Non-intrusive scheduling of flows in networks

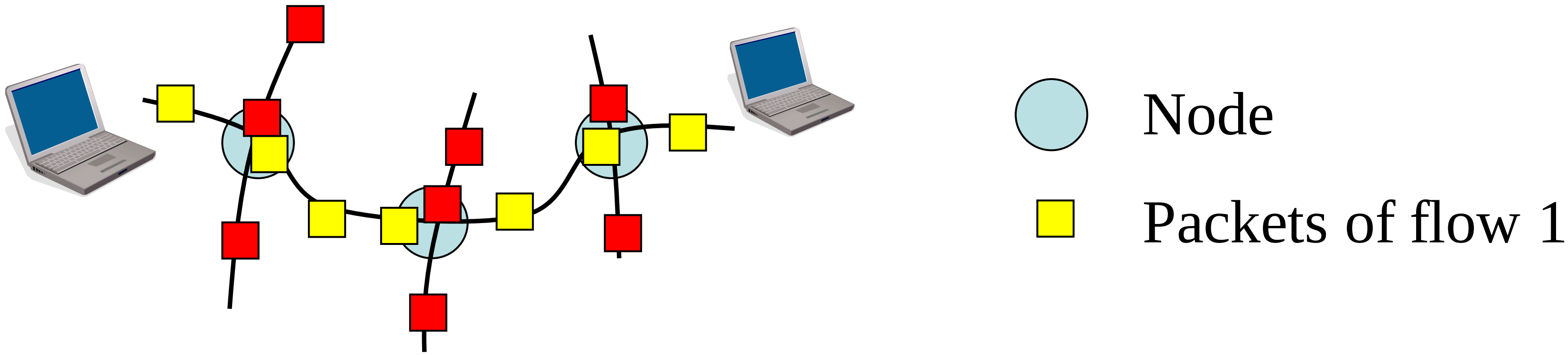
Urtzi Ayesta

LAAS-CNRS & Ikerbasque

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

Eindhoven, 02/06/2015

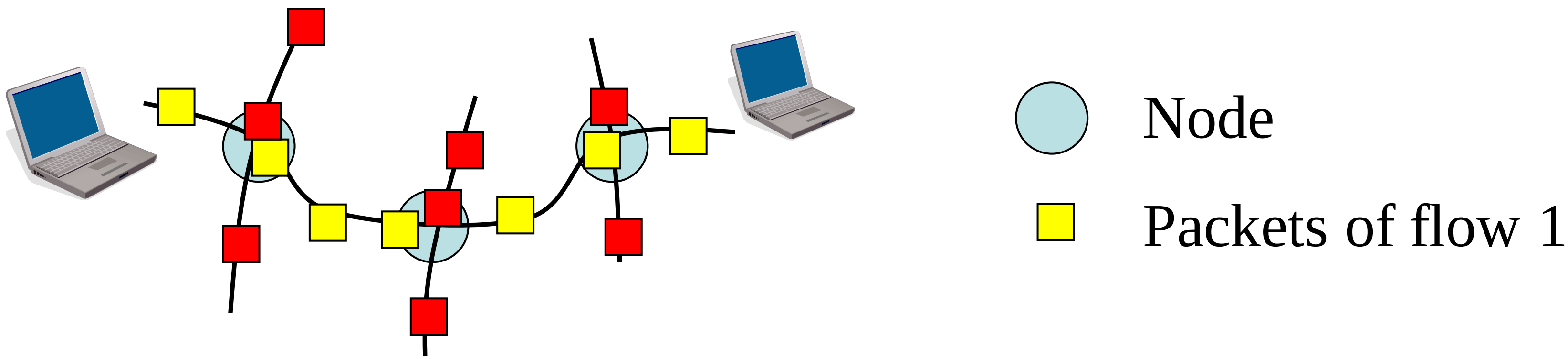
Traffic flows (web pages, email, music and movies…)



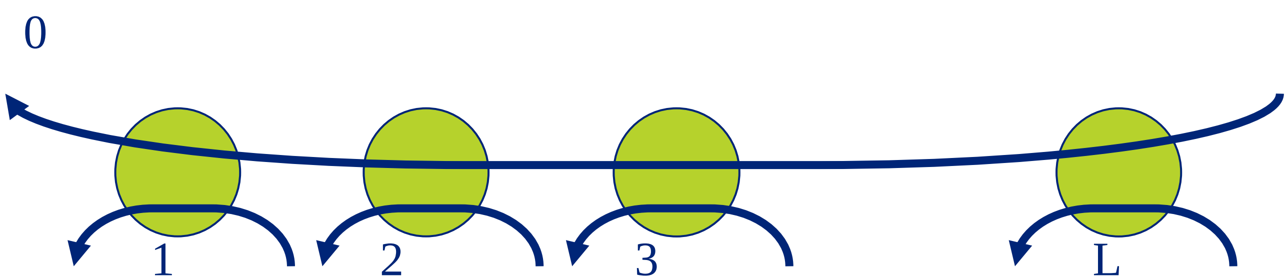
For a given performance criterion, what is the optimal way to schedule flows?

Two “type” of networks

– Internet: packets, congestion control, TCP etc.



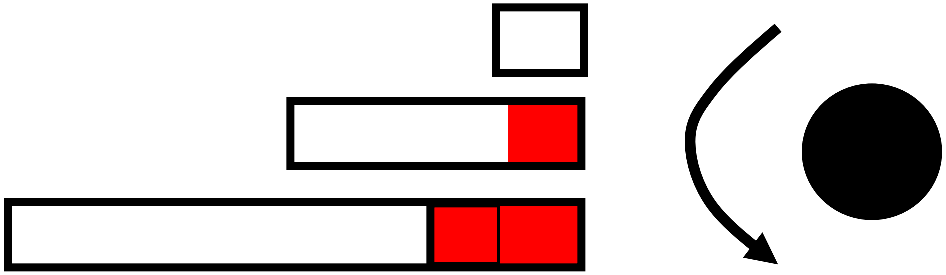
– Bandwidth-sharing network



Outline

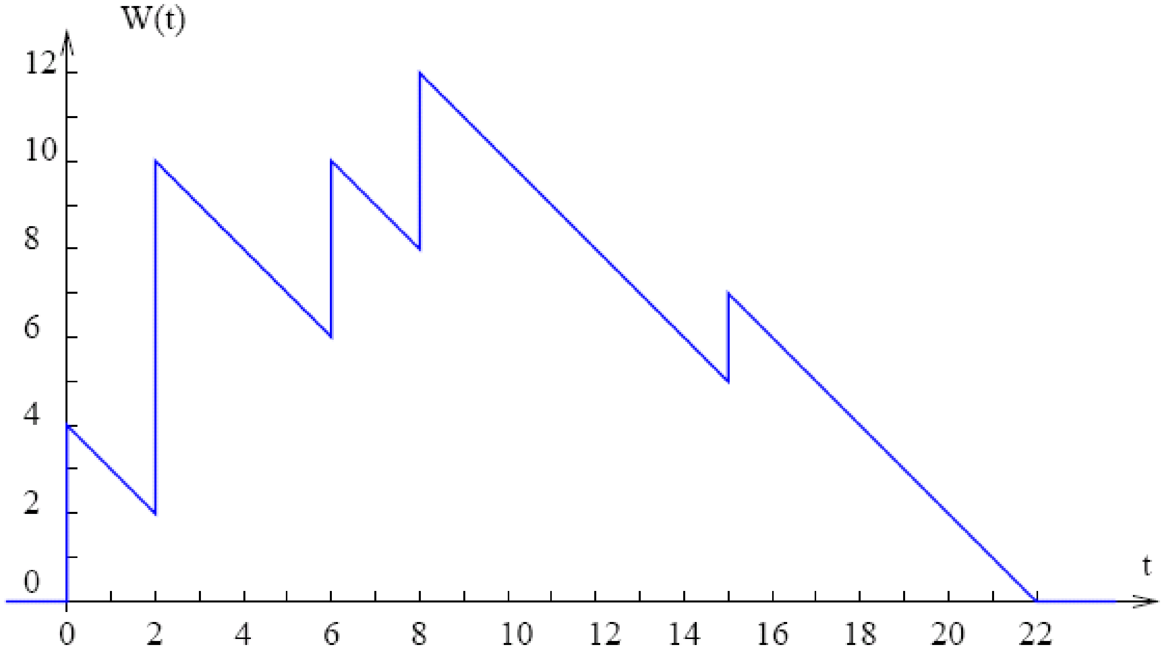
- Scheduling in networks
- Non-intrusive scheduling
  “Local optimality” of size-based
- gTCP: Non-intrusive TCP
  - Decoupling congestion control and scheduling
- Experimental results

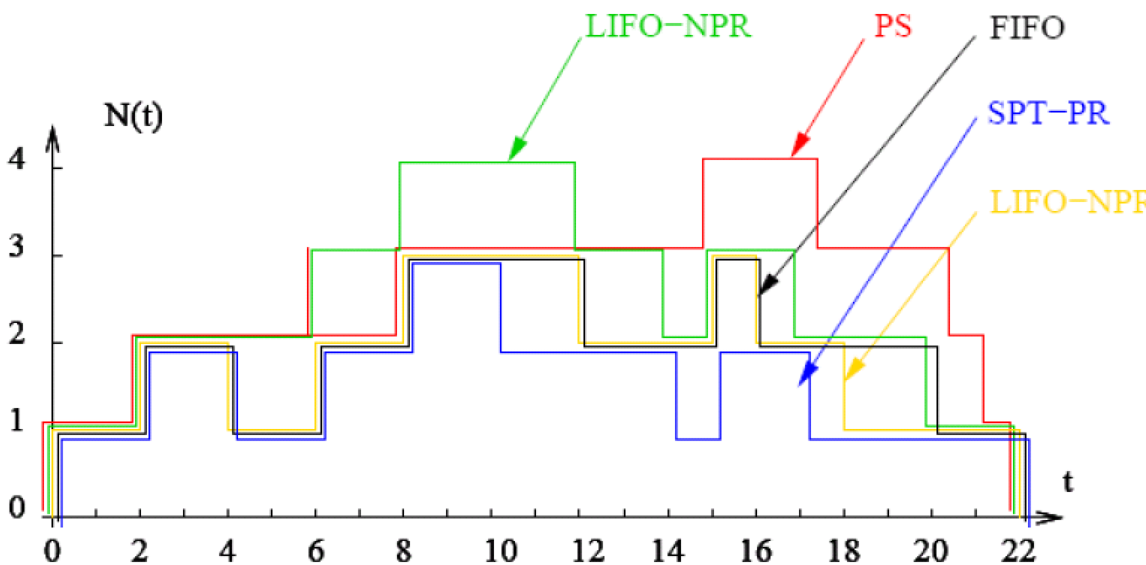
Single link



Broad literature:

- optimality of size-based scheduling:
    - SRPT among size-aware policies
    - LAS among size-unaware policies
- Exact performance analysis for many disciplines

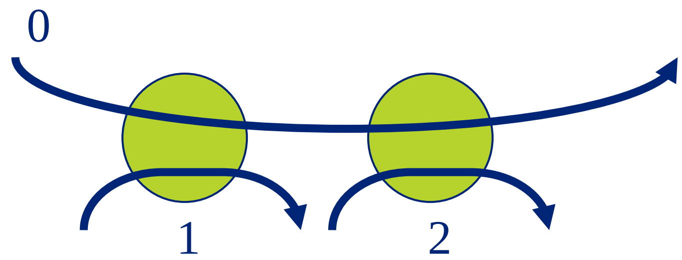

W(t)
12
10
8
6
4
2
0
t
0
2
4
6
8
10
12
14
16
18
20
22

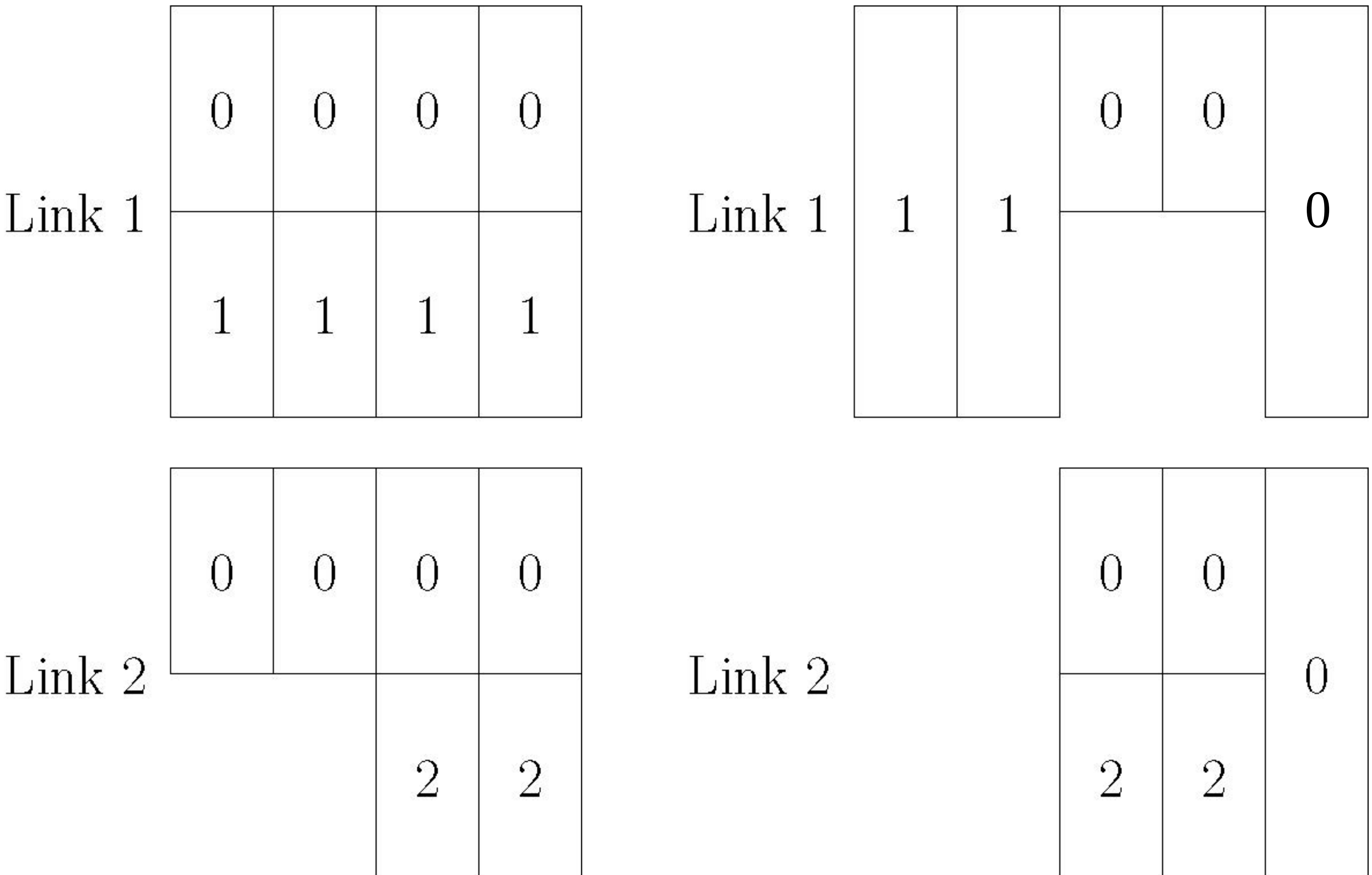

LIFO−NPR
PS
FIFO
SPT−PR
LIFO−NPR
N(t)
4
3
2
1
0
t
0
2
4
6
8
10
12
14
16
18
20
22

Flow 0 starts at t=0, size 2,

flow 1 at t=0, size 2

flow 2 starts at t=2, size 1

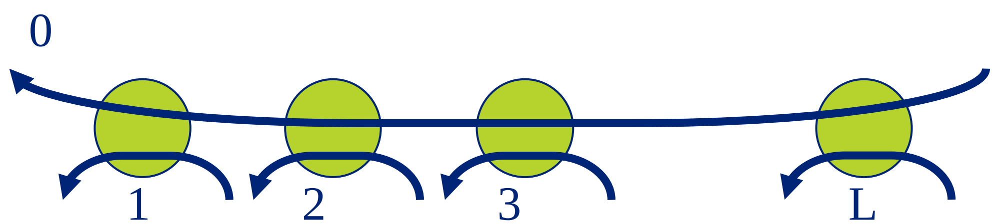




Stability is scheduling dependent

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

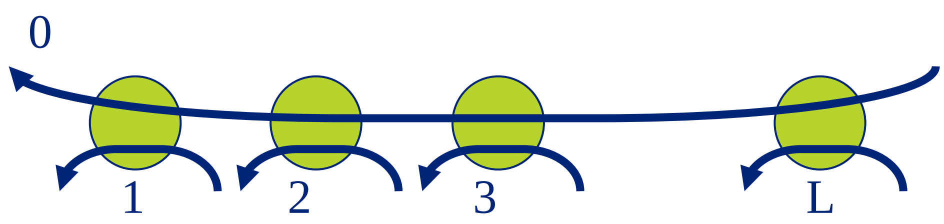
- Let us consider a linear network



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

$$\boldsymbol{\rho_0+\rho_i < 1 \text{ for all } i}$$

Stability is scheduling dependent



Prioritize all classes *1,…,L*

- Class 0 is served only if classes 1,…,*L* are empty
- Stable iff $$\rho_0 < \mathrm{P}\left(N_1 = 0,\ldots,N_L = 0\right) = \prod_{i=1}^{L}\left(1-\rho_i\right)$$
- 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.

I.M. Verloop, S.C. Borst, R. Núñez-Queija, Stability of size-based scheduling disciplines in resource-sharing networks, Performance Evaluation 62, 247-262, 2005

α-fair bandwidth-sharing policies

find $s_i(t)$ that $\max \sum_{i=0}^{L} N_i(t)^\alpha \dfrac{s_i(t)^\alpha}{1-\alpha}$

J. Mo, J. Walrand, Fair end-to-end window-based congestion control. IEEE/ACM ToN, 8(5): 556-567, 2000

$$s.t. \quad \sum_{i \ni r} s_i(t) \le C_r$$

- $\alpha = 0$: Maximizes throughput: $\max \sum_{i=0}^{L} s_i(t)$
- $\alpha \rightarrow 1$: Proportional fairness
- $\alpha = 2$: TCP
- $\alpha \rightarrow \infty$: Max-min fairness

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

Stability of **α**-fair allocation

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

is Markovian with transition rates

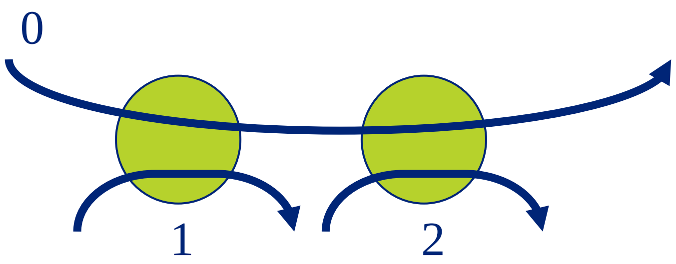
$$\begin{cases} \left(\vec{N}(t)\right) \to \left(\vec{N}(t)\right) + \vec{e}_i : & \lambda_i \\ \left(\vec{N}(t)\right) \to \left(\vec{N}(t)\right) - \vec{e}_i : & \mu_i s_i(t) \end{cases}$$

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

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

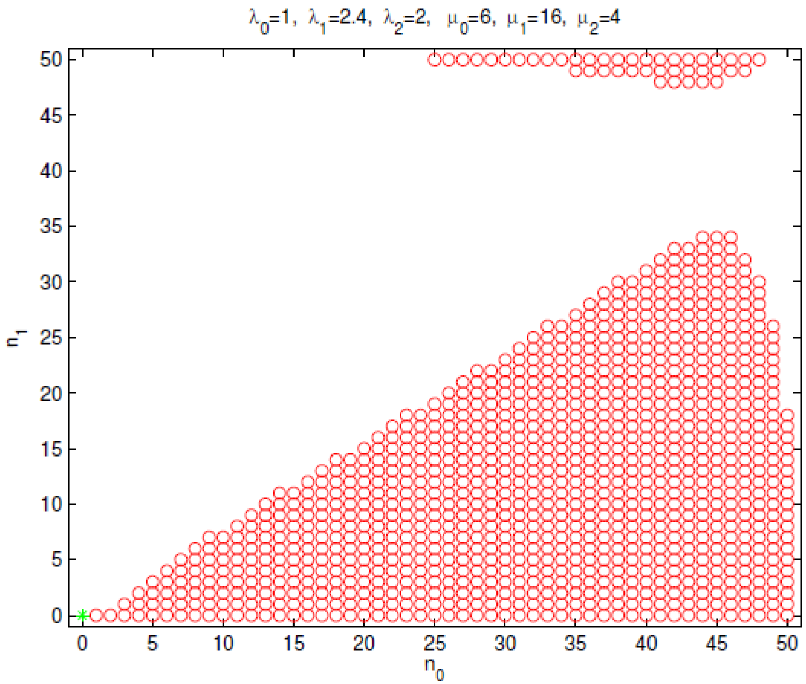
Optimal scheduling

Determine the policy that minimizes $\sum_{i=0}^{2} E(N_i)$



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

I.M. Verloop, S.C. Borst, R. Núñez Queija, Delay optimization in bandwidth-sharing networks. Proc. CISS 2006 Conference on Information Sciences and Systems



Summary

- Stability policy dependent

- Optimality only for a linear network with two nodes

- α-fair: large class of stable policies

"local" optimization

S. Aalto, U. Ayesta, SRPT applied to bandwidth-sharing networks, ANOR 170, 3-19, 2009.

Consider a network with

- a general topology,
- generally distributed flow sizes

Π° = family of stable bandwidth allocation policies.

$$Z_r^\pi(t) = \phi_r^\pi(\mathbf{N}^\pi(t))$$

where

- $Z_r(t)$ = total bandwidth allocated to class r at time t
- $N_r(t)$ = number of flows on route r at time t
- $\mathbf{N}(t) = (N_r(t);\ r \in R)$

Let $\pi \in \Pi^o$ be fixed.

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

- with the same inter-route allocation process,

$$Z_r^{\tilde{\pi}}(t) = Z_r^{\pi}(t) = \phi_r^{\pi}(\mathbf{N}^{\pi}(t))$$

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

π' = modified policy that applies SRPT

π^* = the modified policy that applies LAS

Local optimality

Proposition:

Let $\pi \in \Pi^{\circ}, r \in R$ 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 $b = 0.8$

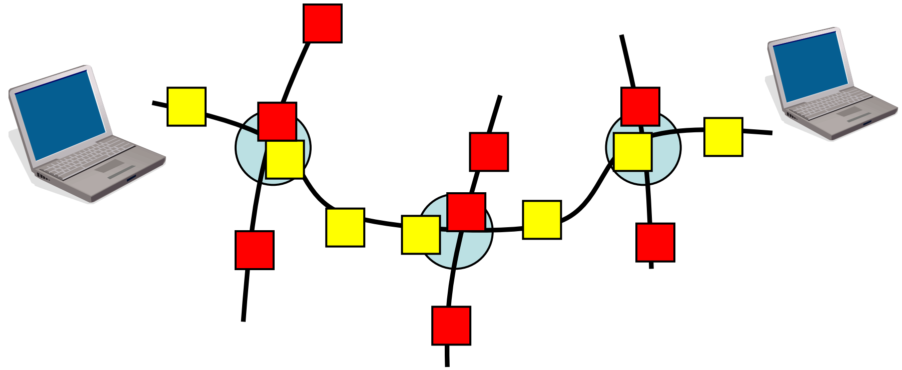
- hyperexponential: $p_1 = 0.9$, $\mu_1 = 9/b$; $p_2 = 0.1$, $\mu_2 = 1/9b$

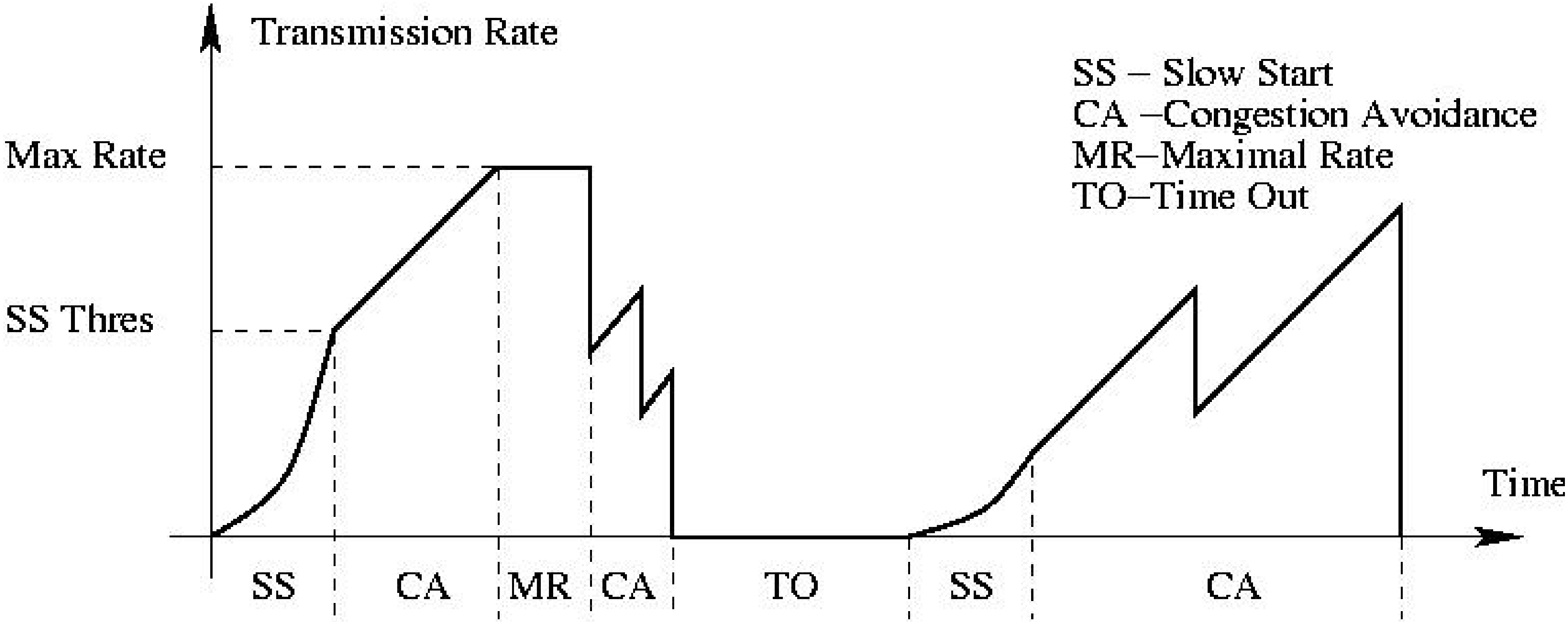
Comparison between π, π' using basic policies, α-fair (α=1), PR0, PR12

Simulations (cont.)

			$\alpha=1$		$\alpha\to\infty$		$\alpha\to 0$	
λ_0	λ_1	λ_2	N^{π}	$N^{\pi'}$	N^{π}	$N^{\pi'}$	N^{π}	$N^{\pi'}$
0.8	0.1	0.1	3.37	1.98	9.10	8.09	4.40	3.01
0.6	0.2	0.2	2.63	1.82	7.10	6.44	3.46	2.58
0.4	0.2	0.2	2.07	1.58	4.62	4.16	2.72	2.14
0.2	0.4	0.4	1.65	1.33	2.80	2.44	2.01	1.61

TCP overview





➔ 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

U. Ayesta, D. Carvin, L. Bertaux, Non-Intrusive scheduling of TCP flows, Proceedings of IFIP Networking 2015.

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 $\boldsymbol{\pi}$

Two main **questions**:

- Can it be **non-intrusive**?
- Performance gain ?

How to measure “intrusiveness”?

Let **tcp** denote a standard TCP implementation

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

Let **gTCP(π)** be a general congestion-control

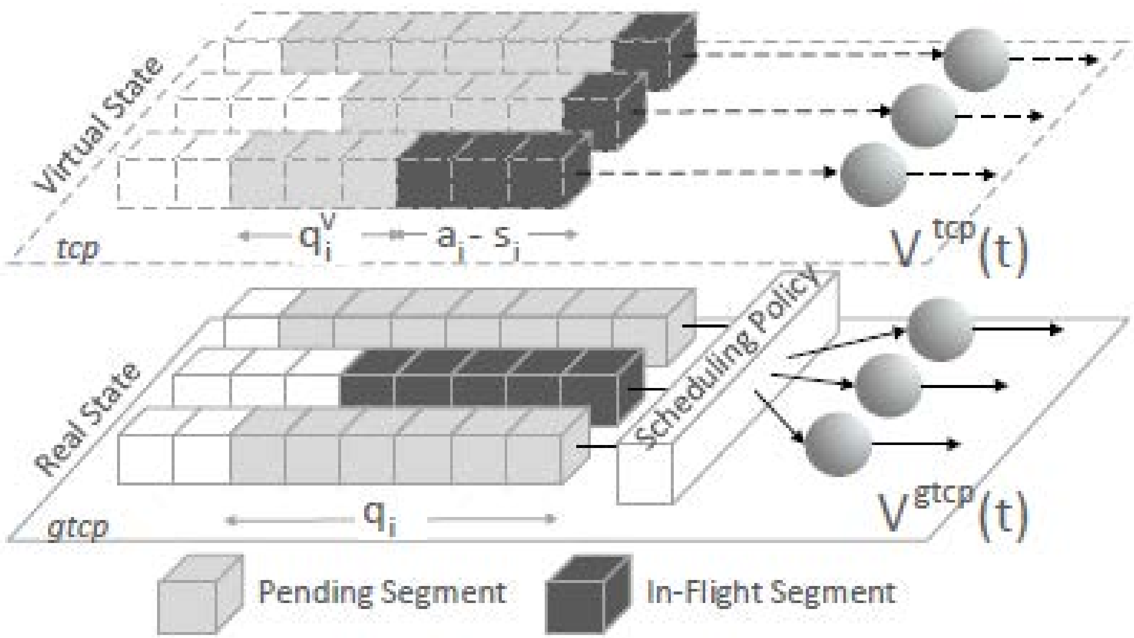
We say that **gTCP(π)** is **non-intrusive** if for any sample-path, and all time t, $\mathbf{V^{tcp}(t) = V^{gtcp(\pi)}(t)}$

Proposition: If the sender and receiver's buffers are unbounded, then $\mathbf{V^{tcp}(t) = V^{gtcp(\pi)}(t)}$

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 **π**

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

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

$P(N^{LAS}(t)>k) \leq P(N^{gtcp}(t)>k)$, 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

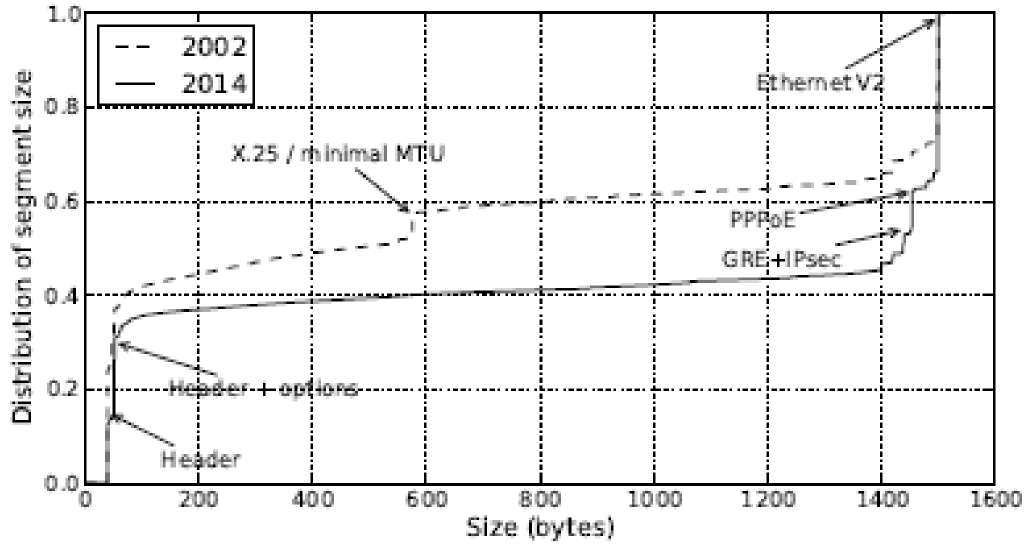
Schedulability

PROPORTION OF SCHEDULABLE TRAFFIC

	Total	Shared/Competing	Ratio
Routes (number)	1788796	102981	5.7%
Flows (number)	5155554	937033	18.1%
Flows (volumes in KB)	261769672	152530580	58.2%

Unbounded buffers

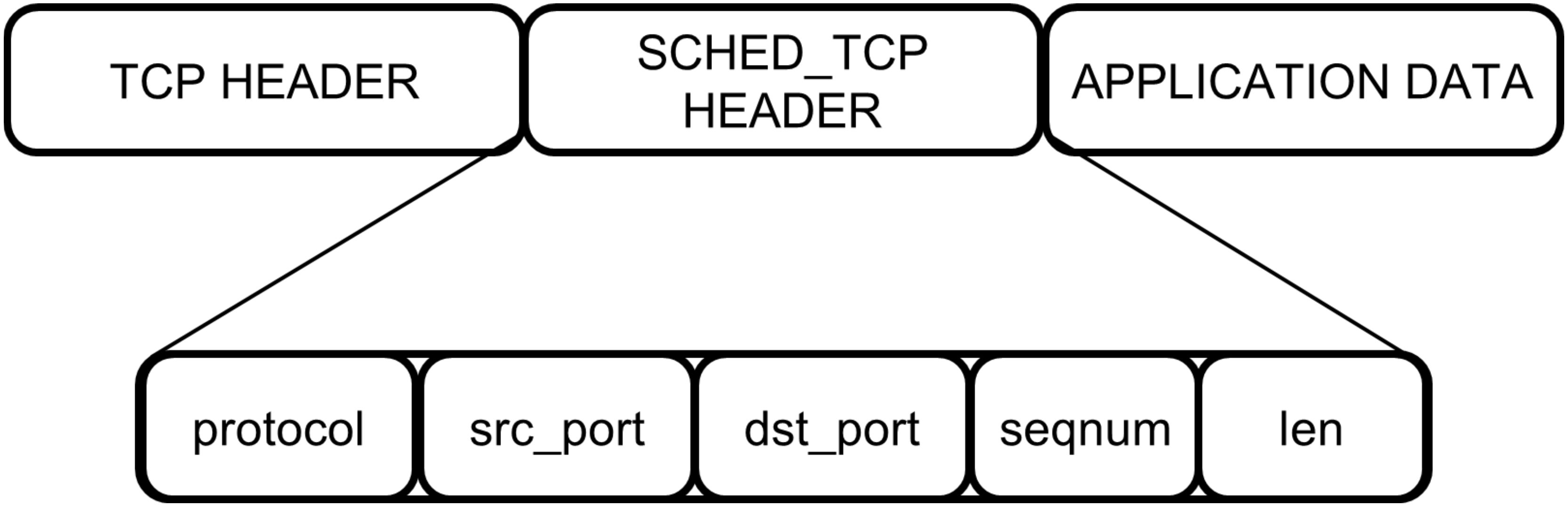
Size of packets



Absence of loses: ECN can be of help

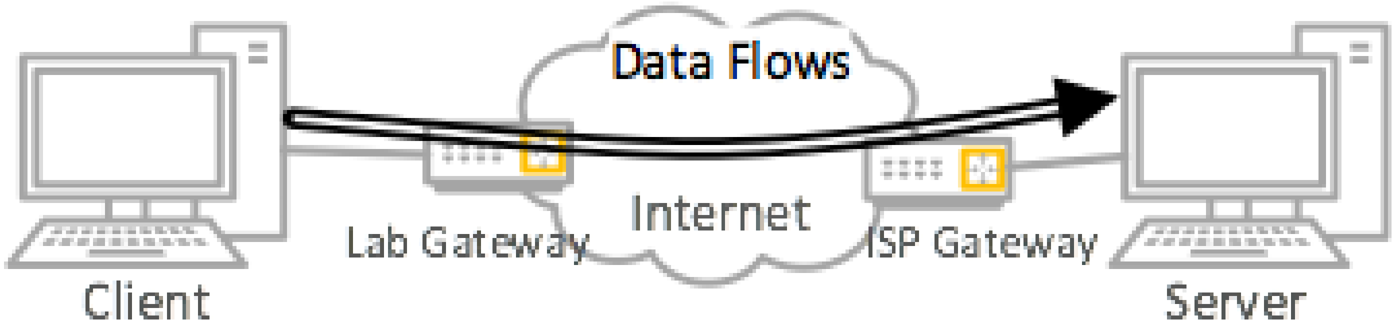
Sched-TCP: an implementation of gtcp

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

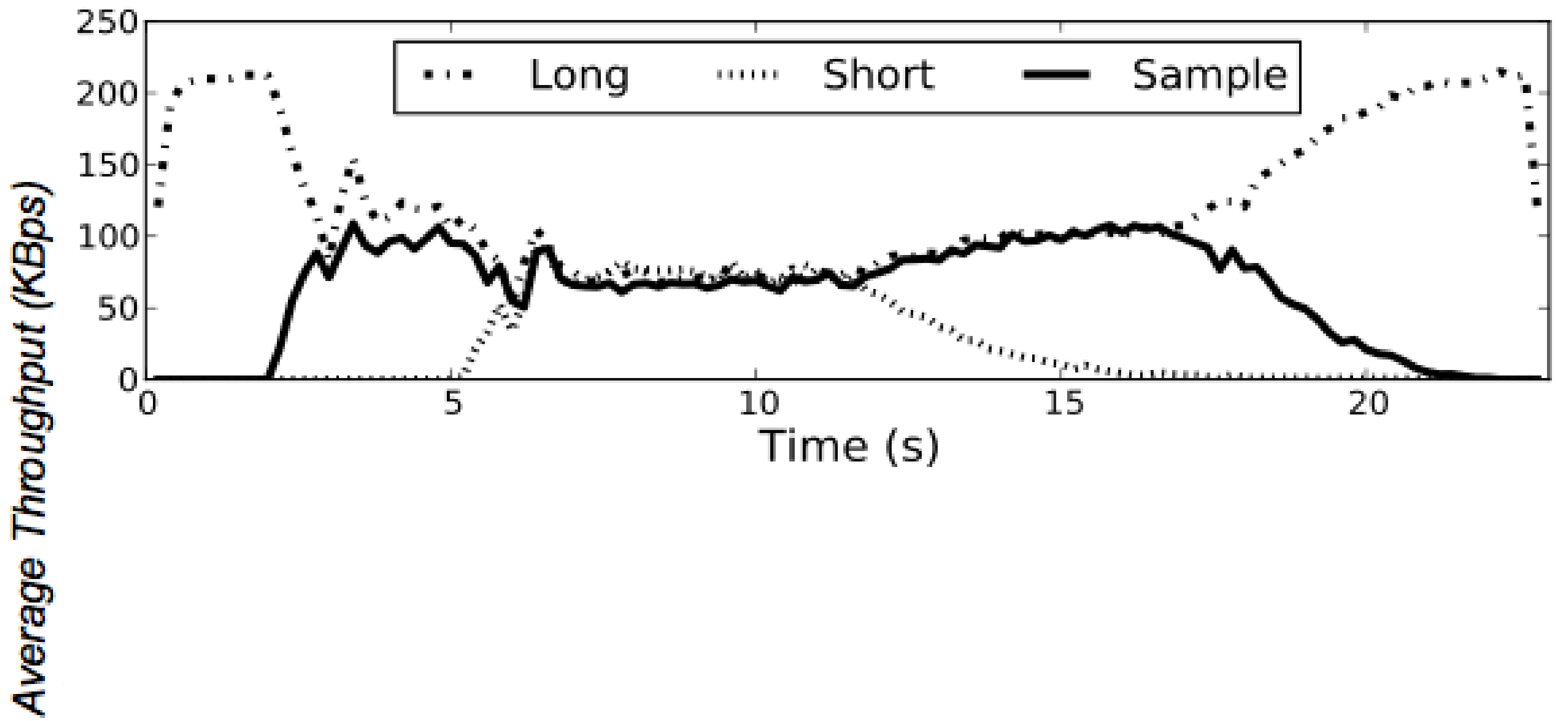


Scenario with 3 flows (short, long and sample flow)

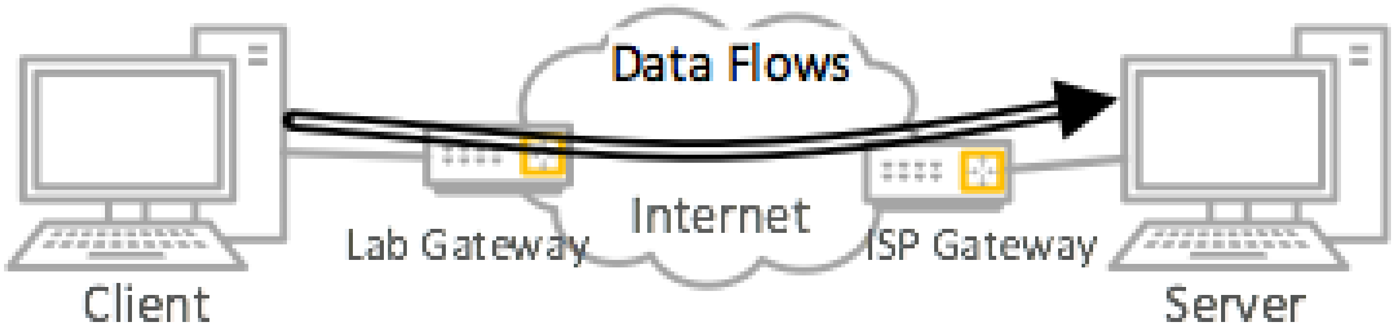
100 repetitions

Comparison of SRPT and TCP

(almost) non-intrusiveness



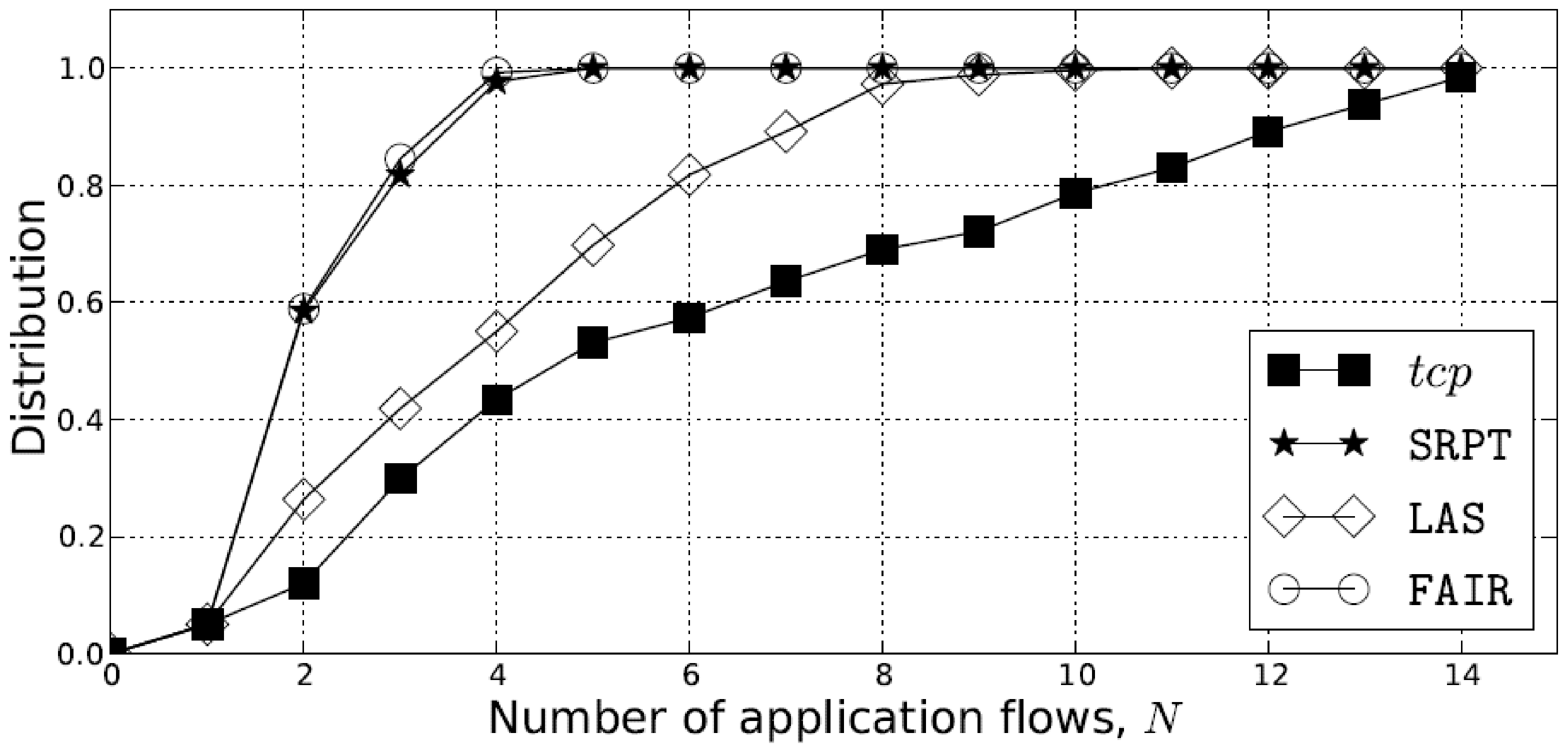
Scenario 2



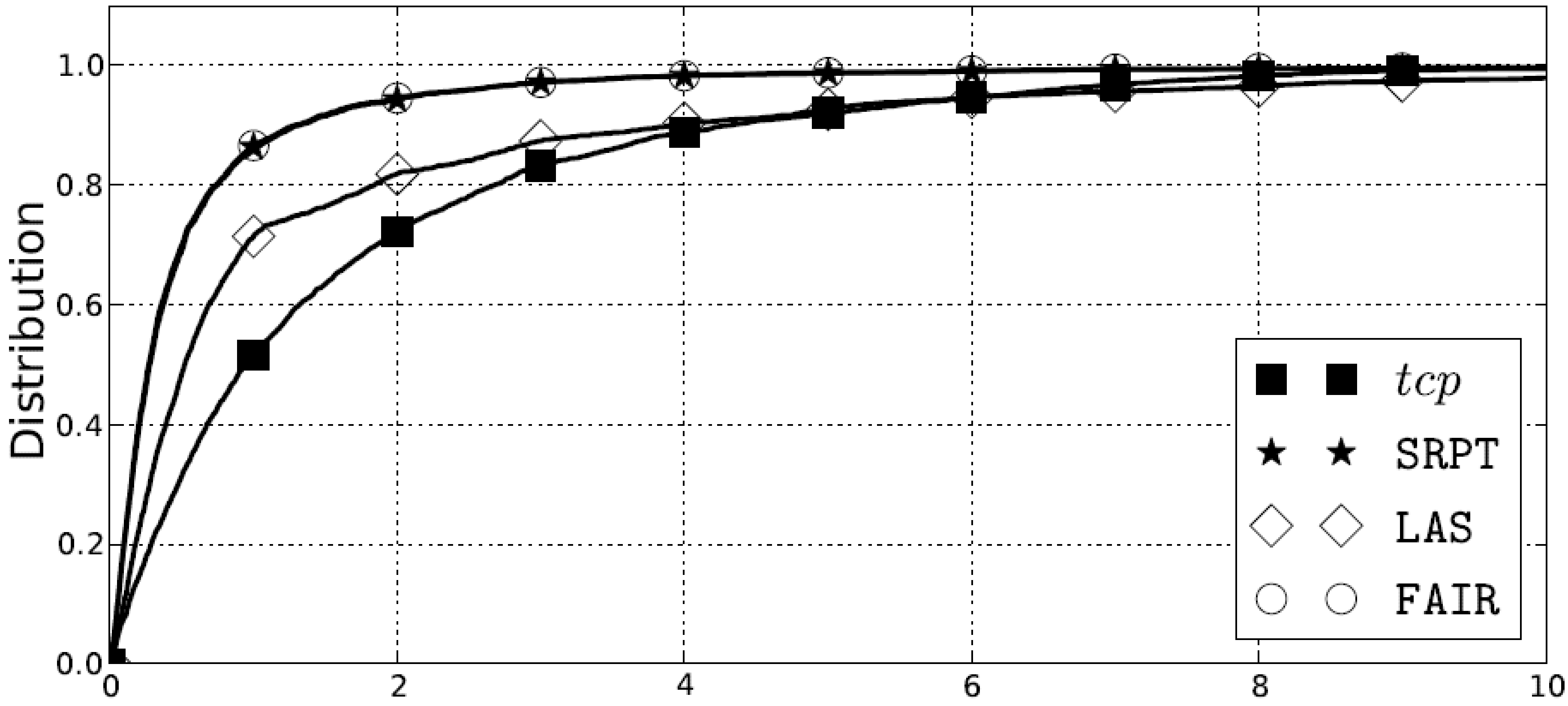
Flow arrivals follow a Poisson process,

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

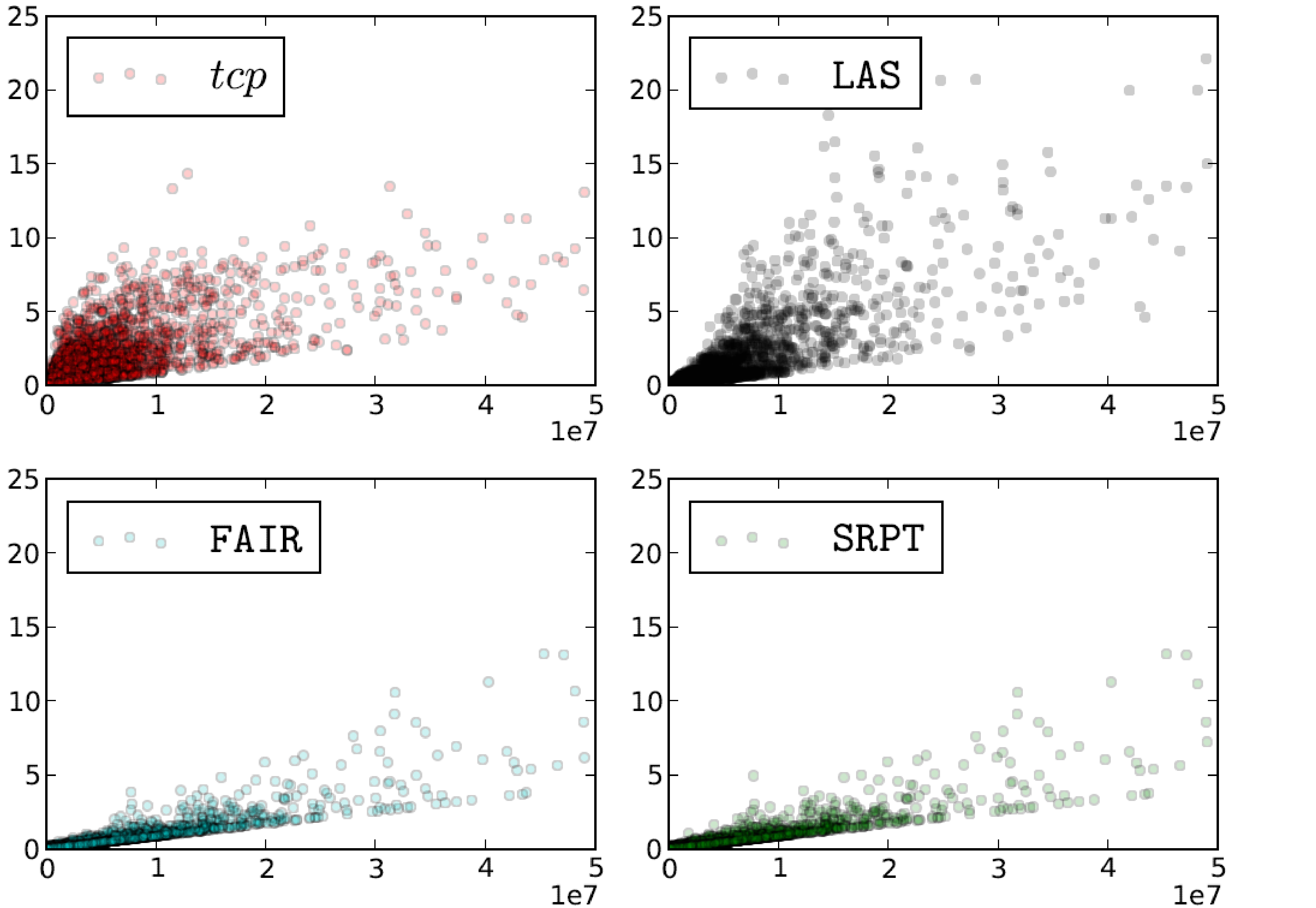
$\mathbb{E}(S) = 5MB$, λ is chosen so load 0.9



Distribution of the number of active connections



Distribution of transfer time



transfer time vs. flow size

Some statistics...

TABLE II

STATISTICS ON TRANSFER TIME OF DIFFERENT SCHEDULING POLICIES.

	Faster	*Slower*	*Equal*	$\overline{Gain}$	$\overline{Loss}$
LAS	61%	25.6%	13.2%	-0.57s	+314ms
SRPT	88.8%	3.2%	8%	-1.11s	+28ms
FAIR	89.6%	2.4%	7.9%	-1.13s	+33ms

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