An Adaptive Cross-Layer Scheduler for Improved QoS Support of Multiclass Data Services on Wireless Systems

IEEE JOURNAL ON SELECTED AREA IN COMMUNICATIONS, VOL. 23, NO. 2, FEBRUARY 2005

Presented by Tsung-Yuan Hsu
Outline

- Introduction
- Commonly Used Scheduling Algorithm
- Adaptive Cross-Layer (ACL) Scheduling Algorithm
- Performance Measures
- Simulation
- Conclusion
Introduction

- Two major focuses on QoS in wireless systems
  1. Minimizing packet delay
  2. Maximizing user throughput

- Satisfying one measure sacrifices the other.

- This paper introduces a new packet scheduler which minimizes a prescribed cost function given the current channel qualities and delay states of the packets in the queue.
Commonly Used Scheduling Algorithm

a) Weighted Fair Queue (WFQ)

- The WFQ scheduling algorithm separates packets into queues according to their delay class.

- These queues are then serviced in weighted round-robin fashion.

- The weights are based on the relative performance requirements among delay classes.

- The scheduling decision does not factor in packet delays or channel qualities directly.

- The WFQ performs poorly with respect to delay and throughput measures under medium to heavy load conditions.
Commonly Used Scheduling Algorithm
b) Earliest Deadline First (EDF)

- It queues packets in ascending order according to packet deadlines, then schedules the first packet in the queue.

- The EDF algorithm does not take channel quality into account.

- When the system load is heavy and interference level is high, the EDF scheduler stalls on low throughput channels resulting in further increases in load and interference.
## Compare between WFQ and EDF

<table>
<thead>
<tr>
<th>load conditions</th>
<th>medium</th>
<th>heavy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>compare</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>better</strong></td>
<td>EDF</td>
<td>WFQ</td>
</tr>
<tr>
<td><strong>reason</strong></td>
<td>EDF algorithm schedules based on delay deadlines, it outperforms the WFQ algorithm with respect to delay under medium load conditions</td>
<td>WFQ algorithm avoids this phenomenon by circulating across all user/channel combinations in the queue</td>
</tr>
</tbody>
</table>
Adaptive Cross-Layer Scheduling Algorithm

- The name derives from the fact that the algorithm adapts the scheduling order to changes in variables across layers
  1. packet delay deadlines on the link layer
  2. channel qualities on the physical layer
Adaptive Cross-Layer Scheduling Algorithm

ACL algorithm schedules packets in the order that minimizes the cost function $J$

\[ J(v) = \sum_{i=1}^{M} \left[ (1 - \beta) \frac{\hat{d}_i}{r_i} + \beta \max \left\{ 0, \frac{\hat{d}_i}{r_i} - 1 \right\} \right]^{\gamma} \]

- $v$: a permutation of scheduling order
- $M$: the total number of packets in the queue
- $\hat{d}_i$: the delay estimate
- $r_i$: the delay requirement
- $\beta$: a weighting parameter between the estimated normalized packet delays and missed deadline penalties
- $\gamma$: determines the relative cost of incremental delays beyond the packet deadline
Adaptive Cross-Layer Scheduling Algorithm

\[ \hat{d}_i = c_i + \sum_{j=1}^{i} \frac{b_j}{e_j} \]

- \( c_i \): current delay
- \( b_j \): the packet’s remaining size
- \( e_j \): estimated channel bit-rate

\[
r(c, l) = \begin{cases} 
0.5l \quad &c = 1, l \leq 128 \\
0.5 + \frac{1.5(l-128)}{1024-128} \quad &c = 1, l > 128 \\
\frac{5l}{128} \quad &c = 2, l \leq 128 \\
5 + \frac{10(l-128)}{1024-128} \quad &c = 2, l > 128 \\
\frac{50l}{128} \quad &c = 3, l \leq 128 \\
50 + \frac{25(l-128)}{1024-128} \quad &c = 3, l > 128 
\end{cases}
\]

### Delay Requirements for Different Priority Classes and Packet Sizes

<table>
<thead>
<tr>
<th>Priority Class</th>
<th>Packet Size &amp; Delay Requirement</th>
<th>128 octets</th>
<th>1024 octets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean (s)</td>
<td>95% (s)</td>
<td>Mean (s)</td>
</tr>
<tr>
<td>1. Predictive</td>
<td>0.5</td>
<td>1.5</td>
<td>2</td>
</tr>
<tr>
<td>2. Predictive</td>
<td>5</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>3. Predictive</td>
<td>50</td>
<td>250</td>
<td>75</td>
</tr>
<tr>
<td>4. Best Effort</td>
<td>Unspecified</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Adaptive Cross-Layer Scheduling Algorithm

- ACL algorithm will reorder the packets in the queue according to the permutation that minimizes $J$ in three conditions:
  1. a new packet enters the queue
  2. an existing packet leaves the queue due to dropping or handover
  3. a queued packet’s channel quality changes
Adaptive Cross-Layer Scheduling Algorithm

- In a system with high mobility and bursty data traffic, channel qualities change at a rapid rate, determining the full scheduling order at any one time is wasteful.

- As user mobility increases, either causes a drop in scheduler performance or increases the amount of computation.

- As traffic conditions become more bursty, such situations call for a simplified version of the ACL scheduling algorithm.
Simplified ACL Scheduling Algorithm

- Instead, at any given transmission opportunity, we prefer to determine the “packet-to-send”.

- Determine the relative scheduling order of any two packets in the queue without knowing the entire order.

- Setting $\delta = 0 \Rightarrow \frac{\hat{d}_i}{r_i} \rightarrow b_j / e_j r_i$

- Instead of calculating the cost of two entire scheduling orders, we only need to calculate two simple ratios.

- Tradeoff between accuracy and processing speed.
When $\mathcal{A} = 0$, the ACL simplifies to the following steps:

1. When a new packet arrives to the queue, we place it at the end.
2. At each scheduling event, we determine the index of the "packet-to-send" in the original queue via the following pseudocode:

```bash
set \( i = 1 \)
for \( j = 2 : M \)
  if \( \frac{b_j}{c_j r_i} < \frac{b_i}{c_i r_j} \)
    then set \( i = j \).
end
```
Performance Measure

- Determine the system’s packet delay performance
  1. NPD (average normalized packet delay)
  2. PEN (missed packet deadline penalty)

- Determine user throughput
  1. EUT (average effective user throughput)

- Determine the load that a scheduler can support
  1. Percent blocking
  2. Percent dropping
## SIMULATION MODELS AND PARAMETERS

<table>
<thead>
<tr>
<th>Traffic Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>User session inter-arrival time</td>
</tr>
<tr>
<td>Files per user session</td>
</tr>
<tr>
<td>File inter-arrival time</td>
</tr>
<tr>
<td>File types</td>
</tr>
<tr>
<td>Traffic scenarios</td>
</tr>
<tr>
<td>Email file size</td>
</tr>
<tr>
<td>WWW file size</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Physical Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path-loss</td>
</tr>
<tr>
<td>Large scale fading</td>
</tr>
<tr>
<td>Small scale fading</td>
</tr>
<tr>
<td>Channel response</td>
</tr>
<tr>
<td>Mobile speed</td>
</tr>
<tr>
<td>Frequency reuse (Site/Sector)</td>
</tr>
<tr>
<td>Cell radius</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modulation/coding</td>
</tr>
<tr>
<td>Diversity</td>
</tr>
<tr>
<td>Power control</td>
</tr>
<tr>
<td>Admission control</td>
</tr>
<tr>
<td>Dropping</td>
</tr>
<tr>
<td>Mobile types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Simulation Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time step</td>
</tr>
<tr>
<td>Simulation length</td>
</tr>
</tbody>
</table>
Leaky bucket 280,
50% e-mail and 50% WWW traffic
Leaky bucket 560,
50% e-mail and 50% WWW traffic
traffic scenario 2 (75% e-mail)
traffic scenario 3 (25% e-mail)
Conclusion

Simulations confirm that the ACL scheduling algorithm greatly outperforms both the WFQ and EDF schedulers with respect to:

1. average normalized packet delay
2. missed packet deadline penalty
3. average effective user throughput
4. user blocking
5. user dropping