PEGASE: A robust and efficient tool for worst case network traversal time evaluation on AFDX

Marc Boyer, ONERA

Jörn Migge, RealTime-at-Work

Marc Fumey, Thales Avionics

AFDX

- Avionics Systems: communicating real-time systems
- AFDX: Avionics Full Duplex ethernet
  - New avionics backbone
  - Ethernet-based
  - Full Duplex => no collision
- Shared network
  - Indeterminism at the switch level
  - Need for guaranteed bounds (e.g. frame Worst-Case Traversal Times and buffers size)
Network Calculus

- Bound computation method: Network Calculus
- Formal Framework
  - Strong background: (min,+) algebra
  - Very general and flexible model

Network Calculus Flexibility

- Modeling (periodic+jitter flow)
  - Simple constraint: Token bucket
  - Tight constraint: Stair Case
Network Calculus and AFDX

- Network calculus used to certify A380 AFDX
- Network calculus bounds never reached
- Challenge: reduce over-approximation => reduce over provisioning

The PEGASE Tool

- Requirements:
  - Accurate results (up to date wrt Network Calculus theory)
  - Extendable (to support exploratory works)
  - Trustable
  - Domain-specific editor (creating networks without being network calculus specialist)
  - Containing computation time
- Conflicting requirements
  => Modular conception
PEGASE Modular Architecture

- Decomposed into components
- Some components have several implementations (tradeoff complexity / accuracy / simplicity)
- Different users – different components

Modular Conception example

- Floating point vs Rational Numbers
  - Floating point (2.0, 0.666): Fast, but rounding errors
  - Rational numbers (2, 2/3): Exact, but slow
- Function classes
  - ICC: Increasing Convex and Concave (Piecewise Linear)
    - 1292 LOC / Rational and floating point Version
    - Coarse modeling: token-bucket constraint
  - UPP: Very general class of Piecewise linear function
    - 3416 LOC / Rational only
    - Tight modeling: sporadic messages
Different modules / different complexities

<table>
<thead>
<tr>
<th>Module</th>
<th>#Lines of code</th>
<th>Complexity (Cyclomatic)</th>
<th>#Methods</th>
<th>Cplx / #Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fractions</td>
<td>862</td>
<td>268</td>
<td>73</td>
<td>3.67</td>
</tr>
<tr>
<td>Double</td>
<td>84</td>
<td>32</td>
<td>24</td>
<td>1.33</td>
</tr>
<tr>
<td>ICC</td>
<td>1292</td>
<td>318</td>
<td>74</td>
<td>4.3</td>
</tr>
<tr>
<td>UPP</td>
<td>3416</td>
<td>719</td>
<td>106</td>
<td>6.8</td>
</tr>
</tbody>
</table>

The network editor

The gray boxes are the switches while the end-systems are the white boxes. The names of the virtual links are shown as labels of the physical links.
The results panel

Red means that the time constraint cannot be guaranteed for a given virtual link.

Illustration on realistic AFDX system

- 104 End-Systems
- 8 Routers
- 4 Priority levels
- 974 Data flows (Virtual links)
- 6501 Latency constraints
- Periods (min: 2ms / max: 128 ms / av: 60 ms)
- Path Lengths (min: 1 / max: 3 / av: 1.3)
- Constraints (min: 1ms / max: 30 ms / av: 10ms)
Computation times for different trade-offs accuracy /computing times

<table>
<thead>
<tr>
<th>Configuration ID</th>
<th>Constraint Model</th>
<th>Number Type</th>
<th>Function Class</th>
<th>Computation duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Token Bucket</td>
<td>Float</td>
<td>ICC</td>
<td>2 s</td>
</tr>
<tr>
<td>#2</td>
<td>Token Bucket</td>
<td>Rational number</td>
<td>ICC</td>
<td>11 s</td>
</tr>
<tr>
<td>#3</td>
<td>Token Bucket</td>
<td>Rational number</td>
<td>UPP</td>
<td>19 s</td>
</tr>
<tr>
<td>#4</td>
<td>Stair-case</td>
<td>Rational number</td>
<td>UPP</td>
<td>33 mn</td>
</tr>
</tbody>
</table>

WCTT Bounds Results

Warning: actual worst case traversal times (WCTT) is unknown

- From [Bauer 2010]:
  - average (WCTT – token bucket) < 13%
- Average gain Stair Case vs Token Bucket: 6%
WCTT Bounds Results for token bucket and stair-case models of the input traffic

Gain with stair-case is larger for low-priority Virtual links
Synthetic results

- By priority
  - High priority: no gain (0.38%)
  - Low priority: significant gains (12.5%)

- By path length (number of hops)
  - Short path: 5.7%
  - Long path (length 3): 7.3%

Conclusion

- Network calculus is a theory that is:
  - Exciting (for academics)
  - Trustable (strong formal background)
  - Flexible

- with an industrial tool: PEGASE
  - Conceived for network designers with a domain specific editor
  - Customizable performances: accuracy vs computation time
  - Enable to reduce HW resources over-provisioning
  - Increase possibility of system evolution and system re-use
Thank you for your attention

http://sites.onera.fr/pegase