Network Calculus: A worst-case theory for QoS guarantees in packet networks

Giovanni Stea
Computer Networking Group
Department of Information Engineering
University of Pisa, Italy

IMT Lucca, Oct 28, 2009

Outline

- Motivation
  - Performance analysis of real-time traffic
  - Why classical queueing theory is unfit
- Network Calculus
  - Basic modeling: arrival and service curves
  - Concatenation, bounds
  - “Pay burst only once” principle / IntServ
  - Advanced modeling: aggregate scheduling
  - Stochastic Network Calculus
Real-time traffic

- Expected to represent the bulk of the traffic in the Internet soon
  - Skype users: $10^7$-$10^8$
  - Cisco white paper: video traffic volume to surpass P2P in 2010
- Revenue-generating only if reliable
- Reliability boils down to “packets meeting deadlines”
  - End-to-end delay bounds are required

Performance analysis

- Tagged flow (of packets) traversing a path
- Cross traffic
- Many queueing points (routers)
- How to compute a bound on the e2e delay?
### Performance analysis (2)

- Service Level Agreement with upstream neighbor
- I will carry
  - up to X Mbps of your traffic
  - from A to B
  - within up to Y ms (!!)
  - for Z$

### Network Calculus and Queueing Theory

- 100 years of Queueing Theory
  - Originated in the area of telecommunications
  - Developed and applied in a variety of areas
  - Erlang Centennial held in Denmark, April 2009.

- ~20 years of Network Calculus
  - Recent development of queueing theory for computer networks
NC and Queueing Theory (2)

- Queueing theory requires **models** for traffic
  - Simplistic models required for tractability
  - What if the traffic mix changes?
    - New applications (social networks, etc.)
    - Flash crowds (e.g., a football match)
    - Topology modifications (routing, link upgrades)
- Queueing theory mainly concerned with **average** performance metrics
  - Real-time traffic needs **bounds**

---

Modeling a network with NC (1)

Fixed delays: propagation, …

Variable delay: queueing

Per-flow scheduling

Aggregate scheduling
Modeling a queueing point with NC

A scheduler serves its queues on a time sharing basis:

- Most guarantee that a queue is visited:
  - For a minimum amount of time
  - After a computable maximum vacation

Modeling a queueing point with NC (2)

- Minimum service over a maximum interval
  - A minimum guaranteed rate
  - With a latency (when the server is away)
  - The latency is upper bounded

- Round robin schedulers (DRR, PDRR, …)
- Fair Queueing schedulers (PGPS, WFQ, WF2Q, STFQ, SCFQ, …)
- Strict priority (for the queue at highest prio)
**Example: a Round Robin scheduler**

- Fixed length packets, 2 packets per queue

**Example: a priority scheduler**

- Strict non preemptive priority, queue scheduled at top priority

**Service curve**: summarizes the service received *in a worst case* by a backlogged tagged flow

Models the presence of other queues

**Rate-latency service curves** most common in practice
Modeling a queueing point with NC (3)

- Worst-case behavior for my queue:
  - served at **minimum** rate
  - with **maximum** latency

Modeling a queueing point with NC (4)

- **Nodes** transform functions of time

\[
D(t) \geq A \otimes \beta(t)
\]

\[
A \otimes \beta(t) = \inf_{0 \leq s \leq t} \{A(s) + \beta(t - s)\}
\]

\[
\int_0^t A(s) \cdot \beta(t - s) \, ds
\]
**Concatenation property**

- Assume a flow traverses a tandem of nodes

\[
A(t) \quad \beta_{1,2}(t) = \beta_1 \otimes \beta_2(t)
\]

\[
D(t) = D(t) \geq A \otimes \beta_{1,2}(t)
\]

It works for any # of nodes

---

**Concatenation property (2)**

- At each of the \(N\) nodes:
  - Cross traffic could be anything
  - **Schedulers** need not be the same

- mid ’90s: analyzing a tandem of **identical** schedulers
  - Took a whole PhD
  - Was enough for a top-level journal paper (Parekh ’93, Saha ’98,…, all on IEEE/ACM TNet)
An end-to-end delay bound (1)

- A bound on the delay for a given arrival process can easily be computed

![Diagram showing an end-to-end delay bound (1)]

An end-to-end delay bound (2)

- Traffic shaping/policing
  - Check conformance with a pre-specified profile
  - Drop/delay out-of-profile traffic
  - Employed to verify SLA conformance

![Diagram showing an end-to-end delay bound (2)]
An end-to-end delay bound (3)

- Arrival curve: a **constraint** on the arrival process

\[
A(t) - A(s) \leq \alpha(t - s)
\]

\[
A(t) \leq A \otimes \alpha(t)
\]

example:

- a token bucket
- burst B, rate r

Backlog and delay bounds in NC

- **service curve** for a path + **arrival curve** for a flow

D is a **bound on the e2e delay**

B is a **bound on the backlog**

Bounds are **tight**
**Convolution of rate-latency curves**

\[
\beta_i(t) = R_i \cdot (t - L_i)^+ \quad \beta_{1,2}(t) = \beta_1 \otimes \beta_2(t) = \inf_{0 \leq s \leq t} \{\beta_1(s) + \beta_2(t-s)\}
\]

- **Sum** of the latencies, **min** of the rates

**“Pay bursts only once” principle**

- When traversing a tandem, your DB consists of
  - **N** latencies
  - One **burst delay**

\[
D = \sum_{i=1}^{N} L_i + \frac{B}{\min_{1 \leq i \leq N} \{R_i\}} < \sum_{i=1}^{N} \left[ L_i + \frac{B}{R_i} \right]
\]

Pay burst only once
(at the min rate)

\[
D(1 \ldots N) < D(1) + \ldots + D(N)
\]
IntServ architecture (1)

- RSVP protocol
- FlowSpec: burst and rate of the flow
  - i.e., its token bucket arrival curve
- Path message
  - Includes FlowSpec and required delay bound
  - Collects node latencies at each hop

\[
\text{PATH } L_1 \quad \text{PATH } L_1 + L_2 \quad \text{PATH } L_1 + \ldots + L_N
\]

IntServ architecture (2)

- At the destination
  - compute what rate you need to reserve
  - Based on your required delay \( D \)

\[
R = \max \left\{ \frac{B}{D - \sum_{i=1}^{N} L_i}, r \right\}
\]
### IntServ architecture (3)

- RESV message travels back and **reserves** a rate $R$ at each node
  - The delay bound is guaranteed from now on

![IntServ architecture](image)

### Traffic aggregation

- Aggregation as "the" solution for **scalable** provisioning of QoS in core networks
- Internet:
  - Differentiated Services
  - MPLS

  Per-flow resource management (e.g., packet scheduling) just doesn’t scale
Per-aggregate scheduling

- Packets of an aggregate normally queued FIFO
- Arbitration (scheduling) among aggregates
  - Forwarding guarantees for the aggregate

NC with aggregate scheduling

- Computations get more involved…

FIFO queues
Traffic aggregation (cont.)

- Aggregates change along a path
  - Per-node guarantees for different sets of flows at each node
- Cannot use concatenation of SCs

Performance evaluation problem

- Users care about their flows, not aggregates
- Users want e2e delay bounds, not per-node forwarding guarantees

How to compute per-flow end-to-end delay bounds from per-aggregate per-node guarantees?
**Performance Analysis**

- **Tandem network** of FIFO rate-latency elements
- All nodes have a rate-latency SC for the aggregate
- All flows have a leaky-bucket AC

**Leftover service curves**

- Th. FIFO Mux - Minimum Service Curves
- [Cruz et al., ‘98]

$$\beta'(t, \tau) = \left[ \beta(t) - \alpha_z(t - \tau) \right]^+ \cdot 1_{\{\tau \geq 0\}}, \tau \geq 0$$
The LUDB methodology

LUDB: Least Upper Delay Bound

Step 1:
Apply the FIFO Mux theorem iteratively so as to “remove” all cross-flows


The LUDB methodology (2)
The LUDB methodology (3)

- **An** **n-dimensional infinity** of e2e SCs for the tagged flow
  - \( n = \# \) of cross-flows
  - Delay bound = fn. of \( n \) parameters

- **Step 2**
  - Solve an **optimization problem**
    \[
    LUDB = \min_{\tau_i \geq 0} \{ D(\tau_1, \ldots, \tau_n) \}
    \]
  - The minimum is the **best**, i.e. **tightest**, delay bound

Nested vs. non-nested tandems

- **Nested iff**
  - \( \text{path}(f_1) \cap \text{path}(f_2) = \emptyset \) (disjoint)
  - \( \text{path}(f_1) \subseteq \text{path}(f_2) \) (nested)

- You can **only** compute an e2e SC for the tagged flow in a **nested tandem**
Two important points

- The LUDB method:
  1. Is **scalable** enough
     - You can use it in paths of 30+ nodes
  2. Yields **accurate** bounds
     - Close to a flow’s **Worst-Case Delay**
     - (Sometimes)


Scalability

- Computation times for **very nasty** non-nested tandems

**Accuracy**

- Delay bounds are as useful as they are tight, i.e. close to the Worst-Case Delay
  - WCD unknown (to date)
  - End-to-end analysis is fundamental

**Sink-tree networks**

- Closed-form delay bound
  \[
  D = \sum_{i=1}^{N} \left[ L_i + \frac{B_i}{CR_i} \right]
  \]
  - = Worst-Case Delay
  - Proof by construction

Accuracy (2)

- Compared to (naïve) per-node analysis

\[
\begin{array}{c}
\text{per-node delay/LUDP ratio} \\
N \\
\end{array}
\]

500-1000 times smaller

Stochastic Network Calculus

- Brings in a probabilistic framework
- Better captures statistical multiplexing
- Concatenation results still hold

\[
D(t) \geq A \otimes \beta(t) \quad P\{A \otimes \beta(t) - D(t) > x\} \leq g(x)
\]

- Currently active field of research
- SIGMETRICS, VALUETOOLS
Conclusions

- Network Calculus allows one to compute e2e delay bounds
  - **Easy** and **tight** in a per-flow scheduling environment
  - **Complex** and **not always tight** in an aggregate-scheduling environment
- Only method known so far
- Stochastic extensions: promising research area
  - Better account for statistical multiplexing

References

More references


Thank you for listening

• Questions?
• Comments?