the Port: $C_0$
Advertised Rate for the $i^{th}$ interval: $r_i$
Port State: Port\_Initialized
Measured Port Capacity for $i^{th}$ Measurement Interval: $C_i$
Bits received in the last measurement interval: $\text{bits\_received}$
Bits transmitted in the last measurement interval: $\text{bits\_sent}$
Bits in Queue at the beginning of current measurement interval: $qlen$
Total Busy (transmit) time during last measurement interval: $\text{time\_busy}$
Limited rate increase: $\Delta r$

**Algorithm Parameters:**
Initial Rate Divider: $N_0$
Measurement Interval in Switches: $T$
$T$ is also used for $\tau$-timer (probe interval) in the sources
Queue Control Parameters: $a, b, c$
Exponential moving average: $\alpha$

**Operational Parameters:**
Queue equilibrium: $Q_{eq}$
Queue Length under severe congestion: $Q_{sc}$

**Default Values:**
$N_0 = 200, T = 1 \text{ ms},$
$Q_{eq} = 192,000 \text{ bits}, a = 1.1, b = 1.002, c = 0.1, \alpha = 0.5$

**Variable Initialization:**
$C_i = C_0, \Delta r = r_{i-1} = r_{i-2} = \frac{C_0}{N_0}$
Switch Event Routines:

procedure EVENT T_Timer_Expired_Routine()
1: while T_timer expires do
2: if Port_Initiaized then
3:    Estimate_Capacity()
4:    if bits_received > 0 then
5:        Rate_Allocation(qlen, bits_received, r_{i-1}, r_{i-2}, C_i)
6:    else
7:        r_i = \frac{C_i}{N_0}
8:    end if
9: else
10:    if bits_received > 0 then
11:        Port_Initiaized = TRUE
12:        Rate_Allocation(qlen, bits_received, r_{i-1}, r_{i-2}, C_i)
13:    end if
14: end if
15: Reset T_timer
16: end while

procedure EVENT Transmit_Packet()
1: if packet_is_tagged then
2: if Rate_in_Tag \geq r_i OR Rate_in_Tag == -1 then
3:    Rate_in_Tag = r_i
4: end if
5: end if
6: Forward the packet

Subroutines:

procedure Rate_Allocation(qlen, bits_received, r_{i-1}, r_{i-2}, C_i)
1: Queue_Control(f_q, qlen, Q_eq)
2: \rho = \frac{bits_{received}}{T \times C_i \times f_q}
3: if \rho == 0 then
4: if qlen < 1 then
5:    r_i = C_i
6: else
7:    r_i = r_{i-1}
8: end if
9: else

Port_Initiaized = FALSE.
\[ r_i = \min \{ C_i, \frac{r_{i-1}}{\rho} \} \]

10: if
11: Exponential_Averaging\( (r_i, r_{i-1}, r_{i-2}) \)
12: Limit_Rate_Increase\( (r_i, r_{i-1}, qlen, Q_{eq}, \Delta r) \)
13: Variable_Capacity_Adjustment\( (C_i, C_{i-1}, r_i, r_{i-1}, r_{i-2}) \)
14: bits_received = 0
15:

\begin{verbatim}
procedure Queue_Control\( (f_q, qlen, Q_{eq}) \)
1: if \( qlen \leq Q_{eq} \) then
2: \[ f_q = \frac{b \times Q_{eq}}{(b-1) \times qlen + Q_{eq}} \]
3: else
4: \[ f_q = \max \left\{ c, \frac{a \times Q_{eq}}{(a-1) \times qlen + Q_{eq}} \right\} \]
5: end if
Note: Queue control function can be implemented as a precomputed table
\end{verbatim}

\begin{verbatim}
procedure Exponential_Averaging\( (r_i, r_{i-2}) \)
1: \[ r_i = \alpha r_i + (1 - \alpha) r_{i-2} \]
\end{verbatim}

\begin{verbatim}
procedure Limit_Rate_Increase\( (r_i, r_{i-1}, qlen, Q_{eq}, \Delta r) \)
1: if \( qlen < Q_{eq} \) then
2: \[ \Delta r = 1.414\Delta r \]
3: else if \( qlen > Q_{eq} \) then
4: \[ \Delta r = 0.707\Delta r \]
5: end if
6: if \( r_i - r_{i-1} > \Delta r \) then
7: \[ r_i = r_{i-1} + \Delta r \]
8: end if
\end{verbatim}

\begin{verbatim}
procedure Variable_Capacity_Adjustment\( (C_i, C_{i-1}, r_i, r_{i-1}, r_{i-2}) \)
1: if \( C_i < C_{i-1} \) then
2: \[ r_i = \frac{C_i}{C_{i-1}} r_i \]
3: \[ r_{i-1} = \frac{C_i}{C_{i-1}} r_{i-1} \]
4: end if
5: \[ r_{i-2} = r_{i-1} \]
6: \[ r_{i-1} = r_i \]
\end{verbatim}
procedure Estimate_Capacity($C_i, bits\_sent, time\_busy$)
1: \[ C_i = \frac{bits\_sent}{time\_busy} \]
2: \[ bits\_sent = 0 \]
3: \[ time\_busy = 0 \]

Source Variables:
Current Rate: $R$
Source Parameters: $\tau = T$
Feedback\_Valid\_Interval: $2T$
Initial Rate: $R_0 = \frac{C_0}{N_0}$

Source Initialization:
Set $\tau$\_timer to expired
Tagging a sampled packet every $\tau$ sec
Set Feedback\_Valid\_Timer to expired

Source Event Routines:
procedure EVENT Send\_packet()
1: if $\tau$\_timer expired then
2: \[ if \ Text{Feedback\_Valid\_timer expired} \ then \]
3: \[ R = R_0 \]
4: \[ end \ if \]
5: Tag an outgoing packet and the rate field in Tag is set as -1;
6: Send following packets at current rate $R$
7: Reset $\tau$\_timer
8: else
9: Send following packet without tag at rate $R$
10: end if

procedure EVENT FECN\_Received()
1: Set $R$ to $rate\_received$
2: Reset Feedback\_Valid\_Timer to $2T$

Note: This pseudocode is continuously evolving. If you plan to use this code for analysis or implementation, please contact Jain@wustl.edu for the latest version of the pseudocode.