PER-FLOW DELAY BOUND ANALYSIS BASED ON A FORMALIZED MICROARCHITECTURAL MODEL

XUEQIAN ZHAO AND ZHONGHAI LU

KTH ROYAL INSTITUTE OF TECHNOLOGY, STOCKHOLM

NOCS’2013, Tempe USA. April 23, 2013
Outline

- The QoS problem
- The QoS analysis methodology
- An illustrative example
- Summary
The QoS problem

- Per-flow delay bound?
  - Flow is a unicast packet stream from one source to one destination
  - Worst-case communication time (WCCT) analysis

- Complexity
  - Distributed interferences over distributed resources (routers, links)
  - Dynamic interference
This work

- Develop a QoS analysis methodology
  - Network calculus
    - A queuing theory tackling worst-case communication performance
    - Foundation for QoS provision in Internet, ATM etc.
  - xMAS model based
    - xMAS (executable Micro-Architectural Specification) is a formal modeling framework for communication fabrics.

- Goals for the QoS analysis methodology
  - Be general
    - Start from a typical VC router
  - Be automatable
    - Make steps well-defined
QoS analysis methodology

**Input**
- Network info
  - Topological graph
  - Router microarchitecture with service policy
- Flow info
  - Deterministic path
  - Flow parameters

**QoS Analysis (Methodology)**
- Validation (correctness, tightness)

**Output**
- Per-flow delay bound

**Steps**
- **S1**: Build an xMAS model using network-flow info
- **S2**: Map the xMAS model to its NC analysis model
- **S3**: Derive closed-form formulas based on the NC model
Network Calculus in a nutshell

- Two key abstract concepts
  - Arrival curve: bound max. cumulative traffic arrival
  - Service curve: bound min. cumulative service

- Basic closed-form results
  - Assumptions:
    - Linear arrival curve \( \alpha(t) = \sigma + \rho(t) \)
    - Latency-rate server \( \beta(t) = R(t - T)^+ \)
  - Bounds
    - Latency bound \( \overline{D} = T + \frac{\sigma}{R} \)
    - Backlog bound \( \overline{B} = \sigma + \rho T \)
Primitives

- source
- sink
- function
- queue
- merge
- switch
- fork
- join

Formal equation example

(a) The fork primitive (left) and its signal equivalent (right)

\[
\begin{align*}
    i & \rightarrow \quad a \quad i.irdy \quad i.data \quad a.irdy \quad a.data \quad f(i.data) \\
    b & \rightarrow \quad b.irdy \quad b.data \quad b.irdy \quad b.trdy
\end{align*}
\]

(b) The join primitive (left) and its signal equivalent (right)

\[
\begin{align*}
    a & \rightarrow \quad a.irdy \quad a.data \quad b.irdy \quad b.data \quad g(i.data) \\
    b & \rightarrow \quad o.irdy \quad o.data \quad o.irdy \quad o.trdy
\end{align*}
\]

\[
\begin{align*}
    a.irdy & := i.irdy \text{ and } b.trdy \\
    b.irdy & := i.irdy \text{ and } a.trdy \\
    i.trdy & := a.trdy \text{ and } b.trdy
\end{align*}
\]

\[
\begin{align*}
    a.trdy & := o.trdy \text{ and } b.irdy \\
    b.trdy & := o.trdy \text{ and } a.irdy \\
    o.irdy & := a.irdy \text{ and } b.irdy \\
    o.data & := h(a.data, b.data)
\end{align*}
\]
xMAS Macros

- A credit counter: Initially $k$ tokens; One token in produces one token for out

- It comprises five components
  - Source and fork issue tokens as the credits
  - Queue counts up to $k$ credits
  - Join and sink ensure the 1-to-1 correspondence between the input and the output
An input-queuing VC router

- **Data path**
  - Queuing unit
  - Crossbar

- **Control path**
  - Routing unit: determine output port
  - VC allocator: determine which VC to use downstream
  - Crossbar arbiter: ordering packets to share the output channels (links)
  - Credit management
    - Realize credit-based flow control
    - Avoid buffer overrun

- In such diagram, data path and control path are isolated.
Model a VC router in xMAS

- Data path is a direct mapping
  - VC to queue
  - DMX to switch
  - MUX to merge

- Control path
  - Credit-based flow control now is explicitly modeled
  - It is integrated in the data path via join and fork
    - Join: move one packet consumes one credit downstream
    - Fork: generate one credit upstream
Model a VC router in xMAS

Different ways of modeling flow control with data flow reflect different design decisions.

- After the crossbar:
  - Arbitration before credit check
  - 1 join per PC (2 joins)

- Before the crossbar:
  - Arbitration after credit check
  - 1 join per VC (4 joins)
Model a VC router in xMAS

What about other control units?

- Crossbar allocator (link arbiters): implicit in the merge policy
- Routing algorithm
  - Imply which output link a flow uses
- VC Allocator
  - Imply which downstream VC a flow will use

Assumptions for determinism:
one flow has exactly one path inside and outside routers

- Deterministic routing
- Static VC allocation
Customize the **generic** xMAS model to get a **specific** xMAS model

- Exploiting the deterministic assumptions
- Keep only relevant parts of the router (no more a uniform router, but like a customized specific router)
Map an xMAS model to an NC model

- An NC analysis model is a graphical composition of NC elements.
  - Source, server, buffer

- Two levels of mapping
  - At the primitive level: Map xMAS primitives to NC elements
    - Source, sink, function, queue, merge, join, switch, fork
  - At the macro level: Map xMAS macros to NC elements
    - Credit counter

- Map connections
<table>
<thead>
<tr>
<th>xMAS primitives</th>
<th>NC elements</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Token Bucket Shaper (TBS)</td>
<td>A TBS may shape any kind of traffic to conform a linear arrival curve.</td>
</tr>
<tr>
<td>Sink</td>
<td>Latency-rate server (LRS)</td>
<td>LRS is characterized by latency-rate service curve</td>
</tr>
<tr>
<td>Function</td>
<td>Analysis cut point</td>
<td>It typically results in data transformation. A cut point for sink and source.</td>
</tr>
<tr>
<td>Queue</td>
<td>Buffer or delimiter</td>
<td>Direct, explicit mapping for packet queue</td>
</tr>
<tr>
<td>Merge</td>
<td>Arbitration server</td>
<td>Implement an arbitration function</td>
</tr>
<tr>
<td>Join</td>
<td>Flow controller server</td>
<td>Implement a flow control function</td>
</tr>
<tr>
<td>Switch</td>
<td>Connector</td>
<td>Provide connectivity and split the flow</td>
</tr>
<tr>
<td>Fork</td>
<td>Connector</td>
<td>Provide connectivity and duplicate the flow</td>
</tr>
</tbody>
</table>
An illustrative example

- To illustrate the procedure
  - From specific router to specific xMAS model
    - Based on the generic router model
    - Use of the flow information
  - From xMAS model to NC model
  - NC analysis
  - Results

- A simple example still considers
  - Link sharing
  - Buffer sharing
  - Control sharing
An illustrative example

- From a specific router model to a specific xMAS model
An illustrative example

- Map the specific xMAS model to its NC model
An illustrative example

- From the NC model to per-flow delay bound

\[ f_0 : \alpha_0 \]
\[ f_1 : \alpha_1 \]

- Analysis results: closed-form delay bound formula for \( f_0 \)

\[ \bar{D} = \begin{cases} 
T_{sys} + \frac{b_0}{C_{sys}}, & B \geq C_{sys} T_{sys} \\
T_{sys} + \frac{b_0}{C_{sys}} + (T' - \frac{B}{C_{sys}}) \lfloor \frac{b_0}{B} \rfloor, & B < C_{sys} T_{sys}
\end{cases} \]

where
\[ T_{sys} = T_1 + T_s + \phi_1 + (b_1 + r_1(\phi_0 + T_1))/C_s, \]
\[ C_{sys} = w_0 C_1 \land (C_s - r_1), \]
\[ w_0 = \phi_0/ (\phi_0 + \phi_1), \]
\[ T' = T_s + \nu. \]

Linear arrival curves and latency-rate servers

- \( T_1 \): latency of the arbitration;
- \( T_s, C_s \): latency and service rate of the sink;
- \( \phi_0, \phi_1 \): service weights of \( f_0 \) and \( f_1 \), respectively;
- \( b_0, r_0 \): burstiness and arrival rate of flow \( f_0 \);
- \( b_1, r_1 \): burstiness and arrival rate of flow \( f_1 \);
- \( B \): ingress buffer size of flow \( \{f_0,f_1\} \);
- \( \nu \): credit feedback delay.
An illustrative example

- Validate analytical results against RTL simulation of the xMAS model

- Simulation results
  - One case without flow control effect: $B \geq C_S(T_s + \nu)$, tightness is 94.7%
  - One case with flow control effect: $B < C_S(T_s + \nu)$, tightness is 98.8%
Analysis validation (no backpressure)

- **Arrival curve of flow** $f_0$
  \[ \alpha_0 = 0.2t + 3 \]
  \[ b_0 = 3, \quad r_0 = 0.2 \]

- **Arrival curve of flow** $f_1$
  \[ \alpha_1 = 0.2t + 3 \]
  \[ b_1 = 3, \quad r_1 = 0.2 \]

- **Service curve of the arbiter** $\sigma$
  \[ \beta^\sigma = 1(t - 0)^+ \quad \phi_0 = 1 \]
  \[ C = 1, \quad T = 0 \quad \phi_1 = 1 \]

- **Service curve of the sink** $\sigma_s$
  \[ \beta^\sigma_s = 0.9(t - 100)^+ \]
  \[ C_s = 0.9, \quad T_s = 100 \]

- **Credit feedback delay**
  \[ \nu = 2 \]

- **Size of the ingress buffer**
  \[ B \geq 92 \]

\[ \bar{D} = T_{sys} + \frac{b_0}{C_{sys}} = 114 \]

\[ T \text{ tightness: } \frac{108}{114} \times 100\% = 94.7\% \]
Analysis validation (with backpressure)

- **Arrival curve of flow** $f_0$
  \[ \alpha_0 = 0.3t + 5 \]
  \[ b_0 = 5, \ r_0 = 0.3 \]
- **Arrival curve of flow** $f_1$
  \[ \alpha_1 = 0.4t + 2 \]
  \[ b_1 = 2, \ r_1 = 0.4 \]
- **Service curve of the arbiter** $\sigma$
  \[ \beta^\sigma = 1(t - 0)^+ \]
  \[ \phi_0 = 1 \]
  \[ C = 1, \ T = 0 \]
  \[ \phi_1 = 1 \]
- **Service curve of the sink** $\sigma_s$
  \[ \beta^\sigma_s = 0.9(t - 500)^+ \]
  \[ C_s = 0.9, \ T_s = 500 \]
- **Credit feedback delay**
  \[ \nu = 2 \]
- **Size of the ingress buffer**
  \[ B = 6 \]

\[ \bar{D} = T_{sys} + \frac{b_0}{C_{sys}} + (T' - \frac{B}{C_{sys}})[\frac{b_0}{B}] = 517 \]

\[ \xi = \frac{511}{517} \times 100\% = 98.8\% \]
Summary

- Define an xMAS-based QoS analysis procedure
- Show how to build the xMAS model of a VC router
- Show a mapping scheme from xMAS model to NC model
- Give a tutorial example showing good tightness

Future work
- Automate the analysis
- Systematic study on tightness
Acknowledgement

Thank you for attention!

Sponsor:

[Intel logo]