A study of large flow interactions in high-speed shared networks with Grid5000 and GtrcNET-10 instruments

Romaric Guillier, Ludovic Hablot, Yuetsu Kodama, Tomohiro Kudoh, Fumihiro Okazaki, Pascale Primet, Sébastien Soudan, Ryousei Takano

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

A study of large flow interactions in high-speed shared networks with Grid5000 and GtrcNET-10 instruments

Romaric Guillier,
Ludovic Hablot,
Yuetsu Kodama,
Tomohiro Kudoh,
Fumihiro Okazaki,
Pascale Primet,
Sébastien Soudan,
Ryousei Takano

November 2006

Research Report N° 2006-43

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr

A study of large flow interactions in high-speed shared networks with Grid5000 and GtrcNET-10 instruments

Romaric Guillier, Ludovic Hablot, Yuetsu Kodama, Tomohiro Kudoh, Fumihiro Okazaki, Pascale Primet, Sébastien Soudan, Ryousei Takano

November 2006

Abstract

We consider the problem of huge data transfers and bandwidth sharing in contexts where transfer delay bounds are required. This report investigates large flow interactions in a real very high-speed network and aims at contributing to high-speed TCP variants evaluation by providing precise measurements. It then also gives an insight on the behaviour of emulated alternative protocols under different realistic congestion and long latency conditions in 10 Gbps experimental environments.

Keywords: bulk data transfers, bandwidth sharing, transfer delay predictability, transport protocol experimentation

Résumé

Dans cet article, nous étudions le problème de transferts massifs de données et du partage de bande passante dans les contextes où il est requis d'avoir des temps de transfert bornés. Nous présentons une étude des interactions de larges flux dans des réseaux très haut-débits et ainsi contribuer à l'évaluation de variantes de TCP, adaptée à ce contexte, en proposant des mesures précises. Nous donnerons aussi un aperçu du comportement de protocoles alternatifs émulé dans des environnements expérimentaux à 10 Gbps sous diverses conditions de congestions et de latences.

Mots-clés: partage de bande passante, expérimentation de protocole de transport, prédiction de temps de transfert total, transferts en masse de données

1 Introduction

The data volumes of future distributed applications such as data and computing grids, distance visualisation and high-end collaborative environment are in the order of terabytes and will likely reach petabytes in some cases. The movement of these data have demanding performance requirements such as reliable and predictable delivery [FFR+04]. This generates specific challenges on the transport protocol and its related mechanisms. The enhancement of TCP/IP is being intensively pursued to tackle limits that classical congestion control solutions encounter in large bandwidth-delay product environment [WHVBPa05]. A range of solutions is proposed and their properties have been analysed by simulation. However, few studies have measured the performance of these proposals in real high speed networks [MFVBP04, C+]. It is acknowledged that more real and systematic experiments are needed to have a better insight on the relevance of metrics, representative scenarii for protocol evaluation and on the potential usage of these protocols in particular applications [Flo06b].

This report contributes to this challenge by exploring several high data transfer scenarii in two experimental environments: the Grid5000[1] testbed and the AIST GtrcNET-10-based[2] testbed [KKT+04]. Grid5000, is an experimental grid platform gathering more than 3000 processors over nine geographically distributed sites in France (see figure 1), interconnected by a dedicated private high-speed network [Ca05]. The particularity of this testbed is to provide researchers with a fully reconfigurability feature to dynamically deploy any OS image or protocol stack on any host of the testbed. The other experimental environment we used is the GtrcNET-10-based emulated and controlled testbed connected within the AIST Super Cluster. Hosts of both testbeds have similar hardware and software configuration.

This report explores how the transport protocol enhancements could benefit to high-end applications in terms of data transfer efficiency and predictability in two environments. It is centred on elephant-like bulk data transfers in very high-capacity (1 Gbps, 10 Gbps) networks like grids are supposed to benefit from today. The systematic evaluation of the protocols in our controlled and realistic environment provides a set of measurements of several metrics proposed by [Flo06a]. In this context, we investigate, the different congestion control proposals as well as the fair sharing optimisation objective and their impact on the network resource utilisation and on individual application utility that have to be simultaneously optimised.

In cluster interconnection context, hundreds of hosts may simultaneously generate large flows through their gigabit interfaces. But as the access links between cluster networks and wide area networks currently offer between 1 to 10 Gbps rates, they constitute a strong bottleneck that may drastically increase the transfer delays and impact the overall distributed environment performance. In the Internet, the endpoints' access rates are generally much smaller (2 Mbps for DSL lines) than the backbone link's capacity (2.5 Gbps for an OC48 link). According to the law of large numbers, coexistence of many active flows smoothes the variation of load, and a link is not a bottleneck unless the load approaches its full capacity [Rob04]. To curb the load, distributed congestion control protocols such as TCP statistically share available bandwidth among flows in a "fair" way. In contrast, for high-end applications, the bandwidth demand of a single endpoint (1 Gbps, say) may be comparable to the capacity of bottleneck link. In such a low multiplexing environment, if no pro-active admission control is applied, a transient burst of load can easily cause active transfers have very long duration, miss their deadline or even fail. In addition, to complete more tasks before their respective deadlines, sharing instantaneous bandwidth fairly among all active flows is not optimal [GR06]. This

[1] http://www.grid5000.org

[2] http://projects.gtrc.aist.go.jp/gnet/gnet10p3e.html

is the reason why researchers [**?**] propose to introduce access control and flow scheduling. This could harmonise network resource management with other grid resources management and serve the global optimisation objective. The ultimate goal of this research is, in different scenarii, to answer questions such as: which transport protocol to use in a given context? How many flows to schedule to obtain minimum interaction and maximum throughput? When starting them to avoid bad interactions during slow start phase?...

The rest of the article is organised as follows. Section 2 gives some insights on parameter space and metrics. Scenario and experiments are described in section 3. Results are discussed in section 4. The article concludes in section 5.

2 Methodology

When the 10 Gbps infrastructure has been set up in Grid5000, very simple experiments, that any grid user could do, were run to see how grid applications could benefit from the deployed network. These tests have shown very disappointing results [GHPS06]. The hosts were not able to obtain correct throughput (45 to 100 Mbps) when competing and the aggregate throughput was very low (about 3 to 6 Gb/s). We then choose to investigate three types of scenario to understand these bad results and improve them in this 10Gb/s context. This work has been inspired by the results and methodologies proposed by [LLS06, Flo06b, HLRX06]. [Flo06b] identifies several characteristics and describes which aspect of evaluation scenario determine these characteristics and how they can affect the results of the experiments. This helped us in defining workloads and metrics that are going to be presented in the following section.

2.1 Traffic characteristics

According to [Flo06b], the aggregated traffic on a link is characterised by:

a) the distribution of per-packet round-trip time

b) the file sizes

c) the packet sizes

d) the ratio between forward-path and reverse-path traffic

e) the distribution of peak flow rates

f) the distribution of transport protocols

Despite no extensive study of grid traffic exists, we assume the specific context we study here presents the following specificities:

a) The distribution of per-packet round-trip time is multi-modal. Nodes are generally clustered, consequently, several modes may appear (about $\frac{N*(N-1)}{2}$ modes for N sites), each mode of the distribution representing the set of given site to site connections.

b) File sizes are not exponentially distributed. For example, in Data Grid like LCG (for LHC) file size and data distribution is defined by the sampling rate of data acquisition. The traffic profile is then highly uniform.

c) Packet sizes are also mostly constant, with a large proportion of packets having the maximum size (the Ethernet MTU).

d) The ratio between forward-path and reverse-path traffic is unknown and depends on the location of the storage elements within the global grid.

e) Distribution of peak flow rates may also be uniform.

f) Today, most of grid applications need reliable transport and use TCP-based protocols. The distribution of transport protocols is modal.

In the rest of the paper, we call these specific conditions, the "grid context". The next section presents the various scenarii we implemented in this context to study the interactions of large TCP flows.

2.2 Scenarii

We examine two types of features that can help users to obtain good performance in such context: parallel streams and TCP variants.

We investigate the two following types of scenarii:

- Range of TCP variants in the Grid5000 real testbed and in the AIST-GtrcNET10 testbed with Grid5000 latency.
- Range of TCP variants with a range of emulated latency in the AIST-GtrcNET-10 testbed.

Different TCP variants have been proposed to improve the response function of AIMD congestion control algorithm in high bandwidth delay product networks. All these protocols are not equivalent and not suited for every context. We investigate here their behaviours in our "grid context" in two different testbeds and we also provide comparaison grounds between our two testbeds.

The following section describes the various parameters and metrics that have been used to chara-terise the behaviour of the TCP flows.

2.3 Measured parameters and metrics

We design and configure our experimental testbeds to have a direct access to the following variable measurements during experiments:

a) Goodput using *iperf* on the receiver side, that corresponds to the actual amount of bandwidth that is available for the high-end applications, i.e. retransmissions and headers are not taken into account.

b) Aggregated throughput via the GtrcNET-10 on the 10 Gbps shared link.

c) TCP kernel variables with the Web100 patch on senders and receivers.

The parameters are evolving along the three following axis:

1) TCP variant, among {Reno, BIC, CUBIC, HighSpeed, H-TCP and Scalable}

2) RTT, ranging from 0 ms to 200 ms

3) congestion level, the ratio between the sum of sources' access link capcity and the bottleneck size ranging from 0 to 120 %

Every test for a given RTT, has been repeated for a given congestion control method and for a given number of nodes. We took great care of fine measurement precision: 0.5 s for *iperf*'s goodput, 20 ms for the GtrcNET-10's throughput and Web100's variables. Even though we specify *iperf* to perform *read()/write()* of 8 kB, we still observe burstiness in goodputs due to delay variation betweeen packets arrivals and *read()* returns, which explains why we sometimes observe goodput larger than link capacity.

To analyse all the data acquired, several metrics have been used to synthetically characterise the behaviour of different TCP variants. These metrics are:

- mean goodput: $\overline{g_i} = \frac{1}{T}\sum_{t=0}^{T} g_i(t)$
- aggregate goodput: $G(t) = \sum_{i=1}^{N} g_i(t)$
- standard deviation of goodput: $\sigma = \sqrt{\frac{1}{T}\sum_{t=0}^{T}(g_i(t) - \overline{g_i})^2}$
- goodput distribution: $\{p_{i,k} = p(\frac{k}{100} * 1\ Gbps \leq g_i(t) < \frac{k+1}{100} * 1\ Gbps) | k \in [\![0; 100[\![\}$
- fairness [JMW84]: $J = \frac{(\sum_{i=1}^{N} \overline{g_i})^2}{N(\sum_{i=1}^{N} \overline{g_i}^2)}$
- aggregate throughput: $X(t) = \sum_{i=1}^{N} x_i(t)$

where N is the number of nodes involved in the experiment (typically 12 in this report), T is the total duration of the experiment (typically 2800 s), $g_i(t)$ the i^{th} node's goodput over time t averaged on the iperf sampling interval, and $x_i(t)$ the i^{th} node's throughput over time t averaged on the GtrcNET-10 sampling interval.

We have now defined the methodology of our research paper by describing the traffic characterics, the scenarii that will be explored and the parameters and metrics that will help us to characterize the behaviour of TCP variants. The following section is dedicated to presenting the configurations that were used to perform our experiments.

3 Experiment description

3.1 System and service description

We used two similar experimental systems, composed of a classical dumbbell topology with twelve 1 Gbps source workstations connected to a 10 Gbps bottleneck link and twelve sink workstations on the other side as described in figure 2. In the first testbed (*testbed 1*) (Grid5000, France), the backbone of the Grid5000 platform is composed of a private 10 Gbps Ethernet over DWDM dumbbell with a bottleneck at 10 Gbps between Rennes and Nancy hubs (see figure 1). The average RTT is 11.5 ms that gives a bandwidth-delay product of 1.507 Mbytes for 1 Gbps connections.

The second testbed (*testbed 2*) (AIST-GtrcNET-10, Japan), is fully controlled. It is built around the GtrcNET-10p3 equipment that allows latency emulation, rate limitation and precise bandwidth measurements at 10 GbE wire speed. GtrcNET-10p3 is a fully programmable network testbed, which is a 10 Gbps successor of a well-established network testbed, GtrcNET-1 [KKT+04] for 1 Gbps Ethernet. GtrcNET-10p3 consists of a large-scale Field Programmable Gate Array (FPGA), three 10 Gbps Ethernet XENPAK ports, and three blocks of 1 GB DDR-SDRAM. The FPGA is a Xilinx XC2VP100, which includes three 10 Gbps Ethernet MAC and XAUI interfaces. By re-programming FPGA configuration, its functions are easily added and modified with keeping 10 GbE wire speed.

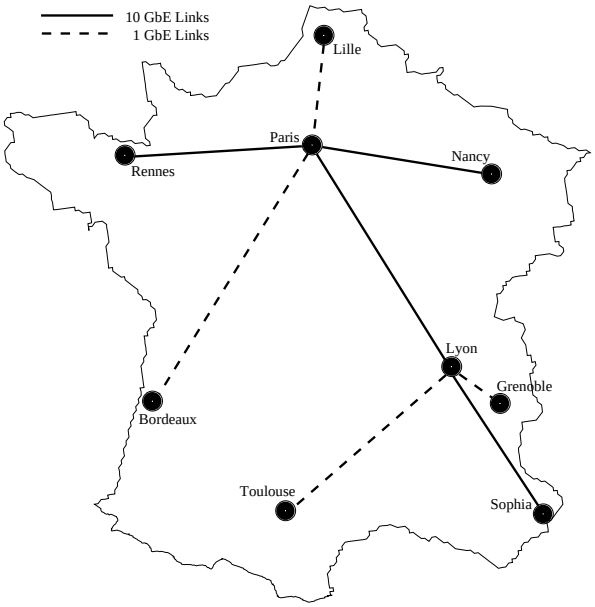


Figure 1: Grid5000 backbone

GtrcNET-10p3 provides many functions such as traffic monitoring in millisecond resolution, traffic shaping, traffic generation and WAN emulation at 10 GbE wire speed. In WAN emulation functions it adds latency, controls the transmission rate from port, generates random packet losses at the specified rate. The step of adding latency is 25.6 ns and maximum latency is 54 seconds. There is no packet loss if the latency is less than 858 ms. It also controls the transmission rate from 154 Mbps to 10 Gbps by changing IFG (Inter Frame Gap) in proportion to the frame length, so traffic is well paced.

In the *testbed 2*, nodes are interconnected by a layer 2 (Ethernet) switch[3], like on figure 2. As it shared by all the PCs in the testbed, separate VLANs were defined. All PCs also have a second Ethernet NIC unto which all the control traffic is sent so that there is no perturbation on the experiments' traffic. The output port of the switch acts as the bottleneck of the system.

In both testbed, the nodes were all IBM e-server 325 with 2 AMD64 Opteron 246 on which we deployed GNU/Linux 2.6.17 kernels patched with the Web100 [MHR03]. The Linux kernels were compiled with *HZ* set to 250. The NIC used were all using the *tg3* driver. Tests were performed using the *iperf*[4] utility. We used a large enough buffer size (50 MBytes for *testbed 2* and 4 MBytes for *testbed 1*) on both receiver and sender buffers of *iperf*[5] to provision for the various latencies we performed experiments with.

We also set the following kernel variables to tune the size of the TCP buffers:

```
net.core.rmem_max = 107374182
net.core.wmem_max = 107374182
net.core.rmem_default = 107374182
net.core.wmem_default = 107374182
net.core.optmem_max = 107374182
```

[3]Cisco Catalyst 4948 10GE

[4]version 2.0.2 compiled with gcc 3.2.2

[5]the value used is actually doubled by the Linux kernel

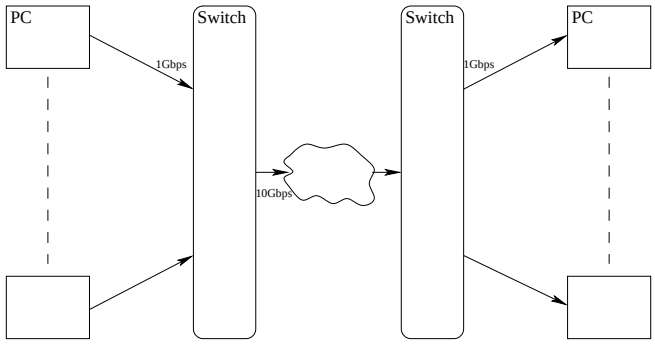


Figure 2: Topology of the experiments, the cloud represents either the Grid5000 or the AIST-GtrcNET-10 backbone

```
net.ipv4.tcp_rmem = 107374182 107374182 107374182
net.ipv4.tcp_wmem = 107374182 107374182 107374182
net.ipv4.tcp_mem = 107374182 107374182 107374182

net.core.netdev_max_backlog = 1000
net.ipv4.tcp_no_metrics_save = 1
```

The *txqueuelen* was set to 10000 for each network interface card (NIC) to prevent any packet losses in the *qdisc*, as it larger than the number of packets that we can send during a RTT period. The **mem* were set to these values accommodate the maximum intended value for the buffers requested by *iperf*. The *netdev_max_backlog* variable specifies the maximum length of the input queues for the processors. It has an impact on networking performance, as the buffer can only be flushed in a schedule slot. As we are using a Linux timer (*HZ*) of 250 and 1500 bytes' packet size, the max bandwidth would be about 375 MBps, which is large enough to use 1 Gbps NICs. The *tcp_no_metrics_save* variable specifies that the kernel isn't supposed to remember the TCP parameters corresponding to a network route and so ensures the independance of each successive experiment.

Please note that, even though we have tried to provide the best experimental environment possible, it appears that there was a bad interaction between the Base Board Management controller firmware of the nodes used at the AIST testbed with the firmware version of the NICs, which caused extra losses in the flows and downgraded the results we might have achieved with this testbed.

With this configuration and these two testbeds, we tried to provide the best situation to perform our tests, whose results will be detailled in the following section.

4 Results analysis

This section presents the results that were obtained during our experiments following the test plan given in section 2.2.

4.1 Single flows experiment with TCP variants

4.1.1 Experiment description

In this experiment, we evaluate different TCP variant protocols (HS-TCP [Flo03], H-TCP [SL04], Scalable TCP [Kel03], BIC TCP [XHR04], CUBIC [RX05]) with the same latency: 11.5 ms, both on the Grid5000 and the AIST-GtrcNET-10 testbed. This will help us in characterizing how each protocol is able to adjust for a given congestion level and how they are able to take advantage of the available resources.

For a given TCP variant and a given RTT, the first tests series were performed as follows:

- At time 0, we start the first couple of client-server.
- A *iperf* client is started 4 seconds after the corresponding *iperf* server to prevent overlap due to *ssh* connexion delay.
- Every $timer$, we start a new couple till all twelve nodes are started.
- As each *iperf* client is set to last $max\_duration - nb\_nodes\_already\_started * timer$, they gradually stop around time $max\_duration$.

The interval between each flow's start is important to avoid flows' interactions during their slow start phase. In this case, we make sure that flow interactions do not occur during any slow start phase by choosing a $timer$ value that is large enough, typically 200 s here. After more than 10 nodes have been started, we start to have congestion as we have reached the bottleneck's size.

The next section is dedicated to present the results of this experiment and compare them in the two testbeds used.

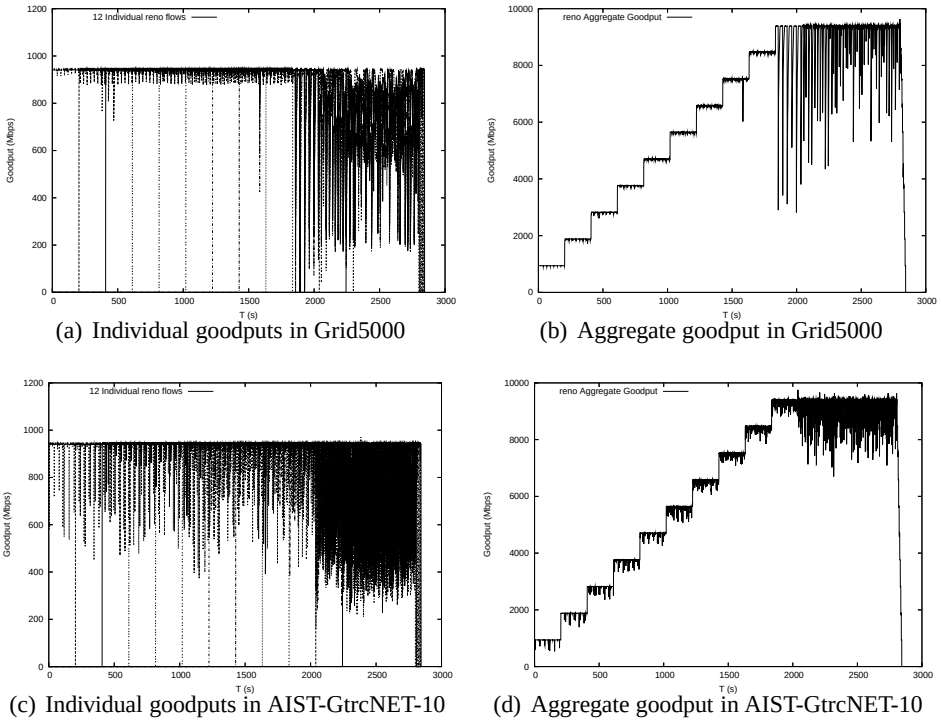
4.1.2 Results

For each protocol, the figures on the left shows all individual flow goodputs, while the corresponding figures at right gives the aggregate goodput value. The two figures on the top correspond to the tests performed in the Grid5000 testbed, while the ones on the bottom were done in the AIST-GtrcNET10 testbed. At this latency, all the protocols manage to fully use the network as the max aggregate throughput is close to 9843 Mbps when all the nodes are present.

All the figures are displaying sharp steps (as far as the 0.5 s mean provided by *iperf* allow us to see), except for Reno (figure 3), CUBIC (figure 5) and H-TCP (figure 7) protocols which are starting to display heavy perturbations, even though there is no congestion in the system yet, starting from the arrival of a fifth node. At the arrival of the tenth node, we start to observe a change in the behaviour of all the protocols even though we aren't over the nominal capacity of our network and we enter a state where the nodes aren't able to maintain a "stable" goodput, which might be caused by the "background noise" caused by other users and the intersite control traffic of the Grid5000 testbed.

We can also notice that some protocols have huge and quick individual variations in goodput such as BIC (figure 4) or HighSpeed (figure 6), which have an impact on the mean aggregate goodput.

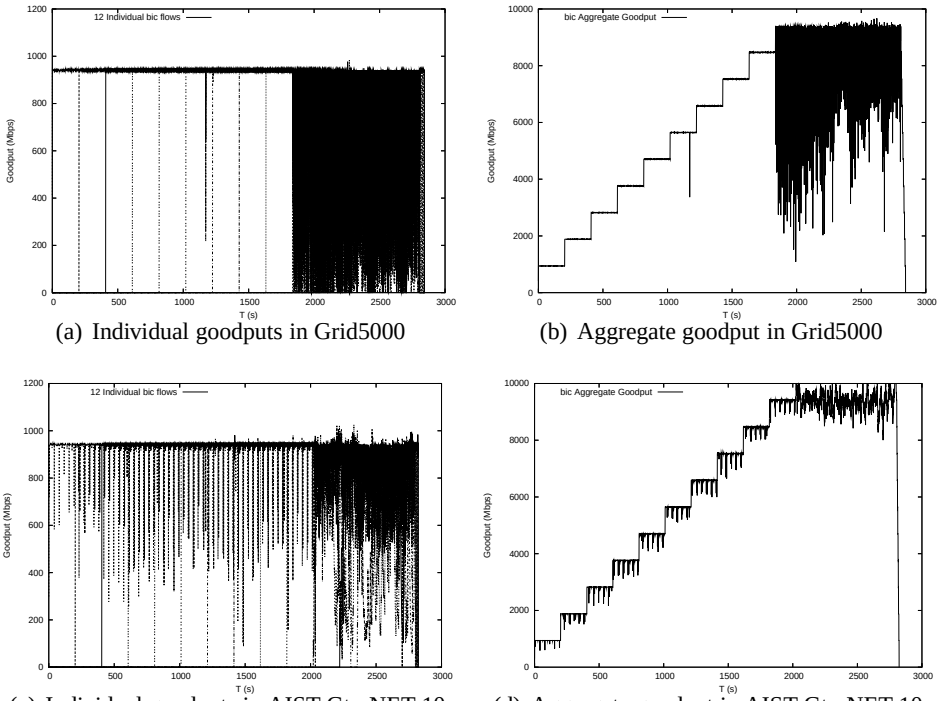
HighSpeed(figure 6) and Scalable (figure 8) are the only protocols in our benchmark in the AIST-GtrcNET-10 testbed that displayed a clear unfairness to the last nodes that are started when there is congestion in the system. For instance, the last node started in the AIST-GtrcNET-10 stays stuck at around 400Mbps during its whole presence in the system, when most of the others are emitting at 800Mbps. Due to the randomness added by control traffic and the agitated aspect of the Grid5000 figures, it is less noticeable in the other testbed.



(a) Individual goodputs in Grid5000 (b) Aggregate goodput in Grid5000

(c) Individual goodputs in AIST-GtrcNET-10 (d) Aggregate goodput in AIST-GtrcNET-10

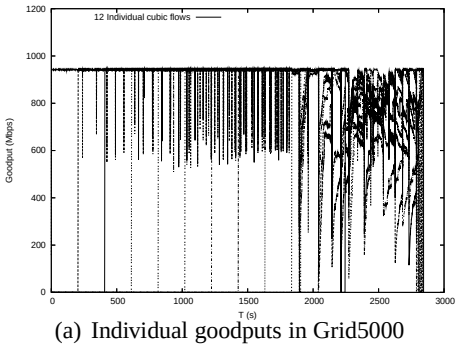
Figure 3: Reno, 11.5 ms RTT



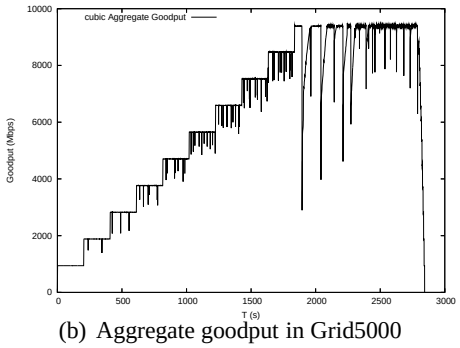
(a) Individual goodputs in Grid5000 (b) Aggregate goodput in Grid5000

(c) Individual goodputs in AIST-GtrcNET-10 (d) Aggregate goodput in AIST-GtrcNET-10

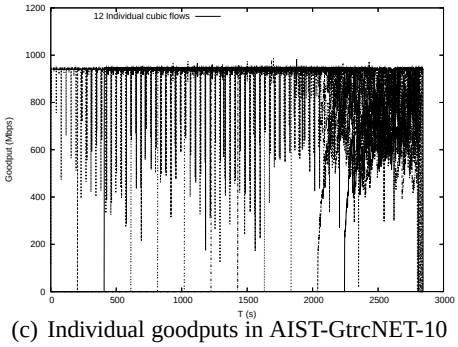
Figure 4: BIC, 11.5 ms RTT



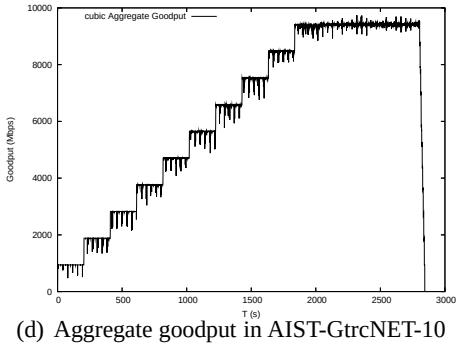
(a) Individual goodputs in Grid5000



(b) Aggregate goodput in Grid5000

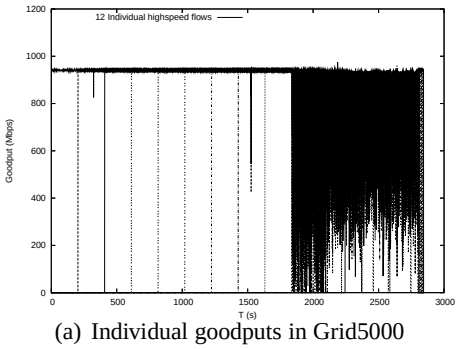


(c) Individual goodputs in AIST-GtrcNET-10

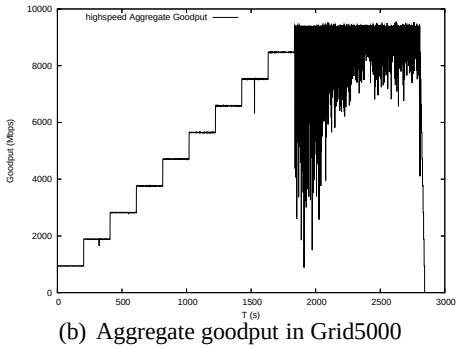


(d) Aggregate goodput in AIST-GtrcNET-10

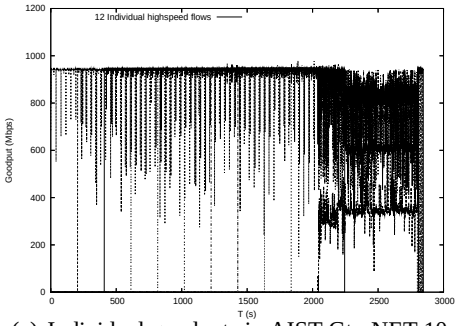
Figure 5: CUBIC, 11.5 ms RTT



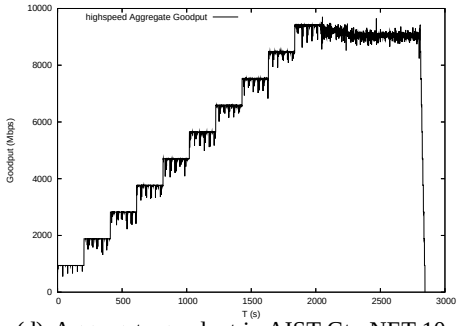
(a) Individual goodputs in Grid5000



(b) Aggregate goodput in Grid5000

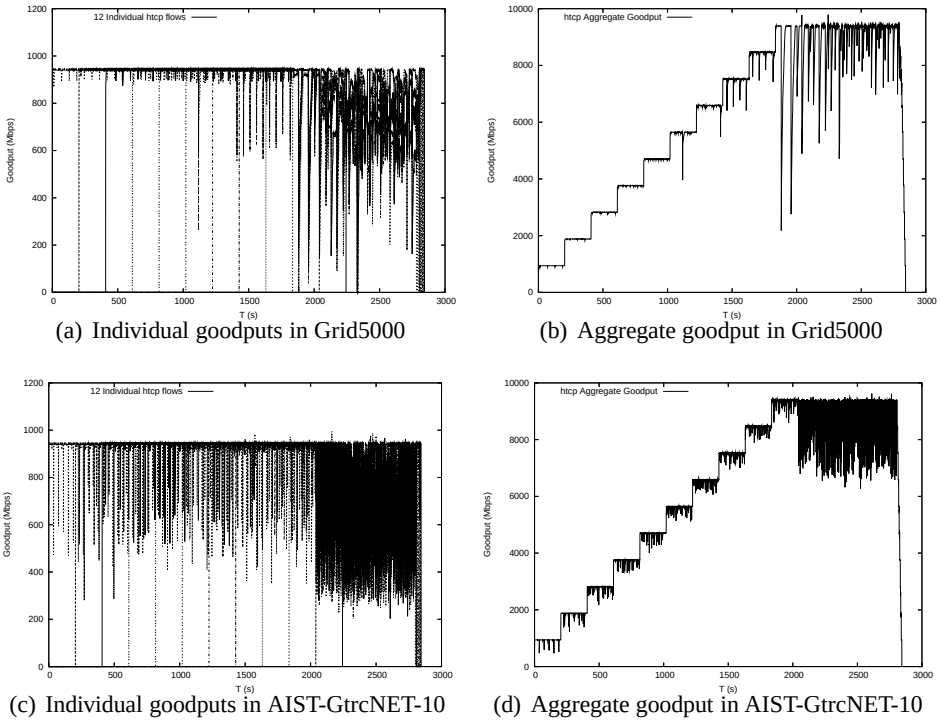


(c) Individual goodputs in AIST-GtrcNET-10



(d) Aggregate goodput in AIST-GtrcNET-10

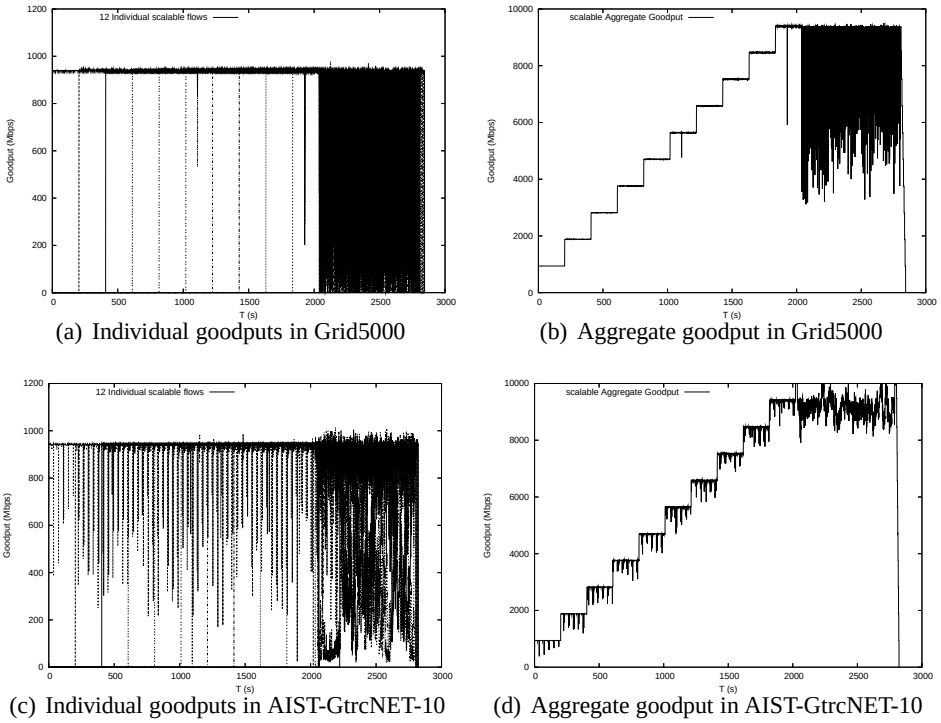
Figure 6: HighSpeed, 11.5 ms RTT



(a) Individual goodputs in Grid5000 (b) Aggregate goodput in Grid5000

(c) Individual goodputs in AIST-GtrcNET-10 (d) Aggregate goodput in AIST-GtrcNET-10

Figure 7: H-TCP, 11.5 ms RTT



(a) Individual goodputs in Grid5000 (b) Aggregate goodput in Grid5000

(c) Individual goodputs in AIST-GtrcNET-10 (d) Aggregate goodput in AIST-GtrcNET-10

Figure 8: Scalable, 11.5 ms RTT

One of the main differences between our two testbeds is that in Grid5000, we can experience perturbations coming from the control traffic (or even other users' traffic), which might account for the agitation that appears regularly in several figures. It can also explain the differences that were observed between the AIST-GtrcNET-10 and Grid5000 figures for CUBIC (figure 5), even though the Web100's logs report more retransmissions than expected in non-congested state. But these interferences aren't inevitably bad as the CUBIC figure (or the Reno figure) in the Grid5000 seems to yield better intra-flow fairness and display more stability as seen later in this report, on figure 17.

These results allowed us to observe that we are able to obtain very similar results with the two differents testbed, showing that an emulated testbed such as AIST-GtrcNET-10 can indeed help to have a acceptable environment/provides a good first approximation to simulate a real testbed with less infrastructure cost.

In the following sections, we will compare the behaviour of the TCP variants for different values of latencies to assert the impact of this parameter on the performance of flows.

4.2 Exploration of TCP variants behaviour in various latency conditions

We experiment the various TCP protocols by applying the same experimental procedure we used for Grid5000 in the AIST-GtrcNET-10 testbed. In this emulated testbed, we explore the impact of the latency on the protocols using this scenario.

4.2.1 Impact of the latency

First, we are going to verify the expected impact of an increasing latency on the various TCP variants we tested, which is a deterioration of the performances.

The figures were generated with the GtrcNET-10 logs for 11 ms and 100 ms RTT and so what is displayed in the figure 9 is the throughput measured after the bottleneck of the 10 Gbps link. From left to right, we present Reno, BIC, CUBIC, HighSpeed, H-TCP and Scalable TCP variants.

In our case, we can notice that the steps due to the addition of another couple of nodes get sloppier when we increase the latency. The effect is particulary noticeable on Reno (first column) and CUBIC (third column) as these protocols aren't able to fill the link. The deficiency observed for Reno is the well-known fact that Reno congestion control method isn't adapted to networks with high BDP product due to the slow evolution of the congestion windows in this condition.

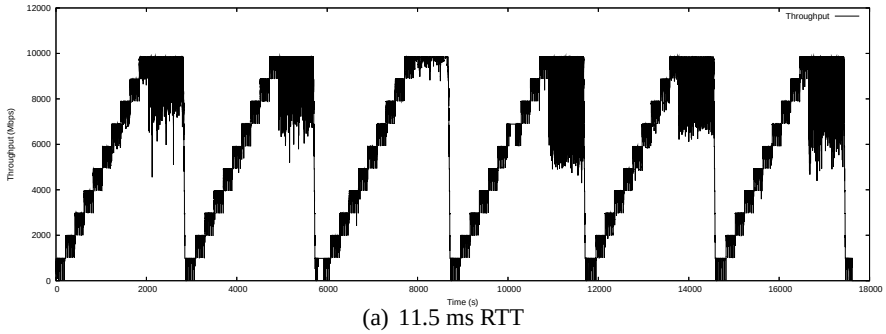
The fact that CUBIC seems to need more time than BIC to achieve a full utilization of the link, is also a consequence of the construction of this algorithm as stated in [RX05], because the growth of the congestion windows is slower when it gets closer to the upper target value, which might be a drawback if we want a TCP variant that is able to react fast to networking conditions and to get quickly advantage of available resources. CUBIC is less aggressive but is less efficient in such conditions.

4.2.2 Impact of the protocol

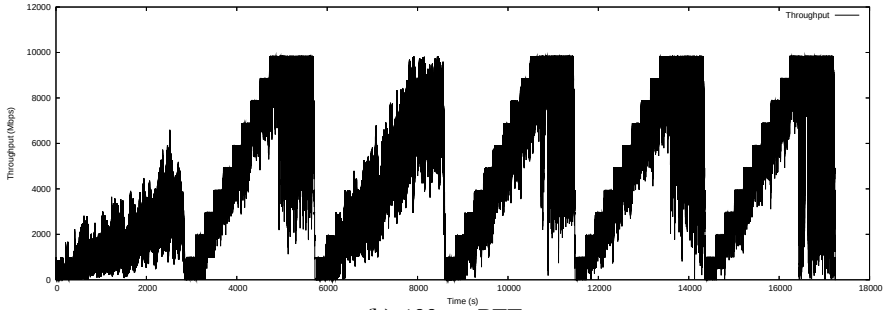
The figures in this section display the results we obtained for individual flows and for aggregate goodput as in the previous section. Figure 11 exhibits synchronisation effects, large number of flows are dropping packets at the same time, when there is congestion in the system. This problem might also explain some of the strange behaviours in the observed fairness on figure 17.

Figure 10 confirms the expected bad behaviour[6] of TCP Reno in large BDP environments as it is unable to achieve a goodput above 3.5 Gbps, when new TCP variants such as BIC (figure 11) are able

[6] as already stated in section 4.2.1

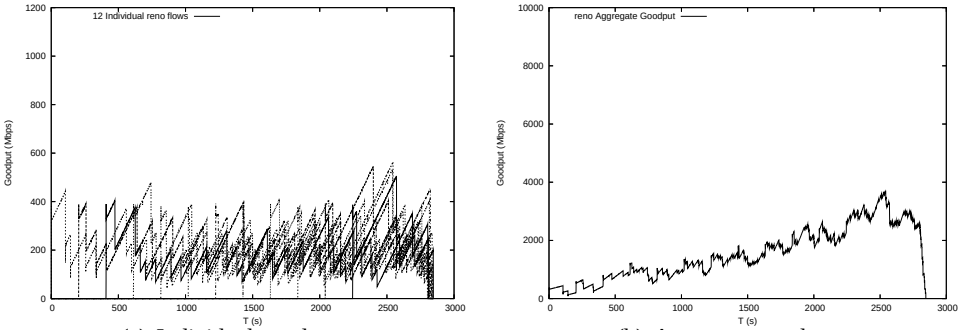


(a) 11.5 ms RTT



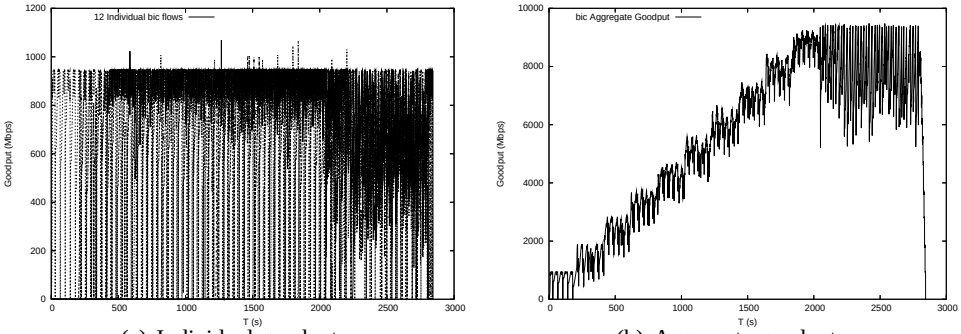
(b) 100 ms RTT

Figure 9: Reno, BIC, CUBIC, HighSpeed, H-TCP and Scalable with various RTT in AIST-GtrcNET-10



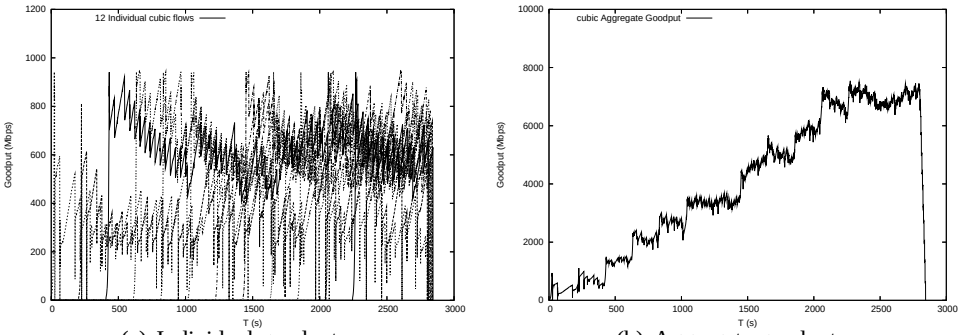
(a) Individual goodputs (b) Aggregate goodput

Figure 10: Reno in AIST-GtrcNET-10 with 100 ms RTT



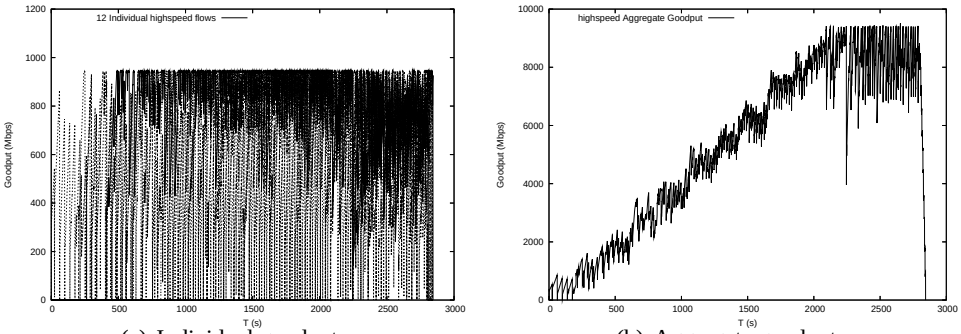
(a) Individual goodputs (b) Aggregate goodput

Figure 11: BIC in AIST-GtrcNET-10 with 100 ms RTT



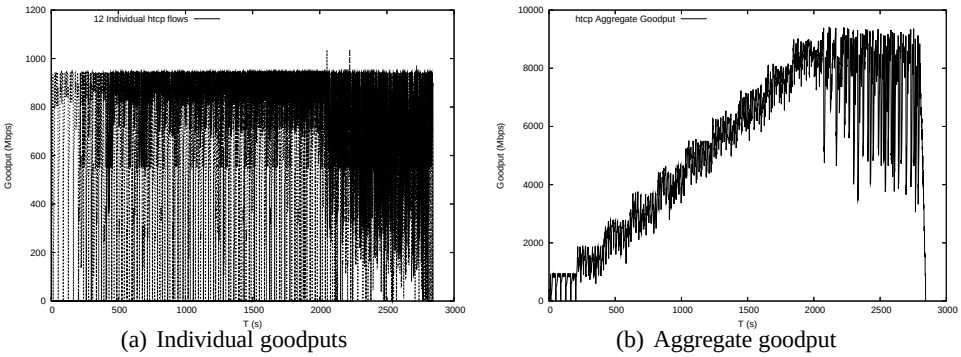
(a) Individual goodputs (b) Aggregate goodput

Figure 12: CUBIC in AIST-GtrcNET-10 with 100 ms RTT



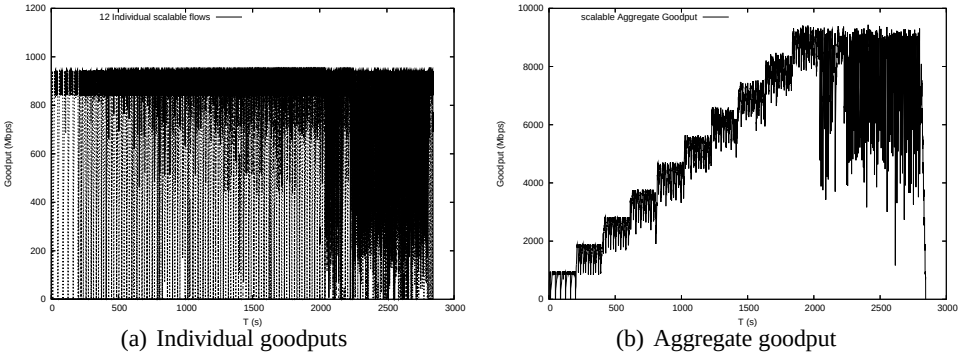
(a) Individual goodputs (b) Aggregate goodput

Figure 13: HighSpeed in AIST-GtrcNET-10 with 100 ms RTT



(a) Individual goodputs (b) Aggregate goodput

Figure 14: HTCP in AIST-GtrcNET-10 with 100 ms RTT



(a) Individual goodputs (b) Aggregate goodput

Figure 15: Scalable in AIST-GtrcNET-10 with 100 ms RTT

to achieve goodputs closer to the capacity of the link. We can clearly see the AIMD mechanism behind TCP Reno by looking at the individual goodputs, from the slow increase in congestion avoidance to the sharp decrease when a packet loss occurs.

Figure 12 presents the results for the CUBIC protocol, which just like Reno, isn't able to fill completely the link before the end of the experiment. We can see that all the nodes aren't able to stay around the maximum goodput achievable, even when there is no congestion in the system. It also seems that some of the nodes, for instance, the two first started nodes, had difficulties during their slow-start and were stuck around 300 Mbps for more than 400 s. We also notice that some of the nodes are very conservative, for instance the thrid node, as they slowly decrease the congestion window as other nodes enter the system, even when there is no actual congestion.

Figure 13 seems to have some difficulties during the earlier phase of the test as the two first nodes that were started aren't able to reach the maximum goodput achievable, which might indicate that HighSpeed require a certain amount of congestion or competition to start working normally. Figure 14 and figure 15 - respectively H-TCP and Scalable - are quite similar except that Scalable seems to be a bit more agitated and blurry. They still display, like BIC in figure 11, the same kind of steps than with lower RTTs, which shows that there are still able to react rather quickly to congestion level changes.

As already stated in section 3, please note that some of the bad behaviours observed in this section may have been caused by some firmware incomptabilities in the AIST-GtrcNET-10, which cause extra-losses in the nodes.

The work in this section showed that the different TCP variants start behaving differently as soon as they are used in high BDP conditions. It highlights the fact that one need to choose carefully its TCP variant according to the current RTT condition, as we will try to point in the following section.

4.2.3 Individual goodputs and fairness as a function of latency and protocol

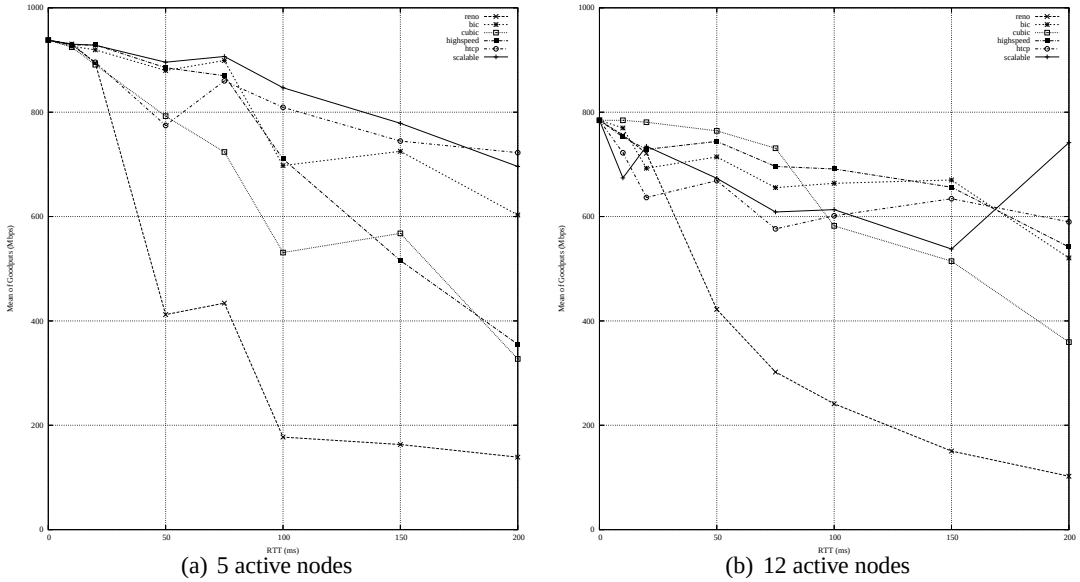
In this section, we tried to aggregate the data acquired from our previous experiments to represent them as function of latency and congestion control method. We used the three following metrics: mean of goodputs, Jain's index and standard deviation to the mean of goodputs. In the last part of this section, we also study the distribution of goodputs as it is a good way to synthesize the three other metrics.

The figures on the left side correspond to the case when only five nodes are emitting, while the ones on the right are the case with twelve nodes, that is to say without and with congestion.

Mean goodput Figure 16 presents the mean of the goodput means on the period of time where 5 or 12 nodes are active. We can see that RTT has an important effect on the goodput as it can cause a diminution of more than 50 % of the mean goodput. The global tendancy is a decrease of the mean goodput, even though there are a few exceptions such as H-TCP which seems to have somehow an erratic behavior around the 100 ms RTT value in the non-congestioned case or such as Scalable which seems to take back the lead if the RTT goes above 170 ms when 12 nodes are active.

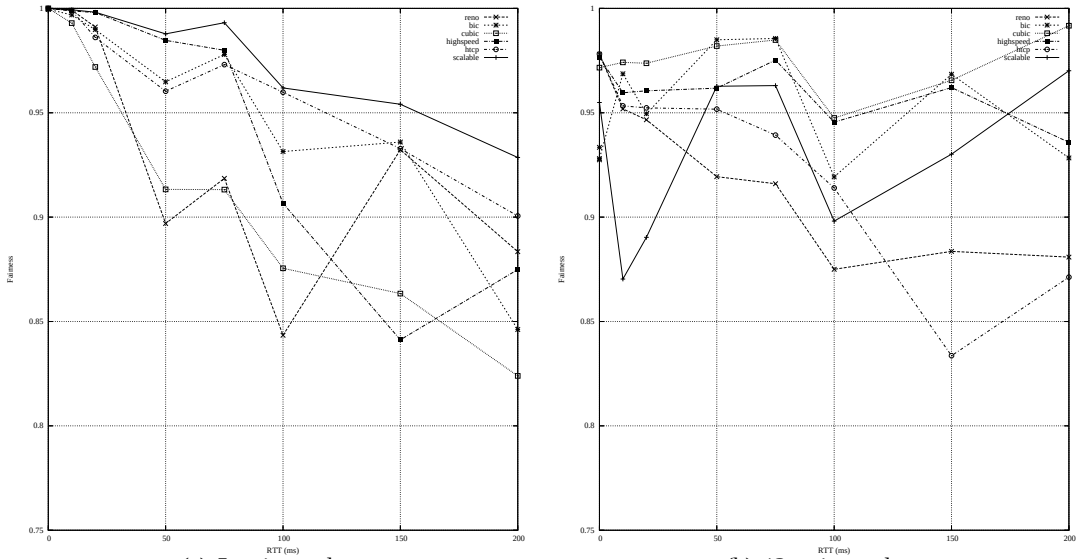
For small values of RTT, the protocols are yielding equivalent results but there are more discrepancies as the RTT increases as we experience more than 100 Mbps mean goodput difference between the different protocols. Most noteworthy, Reno is not able to keep up with up with the other TCP variants as soon as RTT is close to 30 ms.

If our criteria was to optimize the mean throughput achieved, figure 16 indicates that if we were in the non-congestioned case, a reasonnable pick would be to choose Scalable for every RTT lower than 170 ms and perhaps changing to H-TCP afterwards. In the other case, the logical choice would be to pick CUBIC for RTTs ranging from 0 ms to 80 ms, then switching to any TCP variants among



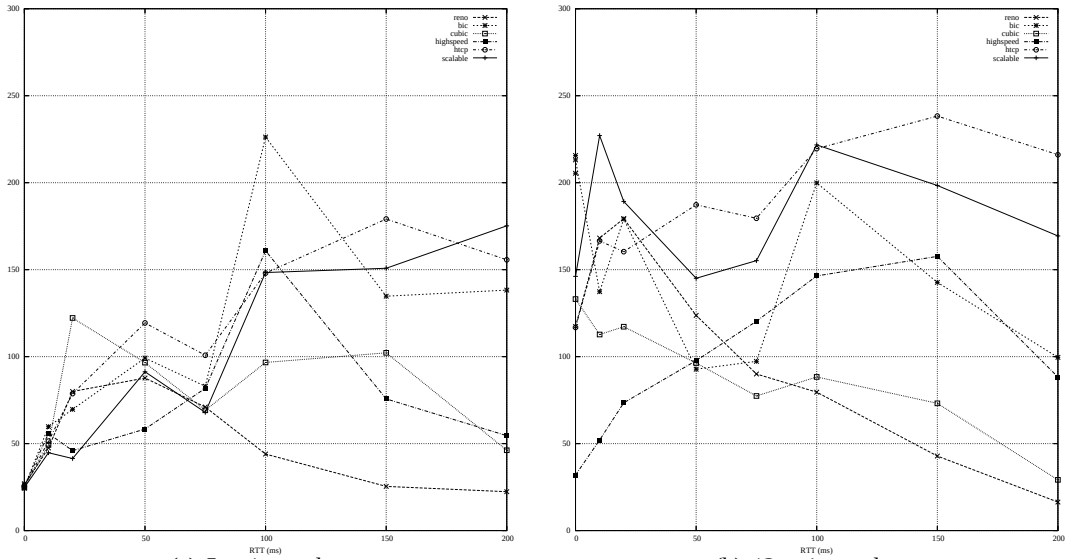
(a) 5 active nodes (b) 12 active nodes

Figure 16: Mean goodputs for TCP variants when 5 or 12 nodes are active in AIST-GtrcNET-10



(a) 5 active nodes (b) 12 active nodes

Figure 17: Fairness for TCP variants when 5 or 12 nodes are actives in AIST-GtrcNET-10



(a) 5 active nodes (b) 12 active nodes

Figure 18: Mean of goodputs' standard deviation for TCP variants when 5 or 12 nodes are actives in AIST-GtrcNET-10

BIC or HighSpeed as they have very similar results for the 80 ms-170 ms RTT values and finally use Scalable for RTT above 170 ms.

This information could help us determine the most adequate TCP variant for a given RTT value, if the mean goodput is the only criteria we have in mind in this particular configuration.

Fairness Figure 17 shows the evolution of the fairness for several TCP variants with respect to the RTT. In the non-congestionned case, the fairness is decreasing when the RTT is increasing, it is mainly due to the fact that the flows need more time to increase their throughput and that they tend to stay for rather long times around the same throughput value generating kind of stratas such as can be seen on figure 10. In the other case, except for a few exceptions (BIC and H-TCP), the fairness remains close to 0.95 in the congestioned case, which is a rather good value.

If our main criteria was to maximize the fairness, in the non-congestionned case, the best solution would be to use Scalable for every RTT values. In the other case, it seems that using CUBIC all along is the most reasonnable choice.

Standard deviation Figure 18 shows the level of variation of the goodputs around the mean goodput achieved, which is an indicator of the stability of a protocol in a given situation. Here we can notice that when there is no congestion and the RTT is low, all the procols are displaying similar standard deviations. For medium RTT values (20 ms to 75 ms), the behaviour is still very similar, even though the maximal difference between protocols reaches up to 100Mbps. It gets worse for some TCP variants for high RTT values as we can reach more than 200Mbps variation, even though some protocols such as BIC are behaving better when the RTT increases. This last point might be explained by the fact that with high RTT values, the fifth node is disadvantaged by its slow start period and requires more time to join when slow-evolving protocols such as Reno or CUBIC are used.

Please note that having a small standard deviation is perhaps not a good stand-alone criteria, as for

instance Reno is displaying very low standard deviation for high RTTs values, but at the same time, it is unable to achieve good mean goodput results.

When there is congestion, the behaviour is more erratic since BIC is showing more than 200Mbps of variation for low RTT values and is more stable (less thant 50Mbps) in high RTT conditions. It is quite hard to find similar patterns in their behavior, especially when the RTT is low, but it seems that we have two categories of protocols in high RTT conditions, those with low standard deviation (BIC and CUBIC) and those with high standard deviation (Reno, HighSpeed, H-TCP and Scalable)

Similar to our previous discussion with the mean goodput, if our main criteria were to minimize this metric, we could easily decide with figure 18 which protocol is most suited to a given RTT value. The reasonnable choice in the non-congestioned case would be to pick any protocol for RTTs lower than 20 ms, to take HighSpeed for the 20 ms-75 ms range and than switch to Reno for the rest of the time. In the congestioned case, a good solution would be to choose HighSpeed on the 0 ms-50 ms RTT range, CUBIC and finally Reno for the rest of the time.

Goodput distributions The figures 19, 20 and 21 correspond to the goodput distributions that were created by using the *iperf* logs on the period where all the 12 nodes were active.

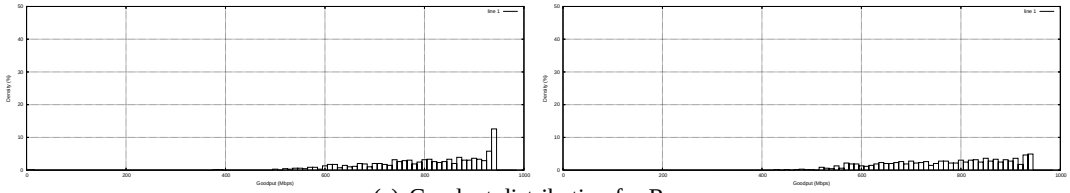
The analysis of goodput distributions shows two flow densities among the twelve. The figure 20 is representative of the different behaviors observed at 11.5 ms. We can see that the CUBIC distributions show an important mode close to the maximal goodput achievable (941 Mbps) for more than 30 % of the time, but there is an heavy tail. HighSpeed distributions look more like an Gaussian distribution, which shows that the HighSpeed goodput tends to be less variable than the one obtained with CUBIC.

When the RTT is low (figure 19), BIC and CUBIC tend to perform rather well as they are able to maintain a goodput close the maximum achievable (941 Mbps) for more than 40 % of the time, even though CUBIC shows a more important tail. The goodput distributions for HighSpeed look more like a Gaussian distribution with a peak. When the RTT is high (figure 21), the distributions are more widespread, failing to reveal a dominant mode, which suggest that all the protocols tested in this report doesn't perform well under high latency with congestion.

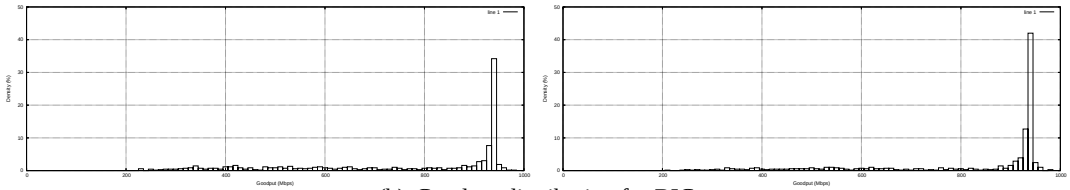
5 Conclusion

In this research report, we have explored the real behaviour of TCP variants in the context of grid-like high-speed networks. We have presented a few metrics that helped us characterize different variants of TCP in various RTT conditions and we have proposed a simple methodology that could be easily reproduced everywhere. This work also permitted to ensure that the AIST-GtrcNET-10 testbed, even tough we had some technical difficulties with firmwares, is a good approximation of a real testbed like the one we used in Grid5000 with interesting extra functionnalities like precise bandwidth measurement and latency emulation. Finally, we have provided a set of experimental measurements that allowed us to give a first insight of the performance of several TCP variants and to provide a few hints if we were to choose to use one of them in given conditions, according to the RTT or the congestion level, even tough for the moment, there is no universal solution.

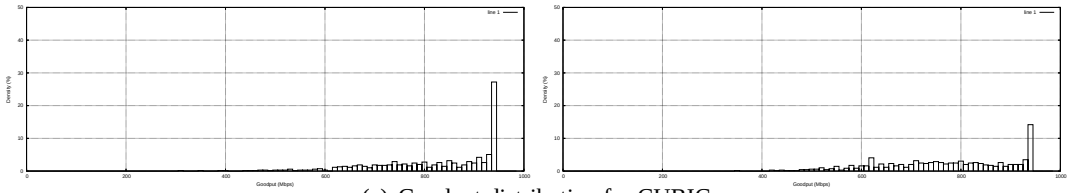
In the future, we plan to extend further our work by studying other aspects that could help us to improve and/or find the most suitable solution for a given set of networking conditions and objectives. To do so, we intend to perform the same kind of tests with parallel streams, which is considered to be an "effective" solution for bulk data transfers **??** and to conduct the study of the evolution of transfer time according to our parameters as well as the RTT fairness problem in the "grid context". Moreover, we will also need to check the impact of other parameters such as reverse and background traffics to be



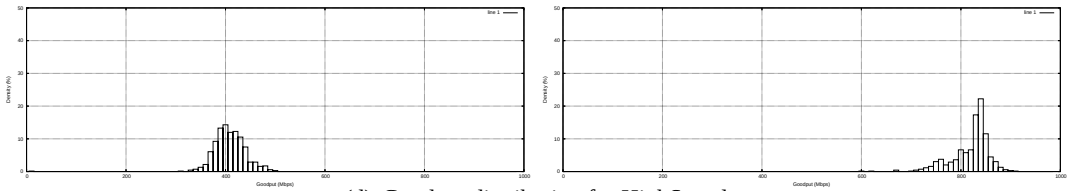
(a) Goodput distribution for Reno



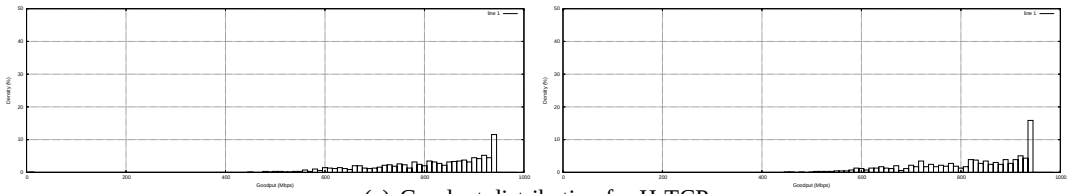
(b) Goodput distribution for BIC



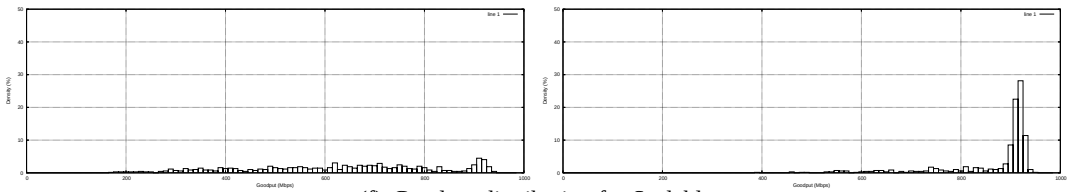
(c) Goodput distribution for CUBIC



(d) Goodput distribution for HighSpeed

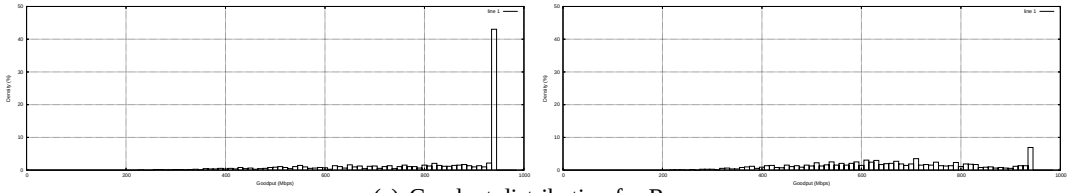


(e) Goodput distribution for H-TCP

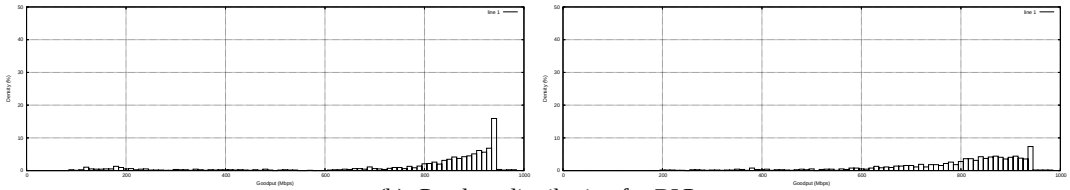


(f) Goodput distribution for Scalable

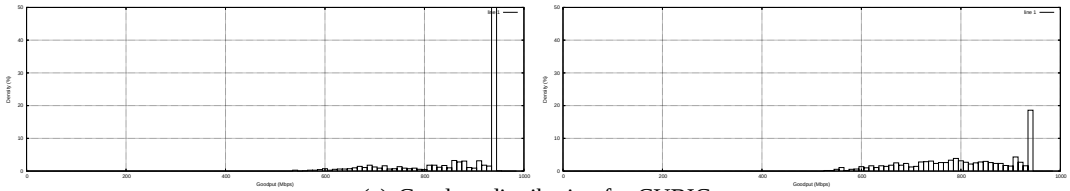
Figure 19: Examples of Goodput distribution for 0 ms RTT when 12 nodes are active, in AIST-GtrcNET-10



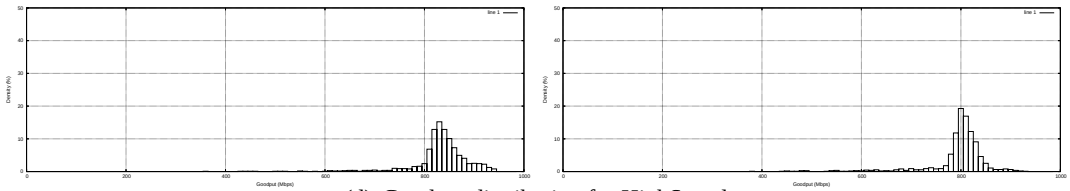
(a) Goodput distribution for Reno



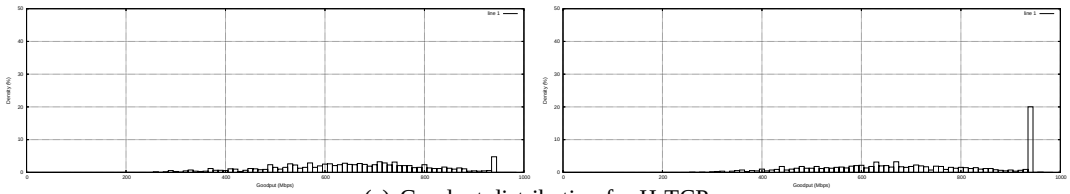
(b) Goodput distribution for BIC



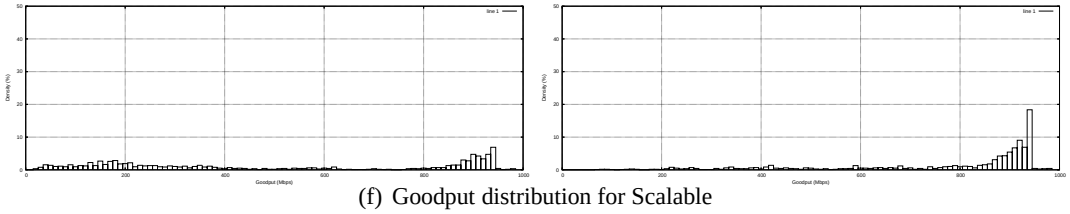
(c) Goodput distribution for CUBIC



(d) Goodput distribution for HighSpeed

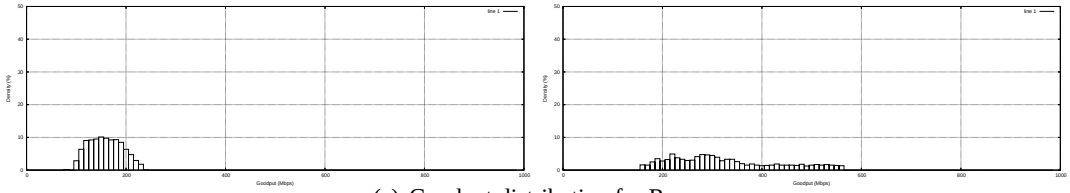


(e) Goodput distribution for H-TCP

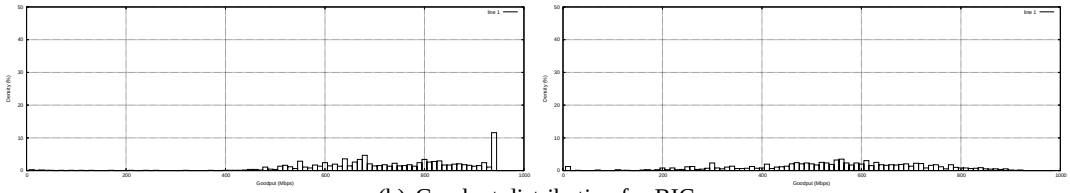


(f) Goodput distribution for Scalable

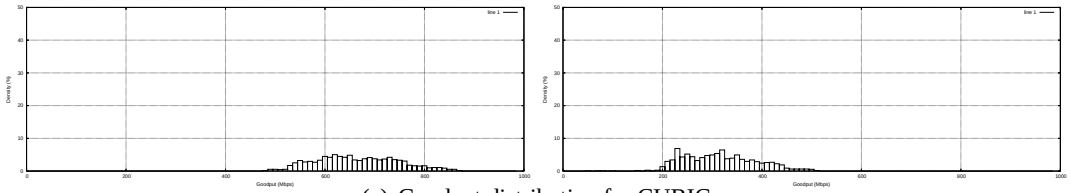
Figure 20: Examples of Goodput distribution for 11.5 ms RTT when 12 nodes are active, in AIST-GtrcNET-10



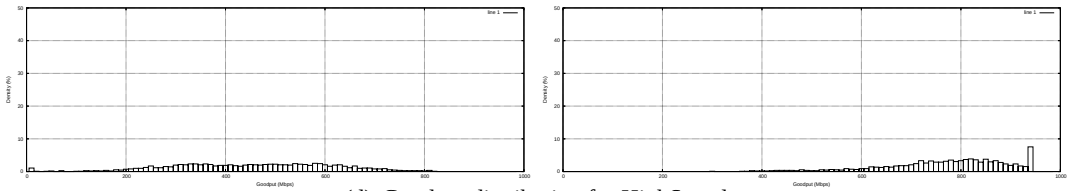
(a) Goodput distribution for Reno



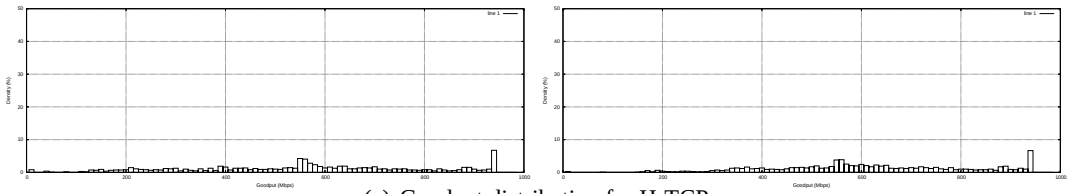
(b) Goodput distribution for BIC



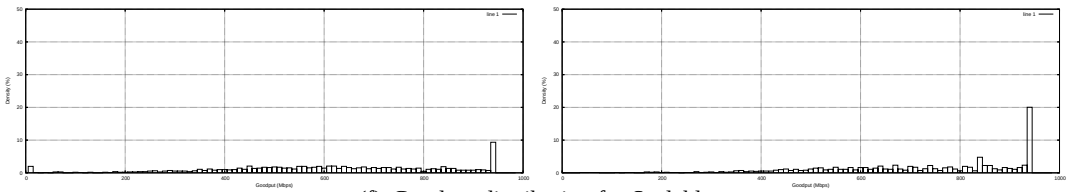
(c) Goodput distribution for CUBIC



(d) Goodput distribution for HighSpeed



(e) Goodput distribution for H-TCP



(f) Goodput distribution for Scalable

Figure 21: Examples of Goodput distribution for 100 ms RTT when 12 nodes are active, in AIST-GtrcNET-10

as close as possible to real networking conditions. Not to forget that we are currently only working on memory to memory transfers and that we will also need to tackle the problem of disk to disk transfers.

6 Acknowledgement

This work has been funded by the French ministry of Education and Research, INRIA, and CNRS, via ACI GRID's Grid5000 project and ACI MD's Data Grid Explorer project, the IGTMD ANR grant, Egide Sakura program, NEGST CNRS-JSP project. A part of this research was supported by a grant from the Ministry of Education, Sports, Culture, Science and Technology (MEXT) of Japan through the NAREGI (National Research Grid Initiative) Project and the PAI SAKURA 100000SF with AIST-GTRC.

References

[C+] R. Les Cottrell et al. Characterization and evaluation of tcp and udp-based transport on real networks. Presented at 3rd International Workshop on Protocols for Fast Long-distance Networks, Lyon, France, 3-4 Feb 2005.

[Ca05] Franck Cappello and al. Grid5000: A large scale, reconfigurable, controlable and monitorable grid platform. In *GRID2005 worskhop of the IEEE SuperComputing Conference*, November 2005.

[FFR+04] Ian Foster, Markus Fidler, Alain Roy, Volker Sander, and Linda Winkler. End-to-end quality of service for high-end applications. *Computer Communications*, 27(14):1375–1388, 2004.

[Flo03] S. Floyd. HighSpeed TCP for Large Congestion Windows. RFC 3649 (Experimental), December 2003.

[Flo06a] Metrics for the evaluation of congestion control mechanisms. In Sally Floyd, editor, *http://www.ietf.org/internet-drafts/draft-irtf-tmrg-metrics-04.txt*, August 2006.

[Flo06b] Tools for the evaluation of simulation and testbed scenarios. In Sally Floyd and E Kohler, editors, *http://www.ietf.org/internet-drafts/draft-irtf-tmrg-tools-02.txt*, June 2006.

[GHPS06] Romaric Guillier, Ludovic Hablot, Pascale Primet, and Sébastien Soudan. Evaluation of 10 gbe links in grid'5000. Technical report, LIP, ENS Lyon, 2006.

[GR06] Sergey Gorinsky and Nageswara S. V. Rao. Dedicated channels as an optimal network support for effective transfer of massive data. In *High-Speed Networking*, 2006.

[HLRX06] Sangtae Ha, Long Le, Injong Rhee, and Lisong Xu. A step toward realistic performance evaluation of high-speed tcp variants. *Elsevier Computer Networks (COMNET) Journal, Special issue on "Hot topics in transport protocols for very fast and very long distance networks" Pascale Vicat-Blanc, Joe Touch, Kasuchi Kobayashi Eds.*, 2006.

[JMW84] R. Jain, Chiu D. M., and Hawe W. A quantitative measure of fairness and discrimation for resource allocation in shared systems. Technical report, Digital Equipment Corporation, 1984.

[Kel03] Tom Kelly. Scalable tcp: improving performance in highspeed wide area networks. *SIGCOMM Comput. Commun. Rev.*, 33(2):83–91, 2003.

[KKT+04] Y. Kodama, T. Kudoh, T. Takano, H. Sato, O. Tatebe, and S. Sekiguchi. Gnet-1: Gigabit ethernet network testbed. In *In Proceedings of the IEEE International Conference Cluster 2004*, San Diego, California, USA, September 20-23 2004.

[LLS06] Yee-Ting Li, Douglas Leith, and Robert N. Shorten. Experimental evaluation of tcp protocols for high-speed networks. In *Transactions on Networking*, to appear 2006.

[MFVBP04] Jean Philippe Martin-Flatin and Pascale Vicat-Blanc Primet, editors. *High Performance Networks and Services for Grid : the IST DataTAG project experience*. Elsevier, dec 2004.

[MHR03] Matt Mathis, John Heffner, and Raghu Reddy. Web100: extended tcp instrumentation for research, education and diagnosis. *SIGCOMM Comput. Commun. Rev.*, 33(3):69–79, 2003.

[Rob04] J.W. Roberts. A survey on statistical bandwidth sharing. *Computer Networks*, Apr. 2004.

[RX05] Injong Rhee and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. In *Third International Workshop on Protocols for Fast Long-Distance Networks*, feb. 2005.

[SL04] R.N. Shorten and Doug Leith. H-TCP: TCP for high-speed and long-distance networks. In *Proceedings of 2nd International Workshop on Protocols for Fast Long-Distance Networks (PFLDnet'04)*, Argonne, Illinois USA, feb. 2004.

[WHVBPa05] Michael Weltz, Eric He, Pascale Vicat-Blanc Primet, and al. Survey of protocols other than tcp. Technical report, Open Grid Forum, April 2005. GFD 37.

[XHR04] Lisong Xu, Khaled Harfoush, and Injong Rhee. Binary increase congestion control for fast long-distance networks. In *INFOCOM*, 2004.

Contents