Typical versus Worst Case Design in Networking

Nandita Dukkipati
Yashar Ganjali, Rui Zhang-Shen

High Performance Networking Group
Stanford University

HotNets-IV, November 2005
Introduction: Worst-case design

- Preference for worst-case design: packet classification algorithms, switch buffer architectures, congestion control, ...

- Lampson’s hint: “... Normal case must be fast. The worst-case must make some progress.”

- Why the preference for worst-case design?
  - Typical case known only after system is deployed
  - Easier to quantify/verify/prove performance
  - Feels good and safe
Introduction: The case for typical-case design

- **Worst-case design** does not always make sense

- **Our position:** Design for the typical-case unless designing for the worst-case is absolutely necessary or cheap

- **Examples:**
  - Buffer sizing in core routers
  - Designing a congestion control algorithm
  - Backbone network design
Typical-case Design: Buffer Sizing for Core Routers

- How much **buffering** do Internet routers need to guarantee near 100% link utilization?
- **Rule of thumb:** $C \times \frac{RTT}{C \times RTT}$
- **Example:** $RTT = 250$ ms, $C = 40$ Gbps, buffer size = 1,250,000 packets
- **Rule of thumb holds for single/synchronized flow(s)**
**Typical-case Design: Small Buffers**

- **Desynchronized flows:** expect less buffering

- **Buffering needed:** \( \frac{C \times RTT}{\sqrt{N}} \)

- **Example:** \( RTT = 250 \text{ ms}, \ C = 40 \text{ Gbps}, \ N = 10000, \) buffer size = 12,500 packets

- **Worst-case:** few flows, **Typical-case:** thousands of flows

- **Benefits:** reduced delay and jitter, simplified router architecture
Typical-case Design: Very Small Buffers

- **Worst-case assumption**: core link should be 100% utilized
- **Recent work**: Buffer size *10-20 packets*, link utilization *75%*
- **Five orders magnitude** reduction from original rule-of-thumb
- **Consequences**: all-optical routers, very high capacity network, low power consumption, low delay...
Typical-case Design: Designing Congestion Control

- Congestion control is deliberately designed to be conservative
  - Starts flows slowly: helps in flash crowd scenarios
  - Works well when most or all flows are long-lived

- Typical case:
  - mean flow size $\geq$ “pipe” size
  - mean flow size $<<$ “pipe” size
Flows arrive in a Poisson process and have a heavy-tailed flow-size distribution. Consequently, flows last many times longer than necessary in the typical case.
Typical-case Design: Rate Control Protocol (RCP)

- Rate Control Protocol (RCP): Designed for fast Flow Completion times --- close to ideal processor sharing

\[ R(t) = R(t - T)[1 + \frac{T}{d_0} \left( \alpha(C - y(t)) - \beta \frac{q(t)}{d_0} \right) \frac{1}{C}] \]

Typical case: One/two orders of magnitude reduction in Flow Completion Time

Worst case: Flash crowds
When Worst-case Design makes sense

- **Worst-case guarantees** are required

- **Cheap** to design for worst case and it doesn’t overly hurt the typical case

- **Don’t know the typical-case** or it is likely to change faster than you expect to change your design
Worst-case Design: Backbone Network Design

- What makes network design hard?
  - Inaccurate design input
  - No guarantees on handling deviant matrices
  - Need to provision for failures

- Example:

\[ T = \{ \Lambda | \sum_j \lambda_{ij} \leq R_i, \forall i; \sum_j \lambda_{ji} \leq R_i, \forall i \} \]

<table>
<thead>
<tr>
<th>% over-provision</th>
<th>% traffic-matrices</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td>0.20%</td>
</tr>
<tr>
<td>25%</td>
<td>2.59%</td>
</tr>
<tr>
<td>50%</td>
<td>15.09%</td>
</tr>
</tbody>
</table>
Worst-case Design: Valiant Load Balancing

\[ r, \quad 2r/N, \quad r, \quad r, \quad r \]
Worst-case Design: VLB Characteristics

- **Worst-case guarantees**
  - Can support all feasible traffic-matrices
  - Is provably the most efficient in supporting all traffic
  - Empirical study: About the same cost as conventional network

- **Typical-case:**
  - Max. propagation delay bounded by 2x network diameter
  - Only load-balance when necessary
  - There are “express paths”
Conclusion

• Blindly designing for worst-case does not make sense
• Immense benefits in typical-case design in terms of performance, cost and complexity
• Design for typical-case unless typical-case is not known or the worst-case design is absolutely necessary or is cheap.