Non-intrusive scheduling of flows in networks

Urtzi Ayesta
LAAS-CNRS & Ikerbasque

Based on joint work with: S. Aalto, D. Carvin, L. Bertaux

Eindhoven, 02/06/2015
Traffic flows (web pages, email, music and movies…)

For a given performance criterion, what is the optimal way to schedule flows?
Two “type” of networks

- Internet: packets, congestion control, TCP etc.

- Bandwidth-sharing network
Outline

- Scheduling in networks
- Non-intrusive scheduling
  "Local optimality" of size-based
- gTCP: Non-intrusive TCP
  - Decoupling congestion control and scheduling
- Experimental results
Broad literature:
- optimality of size-based scheduling:
  SRPT among size-aware policies
  LAS among size-unaware policies
- Exact performance analysis for many disciplines
Flow 0 starts at t=0, size 2, flow 1 at t=0, size 2, flow 2 starts at t=2, size 1.
Stability is scheduling dependent

- **Class** $i$ is stable iff $P(N_i=0) > 0$
- **Network** is stable if all classes are stable

- Let us consider a linear network

  ![Diagram of a linear network]

  **Necessary and sufficient condition for stability of network:**

  $$\rho_0 + \rho_i < 1 \text{ for all } i$$
Stability is scheduling dependent

Prioritize all classes $1, \ldots, L$

- Class 0 is served only if classes $1, \ldots, L$ are empty
- Stable iff $\rho_0 < \mathbb{P}(N_1 = 0, \ldots, N_L = 0) = \prod_{i=1}^{L} (1 - \rho_i)$
- More stringent stability condition

**Proposition [VBN05]:** In a linear network, size-based scheduling (like SRPT and LAS) may lead to instability at arbitrarily low loads.

α-fair bandwidth-sharing policies

find \( s_i(t) \) that

\[
\max \sum_{i=0}^{L} N_i(t)\alpha \frac{s_i(t)\alpha}{1-\alpha}
\]

s.t.

\[
\sum_{i \in r} s_i(t) \leq C_r
\]

- \( \alpha = 0 \): Maximizes throughput:

\[
\max \sum_{i=0}^{L} s_i(t)
\]

- \( \alpha \to 1 \): Proportional fairness

- \( \alpha = 2 \): TCP

- \( \alpha \to \infty \): Max-min fairness

\[
\phi_0 = \frac{n_0}{n_0 + (n_1^\alpha + n_2^\alpha)^{1/\alpha}}, \quad \phi_1 = \phi_2 = 1 - \phi_0
\]

Stability of $\alpha$-fair allocation

The process $\left(\overrightarrow{N}(t)\right)_{t \geq 0} = \left(N_1(t), \ldots, N_L(t)\right)_{t \geq 0}$

is Markovian with transition rates

$$
\begin{align*}
\left\{
\begin{array}{ll}
\left(\overrightarrow{N}(t)\right) \rightarrow \left(\overrightarrow{N}(t)\right) + e_i : & \lambda_i \\
\left(\overrightarrow{N}(t)\right) \rightarrow \left(\overrightarrow{N}(t)\right) - e_i : & \mu_i s_i(t)
\end{array}
\right.
\end{align*}
$$

Proposition [BM01]: The process $\left(\overrightarrow{N}(t)\right)_{t \geq 0}$ is stable under the necessary and sufficient condition $\sum_{i \in r} \rho_i \leq C_r$, for all $r$.

T. Bonald, L. Massoulié, Impact of fairness on Internet performance. SIGMETRICS/Performance 2001: 82-91
Optimal scheduling

Determine the policy that minimizes

$$\sum_{i=0}^{2} E(N_i)$$

The optimal policy will be a function of the entire state-space

Summary

- Stability policy dependent

- Optimality only for a linear network with two nodes

- $\alpha$-fair: large class of stable policies
"local" optimization

Consider a network with

• a general topology,
• generally distributed flow sizes

\[ \Pi^\circ = \text{family of stable bandwidth allocation policies.} \]

\[ Z_r^\pi(t) = \phi_r^\pi(N_r^\pi(t)) \]

where

• \( Z_r(t) = \text{total bandwidth allocated to class } r \text{ at time } t \)
• \( N_r(t) = \text{number of flows on route } r \text{ at time } t \)
• \( N(t) = (N_r(t); r \in R) \)
Let $\pi \in \Pi^o$ be fixed.

$\tilde{\pi}$ = a modified policy

• with the same inter-route allocation process,

\[ Z_{r}^{\tilde{\pi}} (t) = Z_{r}^{\pi} (t) = \phi_{r}^{\pi} (N^{\pi} (t)) \]

• but the intra-route disciplines may be different from the original ones

$\pi'$ = modified policy that applies SRPT

$\pi^*$ = the modified policy that applies LAS
Local optimality

Proposition:

Let \( \pi \in \Pi^o, r \in R \) and \( t \geq 0 \).

Then \( N_{r}^{\pi'}(t) \leq N_{r}^{\tilde{\pi}}(t) \)

among all the size-aware modifications \( \tilde{\pi} \)

and \( N_{r}^{\pi^*}(t) \leq_{st} N_{r}^{\tilde{\pi}}(t) \)

among all size-unaware modifications \( \tilde{\pi} \)
Simulations

Symmetric linear network with $L = 2$ and unit capacities

Poisson arrivals with constant total rate $\lambda = 1$

Flow size distribution with mean $b = 0.8$

- hyperexponential: $p_1 = 0.9$, $\mu_1 = 9/b$; $p_2 = 0.1$, $\mu_2 = 1/9b$

Comparison between $\pi$, $\pi'$ using basic policies, $\alpha$-fair ($\alpha=1$), PR0, PR12
### Simulations (cont.)

<table>
<thead>
<tr>
<th>$\lambda_0$</th>
<th>$\lambda_1$</th>
<th>$\lambda_2$</th>
<th>$N_\pi$</th>
<th>$N_\pi'$</th>
<th>$\alpha \to \infty$</th>
<th>$N_{\pi}$</th>
<th>$N'_{\pi}$</th>
<th>$\alpha \to 0$</th>
<th>$N_{\pi}$</th>
<th>$N'_{\pi}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
<td>3.37</td>
<td>1.98</td>
<td>9.10</td>
<td>8.09</td>
<td>4.40</td>
<td>3.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.6</td>
<td>0.2</td>
<td>0.2</td>
<td>2.63</td>
<td>1.82</td>
<td>7.10</td>
<td>6.44</td>
<td>3.46</td>
<td>2.58</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.4</td>
<td>0.2</td>
<td>0.2</td>
<td>2.07</td>
<td>1.58</td>
<td>4.62</td>
<td>4.16</td>
<td>2.72</td>
<td>2.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
<td>1.65</td>
<td>1.33</td>
<td>2.80</td>
<td>2.44</td>
<td>2.01</td>
<td>1.61</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TCP overview

SS – Slow Start
CA – Congestion Avoidance
MR – Maximal Rate
TO – Time Out
Bringing the scheme into practice

Difficult to “improve” upon an algorithm
→ If we change TCP for a set of flows, their performance might get better, but it will get worse for other flows.

Objective: improving the performance for a set of flows without degrading the rest

Basic idea

For any given TCP algorithm: “Schedule the packets of a set of flows sharing the same origin-destination route, without modifying the bandwidth share that would have been perceived using the given TCP implementation”

Decoupling of congestion and scheduling:
- transmission epochs determined by TCP
- contents of the segment by $\pi$

Two main questions:
- Can it be non-intrusive?
- Performance gain?
How to measure “intrusiveness”?

Let $\text{tcp}$ denote a standard TCP implementation.

Let $V^{\text{tcp}}(t)$ denote the amount of traffic injected by TCP on a given route.

Let $g_{\text{TCP}}(\pi)$ be a general congestion-control.

We say that $g_{\text{TCP}}(\pi)$ is non-intrusive if for any sample-path, and all time $t$, $V^{\text{tcp}}(t)= V^{g_{\text{tcp}}(\pi)}(t)$.
**Proposition:** If the sender and receiver’s buffers are unbounded, then \( V^{tcp}(t) = V^{gtcp}(\pi)(t) \)

**Main idea:**
- Maintain virtual queues.
- Identify and acknowledge packet’s content.

*(Note: we ignore overheads, extra processing times etc.)*
If buffers are unbounded, time of events under TCP can be exactly reproduced

→ gtcp is “non-intrusive”, and we can decouple “congestion” and “scheduling”.

→ gtcp can implement any scheduling policy $\pi$

**Note:** We assume packets of same size.

*If sizes were different, we would need to encapsulate several segments of different queues in the same service*

→ needs a more complex messaging protocol
Local optimality of size-based

Under the infinite buffer conditions, if segments are neither lost nor reordered, we have

\[ N^{SRPT}(t) \leq N^{gtcp}(t), \]

\[ P(N^{LAS}(t)>k) \leq P(N^{gtcp}(t)>k), \text{ if distribution is DHR} \]

\[ T_i^{FAIR} \leq T_i^{TCP} \]

FAIR: Serve the flow that would finish next under TCP (size-aware discipline)
summary

Infinite buffers for the “non-intrusiveness”

and absence of losses and reordering for “optimality”

but what will happen in reality?
On the technical conditions

Schedulability

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Shared/Competing</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Routes (number)</td>
<td>1788796</td>
<td>102981</td>
<td>5.7%</td>
</tr>
<tr>
<td>Flows (number)</td>
<td>5155554</td>
<td>937033</td>
<td>18.1%</td>
</tr>
<tr>
<td>Flows (volumes in KB)</td>
<td>261769672</td>
<td>152530580</td>
<td>58.2%</td>
</tr>
</tbody>
</table>

Unbounded buffers

Size of packets

Absence of loses: ECN can be of help
Sched-TCP: an implementation of gtcp

A User implementation: we schedule segments that are transmitted to the transport layer… but we don’t control exactly which packet is transmitted by TCP.

We run experiments between Lionel’s house and LAAS.
Scenario 1

Scenario with 3 flows (short, long and sample flow)

100 repetitions

Comparison of SRPT and TCP
(almost) non-intrusiveness
Flow arrivals follow a Poisson process,

Pareto distribution of flow size \( \mathbb{P}(S > x) = \frac{1}{(1 + cx)^\alpha} \)

\( \mathbb{E}(S) = 5MB, \ \lambda \ \text{is chosen so load 0.9} \)
Distribution of the number of active connections
Distribution of transfer time
transfer time vs. flow size
Some statistics...

<table>
<thead>
<tr>
<th></th>
<th>Faster</th>
<th>Slower</th>
<th>Equal</th>
<th>Gain</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS</td>
<td>61%</td>
<td>25.6%</td>
<td>13.2%</td>
<td>-0.57s</td>
<td>+314ms</td>
</tr>
<tr>
<td>SRPT</td>
<td>88.8%</td>
<td>3.2%</td>
<td>8%</td>
<td>-1.11s</td>
<td>+28ms</td>
</tr>
<tr>
<td>FAIR</td>
<td>89.6%</td>
<td>2.4%</td>
<td>7.9%</td>
<td>-1.13s</td>
<td>+33ms</td>
</tr>
</tbody>
</table>
Conclusions

Scheduling and congestion control are 2 different services that can be decoupled for a given route.

Decoupling mechanism provides a framework to compare and improve upon scheduling policies.

With *gtcp* is possible to improve the performance of any TCP version:

- scalable and incrementally deployable.
- but what is the criteria to be considered?

Size-based discipline can reduce the average latency. We can schedule TCP flows using any arbitrary scheduling discipline.
Future work

Application to situation where concurrent flows are present like AJAX (Google Maps etc.), Chromebook

What is the sub-optimality gap?

Release the code and more experiments