Configuring a Load-Balanced Switch in Hardware

Outline

- Load Balanced Switch
- Scalability
- Reconfiguration Algorithm
- Hardware Implementation
Typical Router Architecture

Load-Balanced Switch

Load-balancing mesh
Forwarding mesh
100% throughput for broad class of traffic
- No scheduler needed
- Scalable
A Single Combined Mesh

\[ \frac{N \times 2R}{N} = 2R = R + R \]

\[ (N-1) \times \frac{2R}{N} < R + R \]
Scalability

$N=8$

When $N$ is Too Large

*Decompose into groups (or racks)*
When $N$ is Too Large
Decompose into groups (or racks)

When Linecards are Missing
Failures, Incremental Additions, and Removals...

Solution: replace mesh with sum of permutations
When Linecards Fail

Questions

- Number of MEMS Switches?
- TDM Schedule?
Example – 3 Linecards

Example
2 Groups
Example
2 Groups

Group/Rack 1

<table>
<thead>
<tr>
<th>2R</th>
<th>4R</th>
<th>4R/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Group/Rack 2

<table>
<thead>
<tr>
<th>2R</th>
<th>4R/3</th>
<th>4R/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2R</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Group/Rack 1

| 2R |

Group/Rack 2

| 2R |

Number of MEMS Switches

- MEMS switches between groups $i$ and $j$

\[
\begin{bmatrix}
L_iL_j \\
N
\end{bmatrix}
\]

- Total Number of MEMS switches: $M = L + G - 1$
Questions

- Number of MEMS Switches?

- TDM Schedule?
Rules for TDM Schedule

At each time-slot:
- Each transmitting linecard sends one packet
- Each receiving linecard receives one packet
- (MEMS constraint) Each transmitting group $i$ sends at most one packet to each receiving group $j$ through each MEMS connecting them

In a schedule of $N$ time-slots:
- Each transmitting linecard sends exactly one packet to each receiving linecard

TDM Schedule

<table>
<thead>
<tr>
<th></th>
<th>$T+1$</th>
<th>$T+2$</th>
<th>$T+3$</th>
<th>$T+4$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tx Group A</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx LC A1</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Tx LC A2</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><strong>Tx Group B</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tx LC B1</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Tx LC B2</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
### TDM Schedule

<table>
<thead>
<tr>
<th></th>
<th>T+1</th>
<th>T+2</th>
<th>T+3</th>
<th>T+4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx LC A1</td>
<td>A1</td>
<td>A2</td>
<td>B1</td>
<td>B2</td>
</tr>
<tr>
<td>Tx LC A2</td>
<td>B2</td>
<td>A1</td>
<td>A2</td>
<td>B1</td>
</tr>
<tr>
<td>Tx LC B1</td>
<td>B1</td>
<td>B2</td>
<td>A1</td>
<td>A2</td>
</tr>
<tr>
<td>Tx LC B2</td>
<td>A2</td>
<td>B1</td>
<td>B2</td>
<td>A1</td>
</tr>
</tbody>
</table>

### Bad TDM Schedule

<table>
<thead>
<tr>
<th></th>
<th>T+1</th>
<th>T+2</th>
<th>T+3</th>
<th>T+4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx LC A1</td>
<td>A1</td>
<td>A2</td>
<td>B1</td>
<td>B2</td>
</tr>
<tr>
<td>Tx LC A2</td>
<td>B2</td>
<td>A1</td>
<td>A2</td>
<td>B1</td>
</tr>
<tr>
<td>Tx LC B1</td>
<td>B1</td>
<td>B2</td>
<td>A1</td>
<td>A2</td>
</tr>
<tr>
<td>Tx LC B2</td>
<td>A2</td>
<td>B1</td>
<td>B2</td>
<td>A1</td>
</tr>
</tbody>
</table>
TDM Schedule Algorithm

- The algorithm constructs three consecutive schedules.
  1. **Sending Groups to Receiving Groups**
     - Connection Assignment Problem
  2. **Sending Linecards to Receiving Groups.**
     - Matrix Decomposition Problem
  3. **Sending Linecards to Receiving Linecards**
     - Matrix Decomposition Problem

Group to Group Schedule

<table>
<thead>
<tr>
<th></th>
<th>T+1</th>
<th>T+2</th>
<th>T+3</th>
<th>T+4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx Group A</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
</tr>
<tr>
<td>Tx Group B</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
<td>AB</td>
</tr>
</tbody>
</table>
### Linecard to Group Schedule

<table>
<thead>
<tr>
<th>Tx Group A</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx LC A1</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Tx LC A2</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

### Linecard to Linecard Schedule

<table>
<thead>
<tr>
<th>Tx Group A</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx LC A1</td>
<td>A1</td>
<td>A2</td>
<td>B1</td>
<td>B2</td>
</tr>
<tr>
<td>Tx LC A2</td>
<td>B2</td>
<td>B1</td>
<td>A2</td>
<td>A1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tx Group B</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Tx LC B1</td>
<td>B1</td>
<td>B2</td>
<td>A1</td>
<td>A2</td>
</tr>
<tr>
<td>Tx LC B2</td>
<td>A2</td>
<td>A1</td>
<td>B2</td>
<td>B1</td>
</tr>
</tbody>
</table>
Connection Assignment Problem

Not Scheduled

Scheduled

Connection Assignment Problem

After Greedy

Back Tracing
Matrix Decomposition Problem

\[
\begin{pmatrix}
1 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 \\
\end{pmatrix}
= \begin{pmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
+ \begin{pmatrix}
0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
\end{pmatrix}
\]

Matrix Decomposition Problem

- Use of sparsity of matrices to represent the ones as a row-column pair
- Consists of two stages
  - Greedy Algorithm
  - Slepian-Duguid Algorithm
    1. Decomposes all the permutation matrices at once
    2. Uses the row-column pair list structure
Synthesis

- 40 Groups and 640 Linecards
- 0.13u process
- Cycle time within 4ns
- Connection Assignment Problem
  1. 10K gates
  2. 24Kbits memory
- Matrix Decomposition Problem
  1. 25K gates
  2. 230Kbits of memory

Reconfiguration Time
Thank you.