98-0408: Overload Based Explicit Rate Switch Schemes with MCR Guarantees

Bobby Vandalore, Sonia Fahmy, Raj Jain, Rohit Goyal,

Raj Jain is now at
Washington University in Saint Louis
Jain@cse.wustl.edu
http://www.cse.wustl.edu/~jain/
Overview

- General Fairness Definition
- Overview of ERICA+
- Overload Based Algorithms
- Simulation Configurations
- Simulation Results
- Comparisons of Algorithms
- Conclusions
General Fairness

- Define following:
  - \( A_1 \) = Total available bandwidth
  - \( A_b \) = Sum of bandwidth of underloaded connections
  - \( A = A_1 - A_b \), excess bandwidth
  - \( N_a \) = Number of active connections
  - \( N_b \) = Number of active connections bottlenecked elsewhere
  - \( n = N_a - N_b \), number of active connections bottlenecked on this link
General Fairness (Cont)

- $M = \text{Sum of MCRs of active connections}$
- $B(i) = \text{Generalized Fair allocation for connection } i$
- $MCR(i) = \text{MCR of connection } i$
- $w(i) = \text{pre-assigned weight associated with VC } i$
- $\text{FairShare}$
  \[ B(i) = MCR(i) + \frac{w(i)(B - M)}{\sum_{j=1,n} w(j)} \]
ERICA Scheme: Basic

- **Explicit Rate Indication for Congestion Avoidance**
- Set target rate, say, at 95% of link bandwidth
  
  \[ \text{ABR Capacity} = \text{Target Utilization} \times \text{Link Bandwidth} \]

- Monitor input rate and number of active VCs
  
  \[ \text{Overload} = \frac{\text{ABR Input rate}}{\text{Target ABR Capacity}} \]

- This VC’s Share = VC’s Rate/Overload

- Fair share = Target rate/ Number of Active VCs

- \[ \text{ER} = \max \{ \text{Fair share, MaxAllocPrevious, VC’s Rate/Overload} \} \]

- \[ \text{MaxAllocCurrent} = \max \{ \text{MaxAllocCurrent, ER} \} \]
Activity Level

- $AL(i) = \text{Min}\{1, \text{VC’s Rate/FairShare}\}$
- Effective # of Active VCs = $\Sigma AL(i)$
- FairShare = ABR Capacity/Effective # of Active VCs
- Recursive definition.
  Converges in just a few iterations.
New Algorithms

- ER = Max{FairShare, MaxAllocPrevious, VC’s Rate/Overload}
- If FairShare is based on effective number of active VCs, we do not need all three terms ⇒ Four algorithms
  - A: ER = Max{FairShare, VC’s Rate/Overload}
  - B: ER = FairShare/overload
  - C: ER = MaxAllocPrevious/overload
  - D: ER = Max{MaxAllocPrevious, VC’s rate/Overload}

Detailed pseudo-codes in the contribution.
3 Sources. Unidirectional traffic

MCRs of (10, 30, 50) Mbps were used.

Excess bandwidth (149.76 - 90) = 59.76 was shared equally to achieve an allocation of (29.92, 49.92, 69.92)
3 sources. Source 2 is transient.

MCRs were zero for all sources. Simulation time 1.2 s. Source 2 is active (0.4, 0.8s). Allocation was (74.8, 0, 74.8) during (0, 0.4s) and (0.8, 1.2s) and (49.92, 49.92, 49.92) during (0.4, 0.8s)
Configuration 3

Source Bottleneck configuration

- Source S1 is bottlenecked at 10 Mbps for first 0.4 s (i.e., it sends data at a rate of \( \min\{10 \text{ Mbps, ACR}\} \))
- MCRs = \{10, 30, 50\} Mbps
- Fair Allocation = \{39.86, 59.86, 79.86\} during (0, 0.4s) and \{29.92, 49.92, 69.92\} during (0.4, 0.8s).
Configuration 4

- Generic Fairness Config GFC-2 with D=1000 km
- MCRs of zero for all source were used.
  Simulation time 2.5 seconds.
- Allocation for each of (A, B, C, D, E, F, G, H) type VCs was (10, 5, 35, 35, 35, 10, 5, 52.5), respectively.
Table 1: Simulation Parameters

<table>
<thead>
<tr>
<th>Configuration Name</th>
<th>Link Distance</th>
<th>Averaging Interval</th>
<th>Target Delay</th>
<th>Wt Func</th>
</tr>
</thead>
<tbody>
<tr>
<td>Three Sources</td>
<td>1000 Km</td>
<td>5 ms</td>
<td>1.5 ms</td>
<td>1</td>
</tr>
<tr>
<td>Source Bottleneck</td>
<td>1000 Km</td>
<td>5 ms</td>
<td>1.5 ms</td>
<td>1</td>
</tr>
<tr>
<td>GFC-2</td>
<td>1000 Km</td>
<td>15 ms</td>
<td>1.5 ms</td>
<td>1</td>
</tr>
</tbody>
</table>

- Exponential averaging of overload with decay factor of 0.8 was used for algorithms A and D. B and C are more sensitive to variation, so decay factor of 0.4 was used.
Simulation Results

- Configuration 1: Three Sources
  - All algorithms achieved the generalized fairness allocation.

- Configuration 2: 3-Source Transient
  - All algorithms achieved the generalized fairness allocation.
  - Algorithm B has oscillations
Results (Contd)

- Configuration 3: Source Bottleneck
  - Algorithm A and B do not converge since they use CCR field for estimating source rate. If measured source rate was used A and B also converge.

- Configuration 4: GFC2
  - Algorithm B and D have rate oscillations due to queue control.
  - Algorithm C had large switch queue, since it uses maximum always.
# Comparison of Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>End of Interval</th>
<th>Feedback</th>
<th>Max Queue</th>
<th>PerVC SrcRate</th>
<th>Sensitive to Queue control</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>$O(N)$</td>
<td>$O(1)$</td>
<td>Medium</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>B</td>
<td>$O(N)$</td>
<td>$O(1)$</td>
<td>Medium</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>C</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>Large</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>D</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>Medium</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

- Algorithm D is the best
Algorithm A and B use activity levels. Need measured source rate in presence of source bottlenecks

Algorithm C based only on MaxAlloc can have large switch queues

Algorithm D based on VCs rate and MaxAlloc is the best algorithm.