Netslice Bottleneck

David Kelly, dmk257
Ross Hanson, reh237
Context

- High performance packet processing libraries allow for line rate speeds on commodity hardware
- Netslice vs. netmap vs. Intel DPDK
- Netslice seems to hit a bottleneck at ~30Gbps
Motivation

● From an educational standpoint: to learn about packet processing frameworks working at bleeding edge speeds
● From an engineering standpoint: to improve upon Netslice and match its portability with performance
Analysis

- "Static analysis"
- Fuzz testing
- OProfile
Optimizations

- Compile time optimizations
- `schedule()` vs. `schedule_timeout()`
- Batching expensive updates
Results

- Localized where the bottleneck is coming from
- Small successes in mitigating it

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Line</td>
<td>9195.2 Gbps</td>
</tr>
<tr>
<td>2 Lines</td>
<td>17922.4 Gbps</td>
</tr>
<tr>
<td>3 Lines</td>
<td>27926.2 Gbps</td>
</tr>
<tr>
<td>4 Lines</td>
<td>32204.2 Gbps</td>
</tr>
</tbody>
</table>
Future Work

● Create a final fix to the bottleneck
● Test in the wild with a production application
Lessons Learned

- Do not assume “mature” software can be implemented with minimal issues
- According to the internet, everyone innately knows how to profile kernel modules but no one has ever created a tutorial about it
Conclusion

- The current bottleneck does not seem to be the lack of zero-copy
- Given this, it seems plausible that Netslice could eventually reach the performance of netmap with better portability