A Quantitative Assured Forwarding Service

Nicolas Christin, Jörg Liebeherr, and Tarek F. Abdelzaher

University of Virginia
Department of Computer Science
P.O. Box 400740
Charlottesville VA, 22904-4740

[nicolas|jorg|zaher]@cs.virginia.edu
Outline

- Problem and Context
- Related Work
- The Quantitative Assured Forwarding Service
- Mechanisms for QAF
  - Feedback-control based algorithms
  - Implementation
- Evaluation
  - Service guarantees
  - Overhead
- Conclusions
Problem and Context

Challenge: Can we provide strong service guarantees with low computational complexity?
Related Work

- **Flow-Based Service Architectures**
  - FRED [Lin and Morris, 1997]
  - SCORE/CSFQ [Stoica and Zhang, 1998, 1999]

- **Class-Based Service Architectures**
  - Proportional Differentiated Services [Dovrolis et al., 1999, 2000, 2001]
  - Alternative Best-Effort [Hurley et al., 1999, 2000]
  - JoBS [Liebeherr and Christin, 2001]
  - C-DBP [Striegel and Manimaran, 2002]
Quantitative Assured Forwarding

- Guarantees provided on a per-hop, per-class basis
- No admission control, no signaling, no traffic conditioning
  - No per-flow operations
- Proportional and absolute per-class guarantees for both loss and delay and lower bound on throughput
  
  \[
  \frac{\text{Class-2 loss rate}}{\text{Class-1 loss rate}} \approx 2
  \]

  Class-2 delay $\leq 5$ ms

- Concession: service guarantees may need to be temporarily relaxed

None of the existing mechanisms can realize this service
Mechanisms for QAF: Overview

- **Key idea:** Manage the head and the tail of the transmission queue in a single algorithm
  - Combine buffer management and rate allocation

- **How can QAF be implemented?**
  - Service rate allocation to traffic classes, periodically adjusted
  - If no feasible rate allocation exists, drop traffic
  - Rate allocation and packet drop decisions use feedback control
  - If set of guarantees infeasible (no admission control), temporarily relax some guarantees
Arrivals, Departures, Losses at a Node

Class-i Traffic

Arrival Curve

Dropped Traffic

Delay $D_i(n)$

Output Curve

Backlog $B_i(n)$

$t(n)$ time

$A_i(n) = \sum_{k=0}^{n} a_i(k)$

$R_i^{in}(n) = A_i(n) - \sum_{k=0}^{n} l_i(k)$

$R_i^{out}(n) = \sum_{k=0}^{n} r_i(k) \cdot \Delta t(k)$

$B_i(n) = R_i^{in}(n) - R_i^{out}(n)$

$D_i(n) = t(n) - t\left(\max\{k < n \mid R_i^{out}(n) \geq R_i^{in}(k)\}\right)$

Christin, Liebeherr and Abdelzaher

*A Quantitative Assured Forwarding Service*
Combined Rate Allocation and Buffer Management

Class-i Traffic

Drop Traffic

Arrival Curve

Output Curve

Measured Delay

Rate adjustment

\[ \Delta r_i(n) \]

\[ l_i(n) \]

\[ t(n) \]

Present time
Feedback Loops

- Service rate allocation and loss rates can be viewed in terms of a recursion:

\[
\begin{align*}
    r_i(n) &= r_i(n-1) + \Delta r_i(n) \\
    p_i(n) &= p_i(n-1) \frac{A_i(n-1)}{A_i(n)} + \frac{l_i(n)}{A_i(n)}
\end{align*}
\]

- Feedback loops
Delay Feedback Loops

- One loop per class
- Proportional delay guarantees:
  - Each class $i$ delay is compared to a reference
  - Rate is adjusted by controller: $\Delta r_i(n) = K(n) \cdot e_i(n)$
- Absolute delay and rate guarantees:
  - $r_i(n) \geq \mu_i$  
  - $D_i(n) \leq d_i$

Christin, Liebeherr and Abdelzaher  
A Quantitative Assured Forwarding Service 
IEEE INFOCOM 2002
Absolute Delay and Rate Guarantees

- Limit the rate adjustment permitted:

\[ r_i(n) \geq r_{i,\text{min}}(n) \]

with

\[ r_{i,\text{min}}(n) = \max \left\{ \frac{B_i(n)}{d_i - D_i(n)}, \mu_i \cdot \chi_{B_i(n) \geq 0} \right\} \]

- Bound on \( K(n) \)

\[ K(n) \geq \max_i \left( \frac{r_{i,\text{min}}(n) - r_i(n-1)}{e_i(n)} \right) \]
Linearization and Stability

- System is intrinsically non-linear (delay = inverse of the rate)
- Can be linearized with following assumptions
  - Backlog does not change significantly during the time a particular arrival is backlogged
  - \( \Delta r_i(n) \ll r_i(n) \)
- Allows to derive bounds on \( K(n) \) for stability (i.e., convergence to proportional differentiation):
  \[
  0 \geq K(n) \geq -2 \cdot \min_i \left( \prod_{j \neq i} \frac{B_i(n)}{m_j \cdot D_j^2(n)} \right)
  \]
Properties of the Controller

- **Adjustment is simple:**
  - Compute bounds on $K(n)$
  - Compute errors $e_i(n)$
  - Multiply (for each class)

- **Work-conserving scheduler!**
  - Since $K(n)$ common to all classes:
    $$\sum e_i(n) = 0 \Rightarrow \sum \Delta r_i(n) = 0$$
Loss Feedback Loops

- No adjustment here, but three decisions to make
- When to drop
  - Buffer is full
  - Minimum capacity needed for service guarantees exceeds output link capacity
- Which class to drop from
  - Measure distance $e_i'(n)$ between target loss rate (for proportional loss differentiation) and loss rate of class $i$
  - Drop in increasing order of $e_i'(n)$
- How much traffic to drop
  - Drop as long as buffer is full or minimum capacity needed exceeds output link capacity
  - Stop dropping from a given class when absolute loss rate bound is reached
Implementation

- Implementation in FreeBSD kernel
  - Testbed of 6 Pentium IIIs 1Ghz with multiple interfaces
  - Allows testing at 100 Mbps (FastEthernet)
  - Developed for ALTQ 3.0 (package allowing modifications to the network stack), now part of ALTQ 3.1
Experimental Setup

100 Mbps, 200 pkts

<table>
<thead>
<tr>
<th>Class</th>
<th>No. of Flows</th>
<th>Proto.</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>UDP</td>
<td>On-off</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>TCP</td>
<td>Greedy</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>TCP</td>
<td>Greedy</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>TCP</td>
<td>Greedy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>$d_i$</th>
<th>$L_i$</th>
<th>$\mu_i$</th>
<th>$k_i$</th>
<th>$k'_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8 ms</td>
<td>1 %</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>35 Mbps</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Proportional Delay Differentiation

Router 1

Ratios of Delays

Time(s)

0 10 20 30 40 50 60

Class 4 Delay/Class 3 Delay

Router 2

Ratios of Delays

Time(s)

0 10 20 30 40 50 60

Class 3 Delay/Class 2 Delay

Class 4 Delay/Class 3 Delay
Absolute Delay Bounds

Router 1

Delay Bound

Router 2

Class-1 Delays

Delays (ms)

Time (s)

0 10 20 30 40 50 60

0 10 20 30 40 50 60

0 4 8 12

0 4 8 12

Christin, Liebeherr and Abdelzaher
A Quantitative Assured Forwarding Service
IEEE INFOCOM 2002
Proportional Loss Differentiation

Router 1

Class 4 Losses/Class 3 Losses

Router 2

Class 3 Losses/Class 2 Losses

Class 4 Losses/Class 3 Losses
Absolute Loss Rate Bounds

Router 1

Router 2

Loss Rate (%)

Class-1 Loss Rate

Time (s)
Throughput Differentiation

Router 1

- Throughput (Mbps)
- Time(s)
- Class 2
- Class 3
- Class 4

Router 2

- Throughput (Mbps)
- Time(s)
- Class 2 guarantee

Aggregate

Christin, Liebeherr and Abdelzaher
A Quantitative Assured Forwarding Service
IEEE INFOCOM 2002
Computational Overhead

- **Two functions:**
  - enqueue (feedback loops)
  - dequeue (translation of service rates into packet scheduling decisions)

Number of cycles (1 cycle ≈ 1 ns here)

<table>
<thead>
<tr>
<th>Guarantees</th>
<th>enqueue</th>
<th>dequeue</th>
</tr>
</thead>
<tbody>
<tr>
<td>with</td>
<td>15347</td>
<td>2603</td>
</tr>
<tr>
<td>without</td>
<td>2415</td>
<td>837</td>
</tr>
</tbody>
</table>

- A Pentium III-1GHz can process over 50,000 packets/sec.
Deployment

Static configuration of service guarantees

- Delay Class 1 < 5 ms
- Loss Rate Class 2 < 3%
- Throughput Class 1 > 10 Mbps
- Delay Class 3
- Delay Class 2 = 4
Conclusions

- Quantitative Assured Forwarding service: subsume per-class service architectures
- Low complexity/Strong guarantees
- Can be implemented at high-speeds
- Current work:
  - Avoid infeasible set of service guarantees by regulating traffic using TCP congestion control algorithms
  - Implementation at Gbps speeds (Network processor)
- Software and more information is available at: http://qosbox.cs.virginia.edu