B4: Experience with a Globally Deployed Software Defined WAN TO APPEAR IN SIGCOMM'13
Treat all bits the same

30% ~ 40% average utilization

Cost of bandwidth, High-end routing gear
User data copies to remote data centers for availability/durability (lowest volume, most latency intensive, highest priority)

Remote storage access for computation over distributed data sources

Large-scale data push synchronizing state across multiple data centers (highest volume, least latency intensive, lowest priority)

Centralized Traffic Engineering (TE)

Drive links to near 100% utilization
Fast, global convergence for failures
Switch hardware
- Forwards traffic.
- Does NOT run complex control software.

OpenFlow controllers (OFC)
- Maintain network state based on network control application directive and switch events.
- Instruct switches to set forwarding entries.

Central application (logical)
- Central control of the entire network.
Figure 2: B4 architecture overview.
Applications are aggregated to **Flow Group**: {source site, dest site, QoS}

**Bandwidth function**: linear to application weight, becomes flat at required bandwidth. (discuss later)
Target: Achieve max-min fairness.

**Tunnel Selection** selects the tunnels to be considered for each FG.

**Tunnel Group Generation** allocates bandwidth to FGs using bandwidth functions to prioritize at bottleneck links.

**Tunnel Group Quantization** changes split ratios in each FG to match the granularity supported by switch hardware tables.
Find the $k$ shortest tunnels in the topology graph.

Example: Assume $k = 3$.

FG[1]: $A \rightarrow B$
- $T[1][1] = A \rightarrow B$
- $T[1][2] = A \rightarrow C \rightarrow B$
- $T[1][3] = A \rightarrow D \rightarrow C \rightarrow B$

FG[2]: $A \rightarrow C$
- $T[2][1] = A \rightarrow C$
- $T[2][2] = A \rightarrow B \rightarrow C$
- $T[2][3] = A \rightarrow D \rightarrow C$
Composition of FG level bandwidth functions

\[ \text{FG2} = \text{App3} \]
\[ \text{App2} \]
\[ \text{App1} \]
\[ \text{FG1} = \text{App1} + \text{App2} \]
Allocate bandwidth to FGs based on demand and priority.

1. Initialize each FG with their most preferred tunnels.
2. Allocate bandwidth by giving equal *fair share* to each preferred tunnel.
3. *Freeze* tunnels containing any bottlenecked link.
4. If every tunnel is frozen, or every FG is fully satisfied, end.
5. Select the most preferred non-frozen tunnel for each non-satisfied FG, goto 2.
<table>
<thead>
<tr>
<th>FG1 prefer</th>
<th>FG2 prefer</th>
<th>Fair share</th>
<th>FG1 get/need</th>
<th>FG2 get/need</th>
<th>Bottle neck links</th>
<th>Freeze tunnels</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 A→B</td>
<td>A→C</td>
<td>0.9</td>
<td>10 / 20</td>
<td>0.45 / inf</td>
<td>A→B</td>
<td>A→B, A→B→C</td>
</tr>
<tr>
<td>2 A→C</td>
<td>A→C</td>
<td>3.33</td>
<td>8.33 / 10</td>
<td>1.21 / inf</td>
<td>A→C</td>
<td>A→C→B, A→C</td>
</tr>
<tr>
<td>3 A→D→C→B</td>
<td>A→D→C</td>
<td>1.67</td>
<td>1.67 / 1.67</td>
<td>3.34 / inf</td>
<td>all</td>
<td>all</td>
</tr>
</tbody>
</table>

Result:

FG1 (20/20):
- A→B: 10
- A→C→B: 8.33
- A→D→C→B: 1.67

FG2 (5/inf):
- A→C: 1.67
- A→B→C: 0
- A→D→C: 3.34
Determining the optimal split: integer programming problem.

Greedy Approach:

1. Down quantize (round) each split.
2. Add a remaining quanta to a non-frozen tunnel that makes the solution max-min fair (with minimum fair share).
3. If there are still remaining quantas, goto 2.
Example split:
- FG2: 0.3:0.0:0.7
- FG1: 0.5:0.4:0.1

Assume quanta is 0.5.

FG2 (A→C):
- A→C, A→B→C, A→D→C
- Down quantize: 0.0:0.0:0.5
- Add remaining: 0.0:0.0:1.0

FG1 (A→B):
- A→B, A→C→B, A→D→C→B
- Down quantize: 0.5:0.0:0.0
- Add remaining: 0.5:0.5:0.0
Figure 13: TE global throughput improvement relative to shortest-path routing.
Use **prioritized** switch forwarding table entries

“**Big red button**”: disable TE service and fall back to shortest-path forwarding at any time
Figure 8: Multipath WAN Forwarding

g Example.
Figure 9: Layering traffic engineering on top of shortest path forwarding in an encap switch.
In order to avoid packet drops, not all ops can be issued simultaneously.

Rules:
- Configure a *tunnel* at all affected sites before sending TG and FG.
- A *tunnel* cannot be deleted until all referencing entries are removed.

Enforce dependencies (in case of network delays / reordering):
- OFC maintains the highest session sequence number.
- OFC rejects ops with smaller sequence number.
- TE Server retries any rejected ops after a timeout.
Statistics
- 13 topology changes per minute
- Trimming maintenance updates: 0.2 changes per minute
- Edge add/delete events 7 changes per day (TE algorithm runs on aggregated topology view)

Takeaways:
- Topology aggregation significantly reduces path churn and system load.
- Even with topology aggregation, edge removals happen multiple times a day.
- WAN links are susceptible to frequent port flaps and benefit from dynamic centralized management.
Impact of Failures

Centralized TE is not a cure-all.

<table>
<thead>
<tr>
<th>Failure Type</th>
<th>Packet Loss (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single link</td>
<td>4</td>
</tr>
<tr>
<td>Encap switch</td>
<td>10</td>
</tr>
<tr>
<td>Transit switch neighboring an encap switch</td>
<td>3300</td>
</tr>
<tr>
<td>OFC</td>
<td>0</td>
</tr>
<tr>
<td>TE Server</td>
<td>0</td>
</tr>
<tr>
<td>TE Disable/Enable</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5: Traffic loss time on failures.
Link Utilization

% Utilization

% Link Utilization
Overheads in hardware programming.
- Each multipath table operation is typically slow (~100ms), forming the principal bottleneck in reliability.

Scalability and latency of the packet I/O path between OFC and OFA.
- OpenFlow might support two communication channels to separate high-priority operations from throughput-oriented operations.
Q&A