Scalable Scheduling Support for Loss and Delay Constrained Media Streams

Richard West, Karsten Schwan & Christian Poellabauer

Georgia Institute of Technology
Introduction

- Real-Time media servers need to support 100s (even 1000s) of clients with individual RT (QoS) constraints.
- Need fast/efficient scheduling on such servers.
- We describe Dynamic Window-Constrained Scheduling (DWCS):
  - DWCS limits the number of late packets over finite windows of arrivals requiring service.
  - Focus on a scalable implementation of DWCS.
  - Approximating DWCS trades execution speed for service quality.
DWCS Packet Scheduling

- Two attributes per packet:
  - Deadline (max inter-packet gap).
  - Loss-tolerance, $x/y$.
    - $x$ late/lost packets every $y$ arrivals for service from same stream.
- At any time, all packets in the same stream:
  - Have the same current loss-tolerance.
  - Have deadlines offset by a fixed amount from predecessors.
Heterogeneous Scheduling

Higher Priority = Lower Loss-Tolerance
(Time-constrained traffic)

EDF-ordered queues

Network Pipe

Compare first packet with packet at head of 0 loss-tolerance queue to see if schedulable.

Lowest loss-tolerance first
(Non-time-constrained traffic)

Rich West (1999)
### Pairwise Packet Ordering Table

**Precedence amongst pairs of packets**

- Lowest loss-tolerance first
- Same non-zero loss-tolerance, order EDF
- Same non-zero loss-tolerance & deadlines, order lowest loss-numerator first
- Zero loss-tolerance and denominators, order EDF
- Zero loss-tolerance, order highest loss-denominator first
- All other cases: first-come-first-serve

---

Rich West (1999)
Example: $L_1 = 1/2$, $L_2 = 3/4$, $L_3 = 6/8$
$D = 1$, Service Time (C) = 1

$s_1$: $1/2(0), 1/1(1), 1/2(2), 1/1(3), 1/2(4)...


$s_3$: $6/8(0), 5/7(1), 4/6(2), 3/5(3), 3/4(4), 2/3(5), 1/2(6), 0/1(7), 6/8(8)...

Rich West (1999)
Example: \( L_1 = 1/2, \ L_2 = 1/2, \ C_1 = 5, \ C_2 = 3, \ D_1 = 5, \ D_2 = 3 \)

\[
\begin{array}{cccc}
\text{s}_1 & \text{s}_2 & \text{s}_2 & \text{s}_1 \\
0 & 1 & 2 & 3 \\
4 & 5 & 6 & 7 \\
8 & 9 & 10 & 11 \\
12 & 13 & 14 & 15 \\
16 & 17 & 18 & 19 \\
20 & 21 & 22 & 23 \\
24 & 25 & 26 & 27 \\
28 & 29 & 30 & 31 \\
32 & 33 & 34 & 35 \\
\end{array}
\]

\begin{align*}
s_1 & : 1/2(0), 1/1(5), 1/2(10), 0/1(15), 1/2(20), 0/1(25), 1/2(30)\ldots \\
s_2 & : 1/2(0), 0/1(3), 1/4(6), 1/3(9), 1/2(12), 0/1(15), 1/4(18), 1/3(21), 1/2(24), 0/1(27), 1/2(30)\ldots \\
\end{align*}

Rich West (1999)
Loss-Tolerance Adjustment (A)

For stream $i$ whose head packet is serviced before its deadline:

- if $(y_i' > x_i')$ then $y_i' = y_i' - 1$;
- if $(x_i' = y_i' = 0)$ then $x_i' = x_i$; $y_i' = y_i$;

Where:

- $x_i$ = Original loss-numerator for stream $i$
- $y_i$ = Original loss-denominator for stream $i$
- $x_i'$ = Current loss-numerator for stream $i$
- $y_i'$ = Current loss-denominator for stream $i$
Loss-Tolerance Adjustment (B)

For stream $j$ whose head packet misses its deadline:

- if $(x_j' > 0)$ then
  - $x_j' = x_j' - 1$; $y_j' = y_j' - 1$;
  - if $(x_j' = y_j' = 0)$ then $x_j' = x_j$; $y_j' = y_j$;
- else if $(x_j' = 0)$ and $(y_j > 0)$ then
  - $x_j' = 2x_j - 1$; $y_j' = 2y_j + (y_j' - 1)$; (method 1)
  - $x_j' = x_j$; $y_j' = y_j$; (method 2)
  - if $(x_j > 0)$ then $y_j' = y_j' + \lceil (y_j - x_j) / x_j \rceil$; (method 3)
  - if $(x_j = 0)$ then $y_j' = y_j' + y_j$;
DWCS Algorithm Outline

- While TRUE:
  - Find stream i with highest priority (see Table)
  - Service packet at head of stream i
  - Adjust loss-tolerance for i according to (A)
  - Deadline(i) = Deadline(i) + Inter-Pkt Gap(i)
  - For each stream j missing its deadline:
    - While deadline is missed:
      - Adjust loss-tolerance for j according to (B)
      - Drop head packet of stream j if droppable
      - Deadline(j) = Deadline(j) + Inter-Pkt Gap(j)
DWCS Implementation

(a) Deadline Heap
   \[ S_i \]
   \[ S_j \]
   \[ S_k \]

(b) Loss-Tolerance Heap
   \[ S_x \]
   \[ S_y \]
   \[ S_z \]

To Back

Back of Queue

Head Packet (Stream 1)

Head Packet (Stream n)

Select next packet for service from head packets in each stream.
Missed Deadlines (D=500, C=1)

Rich West (1999)
Loss-Tolerance Violations
(D=500, C=1)

Number of Loss-Tolerance Violations

Number of Streams

FIFO
1/80
1/90
1/100
1/110
1/120
1/130
1/140
1/150

DWCS total

Rich West (1999)
DWCS Spreads Losses

<table>
<thead>
<tr>
<th>DWCS</th>
<th>FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

Rich West (1999)
Packets Serviced Per Second

Service Rate (packets per second)

Time (seconds)

Rich West (1999)
Packets Serviced Per Second

Service Rate (packets per second)

Time (seconds)

560 streams

Rich West (1999)
Scheduling Overhead

Number of Streams

Scheduling Overhead (uS)

- **Without heaps**
- **With heaps**

Rich West (1999)
Approximation Overheads (D=200)

Scheduler Overhead (uS)

Number of Streams

Cycles Between Checking Deadlines

Rich West (1999)
Approximation Overheads (D=500)

Number of Streams

Scheduler Overhead (uS)

Cycles Between Checking Deadlines

Rich West (1999)
Deadlines Missed (D=200)

Rich West (1999)
Deadlines Missed (D=500)
Loss-Tolerance Violations (D=200)
Loss-Tolerance Violations (D=500)
DWCS Summary

- Aimed at servicing packets with delay and loss-constraints.
- Attempts to service each stream so that at most $x$ packets are lost/late for every $y$ packets requiring service.
  - DWCS minimizes the number of consecutive late packets over any finite window of packets in a given stream.
- Scheduling overhead can be reduced (and scalability increased) by using appropriate data structures (heaps, circular queues) and approximation methods.
DWCS - Current Work

- DWCS is currently being adapted for use as a CPU scheduler (using Linux), for **hard** real-time threads, so that \((y-x)\) out of \(y\) deadlines can be met.
  - Leads to bounded service delay, and guaranteed service in any finite window of service time.
- Aim is to support **coordinated** thread/packet scheduling.
Scheduling Related Work

- **Fair Scheduling**: WFQ/WF$^2$Q (Shenker, Keshav, Bennett, Zhang etc), SFQ (Goyal et al), EEVDF/Proportional Share (Stoica, Jeffay et al).

- **(m,k) Deadline Scheduling**: Distance-Based Priority (Hamdaoui & Ramanathan), Dual-Priority Scheduling (Bernat & Burns).