Can Realistic BitTorrent Experiments Be Performed on Clusters?

Ashwin Rao, Arnaud Legout, Walid Dabbous

Can Realistic BitTorrent Experiments Be Performed on Clusters?

Ashwin Rao†, Arnaud Legout, and Walid Dabbous
INRIA, France.
{ashwin.rao, arnaud.legout, walid.dabbous}@inria.fr

***Abstract*—Network latency and packet loss are considered to be an important requirement for realistic evaluation of Peer-to-Peer protocols. Dedicated clusters, such as Grid'5000, do not provide the variety of network latency and packet loss rates that can be found in the Internet. However, compared to the experiments performed on testbeds such as PlanetLab, the experiments performed on dedicated clusters are reproducible, as the computational resources are not shared. In this paper, we perform experiments to study the impact of network latency and packet loss on the time required to download a file using BitTorrent. In our experiments, we observe a less than 15% increase on the time required to download a file when we increase the round-trip time between any two peers, from 0 ms to 400 ms, and the packet loss rate, from 0% to 5%. Our main conclusion is that the underlying network latency and packet loss have a marginal impact on the time required to download a file using BitTorrent. Hence, dedicated clusters such as Grid'5000 can be safely used to perform realistic and reproducible BitTorrent experiments.**

***Index Terms*—BitTorrent, Experiment, Performance, Clusters, Latency, Loss Rate.**

I. Introduction

A rich diversity in network latency and packet loss rates have become essential for experimental evaluation of BitTorrent and other communication protocols used in the Internet. The need for such a diversity in network latency and packet loss rates is because of the heterogeneous nature of the Internet [1], [2], [3]. The heterogeneity of the Internet is the primary motivation for the creation of testbeds such as PlanetLab [4], [5]. However, due to the shared nature of the PlanetLab platform, the results of the experiments performed on PlanetLab are not reproducible [6]. In contrast, dedicated clusters, such as Grid'5000 [7], not only offer a reproducible environment but also enable the scaling of BitTorrent experiments by supporting a large number of BitTorrent instances on a single machine. The primary shortcoming of experiments performed on clusters is the absence of the diverse network latency and packet loss rates that can be found in the Internet. As *the impact of the network latency and packet loss on BitTorrent performance is not known*, there exists *a dilemma while selecting a testbed for the BitTorrent experiments*.

The BitTorrent protocol uses TCP to efficiently distribute the *pieces* of a file to a large number of peers using peer-to-peer (P2P) connections [8]. The peers contribute to the file distribution by uploading the pieces that have been downloaded. During the file download, the peers exchange control messages to select the pieces of the file to upload and the peers to whom these pieces are to be uploaded. BitTorrent also allows the users to limit the rate at which data is uploaded and downloaded. These rate limits allow the users to restrict the network bandwidth that BitTorrent can compete for during a BitTorrent session. As the users can control the upload process by limiting the upload rates, and BitTorrent uses control messages to decide the connections to upload to, BitTorrent is inherently different from other TCP based file transfer protocols such as HTTP and FTP.

Over the years BitTorrent performance has received considerable attention [9], [10], [11]. As these studies do not evaluate the interaction between TCP and BitTorrent, the impact of network latency and packet loss rates on BitTorrent performance is not known. Network latency and packet loss introduce a *ramp-up* period which is required by TCP to attain the maximum upload rate that can be achieved [12], [13]. BitTorrent users that limit the upload rate can therefore limit the impact of TCP ramp-up. Apart from the TCP throughput, the BitTorrent performance is also dependent on the control messages exchanged by the peers. The control messages generated by a peer can generate a delayed response at a remote peer because of the network latency and the packet losses. The delayed response can affect the various algorithms used by BitTorrent, such as the peer selection algorithm which is used to decide the peer to upload to. Due to the above reasons, the analytical models for TCP cannot be directly used to provide the impact of network latency and packet loss on BitTorrent performance.

In this paper, we use the download completion time, i.e, the time required to download a file using BitTorrent, as a metric to study the impact of network latency and packet loss. In our experiments, we observe that network latency and packet loss have a marginal impact (less than 15%) on the download completion time of a file. We first study the impact of network latency without packet loss. Network latency causes not only delays in receiving control messages but also TCP ramp-up. We therefore study the impact of network latency by studying the impact of the delays in receiving control messages and TCP ramp-up on the download completion time. We observe that the download completion time, when the round-trip time (RTT) between *any two peers* in the torrent is 1000 ms, is not more than 15% of the download completion time when the RTT is 0 ms.

†This is the author version of the paper published in the Proceedings of the 10^{th} IEEE International Conference on Peer-to-Peer Computing (IEEE P2P'10) in Delft, Netherlands, on August 25-27, 2010.

We then study the impact of network latency with packet loss. We observe that an RTT of 400 ms *between any two peers* and a packet loss rate of 5% does not increase the download completion time by more than 15% of the download completion time observed when the RTT is 0 ms and the packet loss rate is 0%. We also observe that an RTT of 1000 ms *between any two peers* with a packet loss rate of 5% can increase the download completion time by more than 15% of the download completion time observed when the RTT is 0 ms and the packet loss rate is 0%. As an RTT greater than 400 ms between *any two peers* is unrealistic, our results show that for upload rates typically seen in the Internet, the download completion time is not sensitive to the network conditions that can be found in the Internet; *dedicated clusters, such as Grid'5000, can be used to perform BitTorrent experiments.*

The remainder of this paper is structured as follows. The network topologies and the technique used to emulate the latency and packet loss are presented in Section II. We first present the impact of latency without packet loss on download completion time. In Section III, we emulate the same latency between any pair of peers in the torrent. We use this topology to study the impact of TCP ramp-up and the impact of the delays in receiving the BitTorrent control messages on the download completion time. In Section IV, we emulate torrents to study the impact of network latency when the condition of same network latency between any two peers is relaxed. We then study the impact of network latency and packet loss in Section V. We finally conclude in Section VI.

II. Methodology

In this paper, we use the terminology used by the BitTorrent community. A *torrent*, also known as a BitTorrent session or a swarm, consists of a set of peers that are interested in having a copy of the given file. A peer in a torrent can be in two states: the *leecher* state when it is downloading the file, and the *seed* state when it has a copy of the file being distributed. The peers distribute the file in chunks called *pieces*; a piece is further split into *blocks* to facilitate the piece upload and download. A peer is said to *unchoke* a remote peer if it is uploading the blocks of a piece. A *tracker* is a server that keeps track of the peers present in the torrent. For our experiments, we use a private torrent with one tracker, one initial seed (henceforth called the seed), and 300 leechers. We use the *download completion time*, the time required by the leechers to download the file distributed using BitTorrent, as the metric to study the impact of network latency and packet loss.

All the experiments were performed using an instrumented version of the *BitTorrent mainline client* [14] on machines running Linux as the host operating system. The BitTorrent mainline client internally uses TCP. We used TCP Cubic [15], the default TCP implementation for the current series of the Linux kernel, for our experiments. The latest version of uTorrent, a BitTorrent client, is based on uTP which uses UDP as the transport layer protocol [16], [17]. As in the case of TCP, uTP has a window based congestion control mechanism. The design of uTP also ensures that uTP ramp-up is not faster than TCP [16]. The control messages generated by uTorrent using uTP are similar to those that are present in BitTorrent clients that use TCP. Hence, we believe that the results presented in this paper are valid for BitTorrent clients using uTP.

In this section, we first present the various scenarios used in the experiments. We then present the procedure to emulate the network latency and packet loss. This is followed by an overview of Grid'5000, the experimental platform we used. Finally, we present the parameters used while performing the experiments.

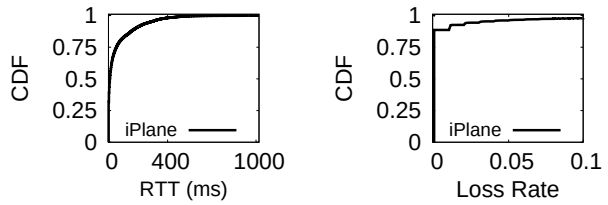
A. Experiment Scenarios

We now enumerate the scenarios used to study the impact of network latency and packet loss on the download completion time of a file.

1) *Scenario of Homogeneous Latency.* In this scenario, we emulate *a fixed network latency between any two peers* in the torrent. The fixed network latency, though unrealistic, provides a controlled environment to study the impact of network latency on the download completion time of a file. We use this scenario to get an insight on the threshold of network latency beyond which the network latency affects the download completion time. The scenario also gives the download completion time when the maximum latency between any two peers in a torrent in known. The results of this study can also be used to give the impact of network latency when the hosts are geographically distributed and are connected with links that have negligible packet loss.
2) *Scenario of Heterogeneous Latency.* In this scenario we relax the condition of fixed network latency to emulate a more realistic network topology. We use this scenario to confirm that the results obtained in the scenario of homogeneous latency are valid even when the condition of fixed network latency between any two peers is relaxed. For this scenario, we group peers that have the same network latency among themselves in an emulated Autonomous System (AS). We assume these ASes to be fully meshed and that the inter-AS latency is greater than the intra-AS latency; we assume the network latency between the peers in a given AS is the same.

BitTorrent allows its users to limit the upload and download rate. These rate limits restrict the network bandwidth that BitTorrent can compete for with other applications such as Web browsers. In this paper, we do not place any restrictions on the download rate. We set the upload rate limit of the peers from a wide range of values, from 10 kB/s to 100 kB/s. We use this range of upload rates as Choffnes *et al.* [3] show that 90% of the hosts present in public torrents upload at rates that are smaller than 100 kB/s. In our experiments, we assume the same upload rate limit at all the leechers while studying the impact of network latency and packet loss. We use the following torrent configurations to set the limit on the upload rate of the peers.

1) *Slow Seed and Slow Leechers.* In this scenario, we assume all the peers in the torrent have a low upload



(a) Network latency observed in the Internet.

(b) Loss rate observed in the Internet.

Fig. 1: iPlane measurements of network latency and loss rate observed in the Internet. *98% of the links have an RTT less than 400 ms and 99.8% of the links have an RTT less than 1000 ms. 95% of the links have a loss rate less than 0.05, i.e., 5%.*

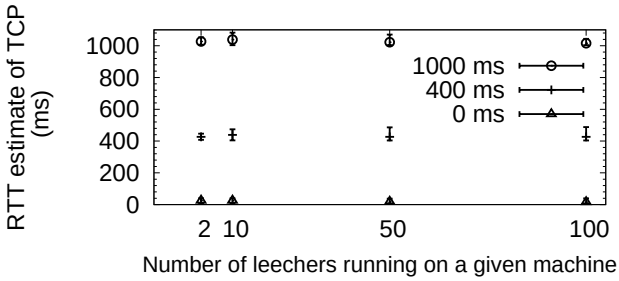


Fig. 2: Impact of the number of leechers running on a machine on the RTT estimate of TCP. *The machines used can be used to run up to 100 leechers.*

rate. We also assume the same upload rate at the seed and leechers. We performed two experiments for this scenario; we limit the upload to 10 kB/s for the first experiment and 20 kB/s for the second experiment.

2) *Fast Seed and Slow Leechers.* Some torrents have seeds that upload faster than the leechers. Public torrents are also known to have leechers that are capable of high upload and download rates. In torrents where the seed favors fast leechers [18], these leechers are able to download the file faster than their slower peers. As these leechers are capable of downloading pieces faster than their slower peers, they act like a fast seed to the slow peers in their peer-set. We emulate torrents that have a fast seed by limiting the upload rate of the seed to 50 kB/s and the upload rate of the leechers to 20 kB/s.
3) *Fast Seed and Fast Leechers.* We perform these experiments to emulate torrents where all the peers are capable of high upload rates. We performed two experiments for this scenario; we limit the upload to 50 kB/s for the first experiment and 100 kB/s for the second experiment.

B. Emulation of Network Latency and Packet Loss

We use the Network Emulation (NetEm) module for the Linux kernel [19] to emulate the network latency and packet loss between the peers. The NetEm module operates between the TCP/IP implementation and the device driver for the network device. In our experiments, the peers used the ethernet and the loopback device to communicate with each other. The NetEm module emulates network latency by en-queuing the packets at the ingress interface and the egress interface of a network device; packets losses are emulated by dropping the packets at the ingress and egress interfaces. When the loss is on the egress interface, due to cross-layer optimizations performed by the TCP implementation in Linux, TCP retransmits the packet without considering it as a loss in the network. To avoid such retransmissions, we introduce packet losses on the ingress interface of the network device.

A shortcoming of NetEm is that the network latency and packet losses are *inherently* bound to the network device. NetEm can be used to emulate network latency and losses for each connection, however such topologies are hard to verify and manage. In this paper we set the network latency and packet loss rates for a particular network device. The network latency between a pair of peers is the sum of the network latency at the devices used; the loss rate seen by a packet is the loss rate at the ingress interface of the destination.

We use the publicly available iPlane measurements [20], [21] to obtain the values of the network latency and loss rate to emulate. Figure 1 shows the distribution of RTT and loss rate observed in 5 iPlane samples; each sample contains the measurement of about $6 * 10^5$ links. In Figure 1a, we observe that 98% of the links have an RTT less than 400 ms, and 99.8% of the links have an RTT less than 1000 ms. We use these results to emulate a wide range of RTT values, from 0 ms to 1000 ms, between a pair of peers in the torrent. In Figure 1b, we observe that 95% of the links probed have a loss rate less than 0.05, i.e, 5% packet loss. We emulate a 5% packet loss on the ingress of each interface of the machines while studying the impact of packet losses.

C. Experiment Setup

We performed our experiments on the Grid'5000 experimental testbed [7]. Grid'5000 consists of a grid of clusters that are geographically distributed across France. Each cluster consists of several machines connected by a dedicated and high speed LAN; these clusters are inter-connected by high speed links. We performed our experiments on *a single cluster of Grid'5000*. In this cluster, we observe an RTT of less than 1 ms between a pair of machines when network latency is not emulated. Due to the very small network latency and the absence of packet loss between the machines in the LAN, the cluster does not reflect the network conditions present in the Internet. Unlike other testbeds such as PlanetLab, the machines in the cluster are not shared. Grid'5000 thus provides a reliable and robust platform for performing reproducible experiments.

We scale our experiments by running 100 leechers on a single machine of the cluster. We performed the following test to ensure that the machines can support up to 100 leechers while emulating the desired network latency. We distribute a 50 MB file in a private torrent with a single seed and a single tracker. The seed and tracker ran on one of the machines of the cluster. We used another machine in the cluster to run the leechers. We limit the upload rate of each of the peers to

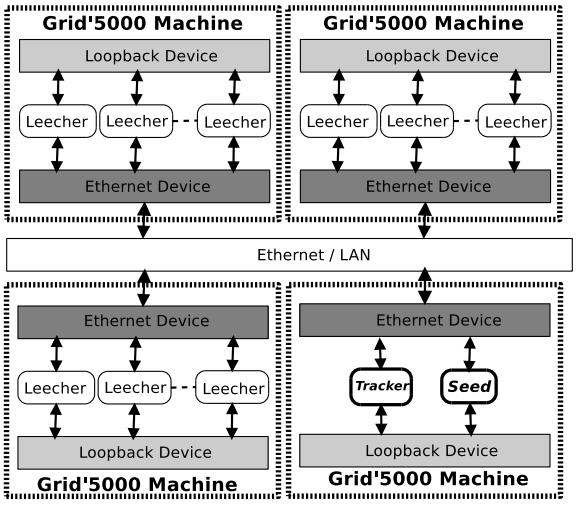


Fig. 3: Experiment setup. *One machine for the tracker and the initial seed, and three machines each with 100 leechers.*

100 kB/s and we vary the number of leechers running on a machine from 2 to 100. To study the impact of the number of leechers, we monitor the RTT estimate of TCP in the following manner. All the peers in the torrent use the socket interface of TCP to communicate with each other. The `send` method of this interface is used by the peers to send data to other peers in the torrent. On each call of `send`, we sample the RTT estimate of TCP by using the `TCP_INFO` option of the `getsockopt` method. Figure 2 shows the average RTT estimate of TCP when we emulate an RTT of 0 ms, 400 ms, and 1000 ms, between the peers; the error bars represent the minimum and maximum RTT estimate observed in five iterations. We observe that the number of leechers running on a given machine has a less than 15% impact on the average RTT estimate of TCP.

The NetEm module buffers the TCP frames which are in flight for a time period equal to the RTT being emulated. An RTT of 1 second (1000 ms) for an upload rate of 100 kB/s would require 100 frames of 1 kB to be in flight. For our experiments, we use a buffer size of 10^5 frames to support 1000 frames of each of the 100 peers to be in flight.

We perform our experiments in a private torrent with one tracker, one initial seed (henceforth called the seed), and a flash crowd of 300 leechers. We distribute a 50 MB file in this torrent. For our experiments, we assume that the peers remain in the torrent until all the leechers have finished downloading the file. As shown in Figure 3, we use one machine for the tracker and the seed. We use three machines for the 300 leechers; each machine runs 100 instances of the leechers. A pair of peers in the torrent use either the loopback interface or the ethernet interface to communicate with each other.

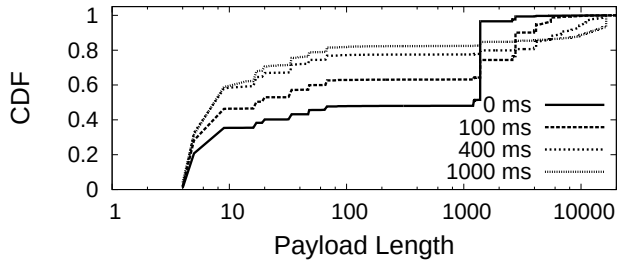
D. Impact of TCP Segmentation Offloading

The Maximum Segment Size (MSS) of TCP specifies the maximum payload length that can be exchanged over a connection [12]. One factor that contributes to the MSS is the Maximum Transmission Unit which is typically 1500 bytes for devices used in LANs. The MTU is set to 16436 bytes for the loopback interface on many Linux distributions. To ensure that the payload length exchanged by the peers does not depend the interface used by the connection, we set the MTU on the loopback interface to 1500 bytes. Despite this limit, we observed that a significant number of TCP segments have a size larger than the MSS negotiated during connection establishment. We observe these large segments because of TCP Segmentation Offloading (TSO) which is enabled by default in the 2.6 series (the current series) of the Linux kernel.

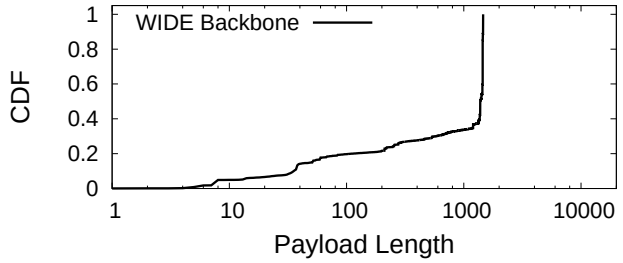
TCP Segmentation Offloading (TSO) enables the host machine to offload some of the TCP/IP implementation, such as segmentation, and calculation of IP checksum, to the network device. TSO also supports the exchange of data in frames of sizes that can be greater that the underlying MTU size [22]. The increase in the frame size can result in significant improvement in throughput; the improvement depends on various factors such as CPU processing power and the amount of data being transferred [23]. Clusters, such as those present in Grid'5000, include hosts that support TSO.

We now study the impact of RTT on the TCP payload length when TSO is enabled. We use four machines to create a private torrent with 300 leechers, one tracker, and one initial seed, to distribute a 50 MB file. We assume the same RTT between any two peers for the results presented in Figure 4a and Figure 4c. We also limit the upload rate to 20 kB/s. Figure 4a shows the distribution of payload length for different RTT values when TSO is enabled; the results present the distribution of the payload lengths observed in 5 iterations. We observe that an increase in the network latency between the peers results in an increase in the number of TCP segments with a large payload length. This was observed when the MTU was set to 1500 bytes indicating that TSO can result in payload lengths greater than the MTU value specified. Figure 4b shows the payload lengths from the publicly available traces of the Internet traffic observed in the WIDE backbone [24]; the values presented are from the sample taken on the WIDE backbone on November 29, 2009. In Figure 4a and Figure 4b, we observe that the maximum payload length of the packets sent over the Internet is smaller than the maximum payload length observed in Grid'5000 when TSO is enabled. We observe small payload lengths in Figure 4b because hardware support on all the intermediate devices is essential for exchange of large segments. When TSO is disabled and the MTU is set to 1500 bytes, Figure 4c shows the distribution of the payload length for different RTT values; the results present the distribution of the payload lengths observed in 5 iterations. We observe that when TSO is disabled and the MTU is set to 1500 bytes, the maximum payload length is similar to that observed in the Internet.

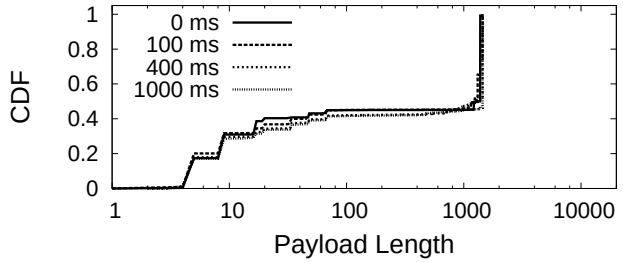
We set the MTU to 1500 bytes on the loopback interfaces and disabled TSO on the ethernet and loopback interface for the experiments presented in the Section III and Section IV. The outcome of the experiments with TSO enabled are discussed with the impact of packet loss in Section V.



(a) Payload length with TSO Enabled. Increasing the RTT causes an increase in payload length.



(b) Payload lengths observed in the WIDE backbone. Maximum payload length less than 1500 bytes as TSO requires hardware support on all the links on a connection.



(c) Payload length with TSO Disabled and MTU set to 1500 bytes.

Fig. 4: Impact of TSO on TCP payload length. *Setting the MTU to 1500 bytes and disabling TSO ensures that the maximum payload length is similar to that observed in the Internet.*

III. Homogeneous Latency

In this section, we assume the same network latency between any two peers in the torrent because *this scenario gives the worst case impact of a given network latency for a given upload rate limit*. The same network latency results in the same RTT between any two peers in the torrent because the sum of the other delays, such as queuing delays at the intermediate routers, is less than 1 ms in a Grid'5000 cluster. In each experiment, we choose an RTT to emulate from a wide range of values, from 0 ms to 1000 ms. In Section II-B, we observed that 99.8% of the Internet links have an RTT less than 1000 ms. *An RTT of 1000 ms between any two peers thus gives a worst case impact of the network latency that can be observed in the Internet*. We use a wide range of values, from 10 kB/s to 100 kB/s, to set the upload rate limit of the peers in the torrent; we assume the same upload rate limit for all the leechers in the torrent. For a given upload rate limit, Figure 5 shows the impact of the RTT between any two peers on the download completion time. Each point

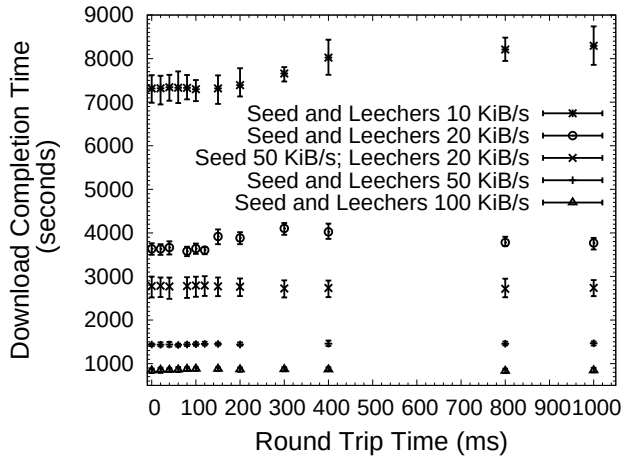


Fig. 5: Impact of latency on the average download completion time. The error bars indicate the minimum and maximum download completion time. *The latency increases the download completion time by at most* 15%.

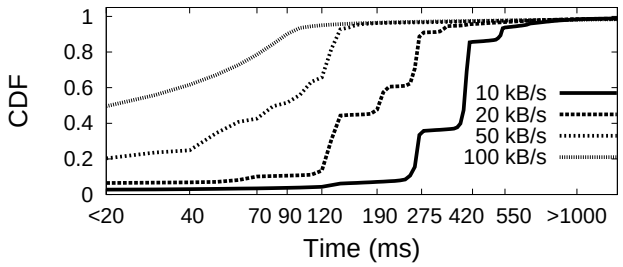
on the plot is the average download completion time in 10 iterations; the error bars indicate the minimum and maximum download completion time observed in 10 iterations. The two factors that can be affected by network latency and cause an increase in the download completion time are TCP ramp-up and the delays in receiving the control messages. We study the impact of network latency on the download completion time by studying the impact of network latency on these two factors in Section III-A and Section III-B respectively.

In Figure 5, for a given upload rate limit from 10 kB/s to 100 kB/s, we observe that an RTT of up to 1000 ms between any two peers does not increase the average download completion time by more than 15% of the average download completion time when the RTT is 0 ms. When the maximum upload rate of the seed and leechers is limited to 10 kB/s, Figure 5 shows that the download completion time increases when the RTT is greater than 200 ms. In Section III-A, we show how TCP ramp-up is responsible for this increase in the download completion time. When the upload rate of all the peers is limited to 20 kB/s, in Figure 5, we observe that the download completion time is not a monotonous function of the RTT. Peers having an RTT of 1000 ms have a lower download completion time compared to peers having an RTT of 400 ms. We show that this is the impact of network latency on the delays in receiving BitTorrent control messages in Section III-B.

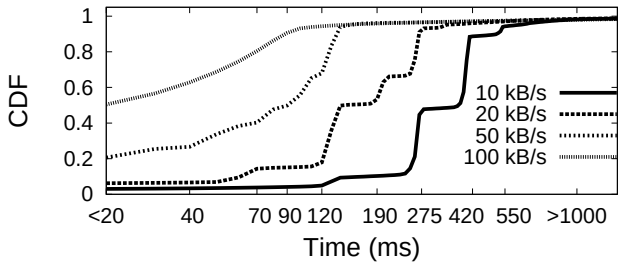
An RTT of 1000 ms between any two peers gives the worst case impact of network latency. Figure 5 thus shows that *network latency has a marginal impact, less than 15%, on the average download completion time for the upload rates observed in public torrents*.

A. Impact of TCP Ramp-Up

The BitTorrent application uses the `send` system call of the socket interface of TCP to upload the pieces of a file. BitTorrent limits the upload rate for a given TCP connection by



(a) Distribution of the time between successive send system calls at the *leechers* when RTT between peers is 0 ms.



(b) Distribution of the time between successive send system calls at the *leechers* when RTT between peers is 400 ms.

Fig. 6: Distribution of the time between successive send system calls at the leechers (semi-log scale). *The RTT does not affect the upload process at the seed and the leechers for high upload rates.*

periodically calling the `send` system call. The time between successive calls of `send` is high when the upload rate is low. Each call of `send` typically results in creation of a TCP segment which is transmitted. If the acknowledgment for the transmitted segment arrives before the subsequent call of `send`, then the upload rate of the application does not require a congestion window ramp-up. *A ramp-up is required on the TCP connections where the time between successive calls of* `send` *is smaller than the RTT*. BitTorrent also enables a peer to simultaneously upload pieces of the file to many peers of a torrent in parallel. An increase in the number of parallel unchokes results in an increase in the time between successive calls of `send`. This is because the upload rate limit is divided over the connections used for the unchoke.

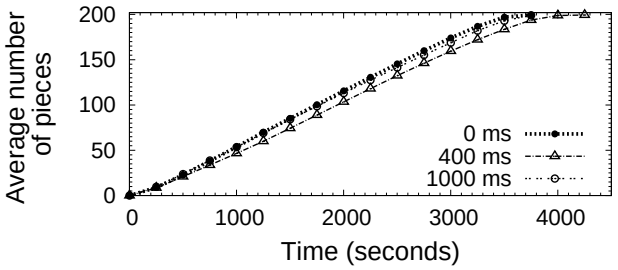
We present the distribution for the time between successive calls of the `send` at all the leechers (over 5 iterations) in Figure 6; the RTT between any two peers is 0 ms in Figure 6a and 400 ms in Figure 6b. For an upload rate of 10 kB/s, in Figure 6a and Figure 6b, we observe that the time between successive `send` calls is greater than 200 ms for a significant number of calls. For these calls, when the RTT between the peers is less than or equal to 200 ms a ramp-up shall not be required. Due to the absence of ramp-up, in Figure 5 for an upload rate limit of 10 kB/s, we observe that an RTT less than or equal to 200 ms does not increase the average download completion time by more than 5% of the average download completion time observed when the RTT is 0 ms. For the upload rate limit of 10 kB/s, an RTT larger than 200 ms may require a TCP ramp-up. Similarly, for the upload rate limits of 20 kB/s, 50 kB/s, and 100 kB/s, in Figure 6, we observe that ramp-up may be required for an RTT greater than 120 ms, 20 ms, and 20 ms respectively. Hence, for an upload rate limit in the range of 10 kB/s to 100 kB/s, TCP ramp-up will be required for an RTT greater than 200 ms. In Figure 5 for an RTT of up to 1000 ms, we observe that TCP ramp-up does not increase the average download completion time by more than 15% of the download completion time observed the RTT is 0 ms. Thus, *TCP ramp-up has a marginal impact, less than 15%, on the average download completion time for the upload rates observed in public torrents*.

B. Impact of Latency on Control Messages

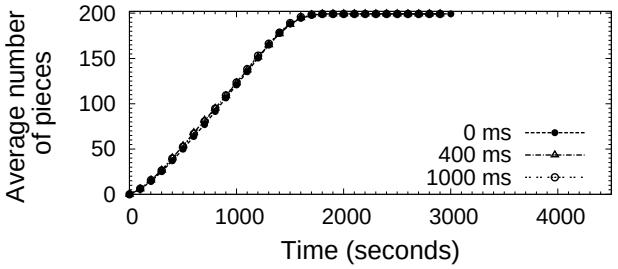
In Figure 5 for an upload rate of 20 kB/s, we observe that the download completion time is not a monotonous function of the RTT. The download completion time when the RTT is 1000 ms is smaller than the download completion time when the RTT is 400 ms. We now present the reasons for this behavior. We would like to comment that this discussion is specific to the implementation of the BitTorrent client we used in the experiments and that *these observations may not be true for other BitTorrent clients that are available*.

Once every 10 seconds, a seed selects from its peer-set a set of leechers to unchoke. The leechers are selected based on their download rate and time since the start of their last unchoke. During an unchoke, the leecher downloading the piece requests for multiple blocks of a piece to pipeline the blocks for the unchoke. The number of blocks requested is a function of the estimated download rate at the leecher. This estimate of the download rate is a moving average and it grows slowly to attain the rate at which the seed uploads to the given leecher. This growth is even slower if the connection requires a TCP ramp-up. The slow growth in the estimated download rate thus results in a slow increase in the number of blocks that are in the pipeline at the seed. We observe that this slow increase, along with the latency in receiving the block requests, results in time periods during which the seed unchoking the leecher is idle, i.e., awaiting block requests. If the leecher selection algorithm of the seed is invoked during these *idle periods*, then the leecher *will not be selected* for an unchoke resulting in an abrupt termination of the unchoke. We observe that this abrupt termination of the unchoke occurs frequently when the upload rate limit is 20 kB/s and when the RTT between the peers is greater than 120 ms and less than 800 ms.

We now show how the abrupt termination of an unchoke affects the availability of pieces at the leechers. Once a piece is downloaded, the leecher can then upload this piece to other leechers in its peer-set. After the piece download, the leecher sends a *HAVE* message with the piece identifier to the peers in its peer-set. The *HAVE* message indicates that the piece can be downloaded from this leecher. In Figure 7, we use the *HAVE* messages sent by the leechers to show the evolution of the average number of pieces (over 10 iterations) that are available at the leechers. For reasons mentioned above, when the upload rate limit of the seed and leechers is 20 kB/s, an RTT of 400 ms between the seed and a leecher typically results in the abrupt



(a) Number of pieces available with the leechers when the upload rate is limited to 20 kB/s.



(b) Number of pieces available with the leechers when the upload rate of the leechers is limited to 20 kB/s and that of the seed is limited to 50 kB/s.

Fig. 7: Evolution of pieces available at the leechers. *Latency has marginal impact on the pieces available with the leechers.*

termination of the unchoke. The abrupt termination results in an incomplete download of a piece and affects the availability of pieces as shown in Figure 7a. We do not observe abrupt termination of the unchoke when the RTT is 0 ms and when the RTT is 1000 ms. Hence, the average download completion time when the RTT is 400 ms is greater than the average download completion observed when the RTT is 1000 ms.

We do not observe abrupt termination of unchokes when we have a fast seed (upload rate limited to 50 kB/s) in a torrent with slow leechers (upload rate limited to 20 kB/s); Figure 7b shows the evolution of pieces for this torrent configuration. We observe a similar evolution for the *HAVE* messages when the upload rate at the peers is limited to 50 kB/s and 100 kB/s.

C. Summary

In this section, we emulated a large range of RTT values, from 0 ms to 1000 ms, to study the impact of network latency on the download completion time. We use an RTT of 1000 ms to give the worst case impact of network latency for a given upload rate. We observe that an RTT of up to 1000 ms between any two peers has a marginal impact, less than 15%, on the average download completion time of a file. For an upload rate of 20 kB/s, we observe that the download completion time is not a monotonous function of the RTT. This behavior emphasizes *that the models for TCP throughput cannot be directly used to study the impact of network latency on the time required to download a file using BitTorrent.*

In the next section, we relax the condition of same latency between any two peers in the torrent to confirm that the scenario of same latency gives us the worst case impact of network latency.

AS	Latency on Loopback (ms)	Latency on Ethernet (ms)
AS_1	2	5
AS_2	5	15
AS_3	10	25
AS_4	25	100
AS_5	50	100

TABLE I: Latency values on the loopback and ethernet device while emulating an AS on a machine.

	AS_1	AS_2	AS_3	AS_4	AS_5
AS_1	8 ms	40 ms	60 ms	210 ms	210 ms
AS_2	40 ms	20 ms	80 ms	230 ms	230 ms
AS_3	60 ms	80 ms	40 ms	250 ms	250 ms
AS_4	210 ms	230 ms	250 ms	100 ms	400 ms
AS_5	210 ms	230 ms	250 ms	400 ms	200 ms

TABLE II: RTT between a pair of leechers. *RTT between a leecher in AS_1 and a leecher in AS_5 is 210 ms.*

IV. Heterogeneous Latency

We now present results to confirm that the observations made in Section III are valid even when the condition of fixed latency is relaxed. We relax the condition of same latency between any two peers by emulating ASes in the following manner. Public torrents have peers that are spread out geographically. A pair of peers in the same AS typically have a smaller network latency compared to a pair of peers that are present in different ASes. For our experiments, we use a private torrent with peers distributed in emulated ASes. For our experiments, we emulate an AS using one machine. We assume the same intra-AS latency and we also assume that the intra-AS latency is less than the inter-AS latency. We assume that all the ASes are fully meshed.

A. Emulation of ASes

As in the case of homogeneous latency, we use four machines in each of the experiments. We use a private torrent with 300 leechers, one tracker, and one initial seed. We emulate four ASes: three ASes each with 100 leechers, and the fourth AS to emulate the AS of the seed and the tracker. The four ASes used in these experiments were chosen from a set of five ASes (AS_1, AS_2, AS_3, AS_4, and AS_5).

We now present an explanation for emulating these five ASes. In Figure 5, for an upload rate of 20 kB/s, we observe that an RTT smaller than 120 ms between any two peers has a smaller impact on the download completion time as compared to an RTT larger than 120 ms. In three of the five ASes, namely AS_1, AS_2, and AS_3, the RTT between a pair of peers in these three ASes is less than 120 ms. The RTT between a pair of peers in AS_4 is less than 120 ms; the RTT between a peer in AS_4 and any other peer is greater than 120 ms. Finally, a peer in AS_5 has an RTT greater than 120 ms with any other peer. As we use one machine to emulate an AS, peers in the same AS use the loopback device to communicate with each other; the peers use the ethernet device to communicate with all the other peers in the torrent. We emulate the latency values given

	AS_1	AS_2	AS_3	AS_4	AS_5
AS'_1	20 ms	40 ms	60 ms	210 ms	210 ms
AS'_2	40 ms	60 ms	80 ms	230 ms	230 ms
AS'_3	60 ms	80 ms	100 ms	250 ms	250 ms
AS'_4	210 ms	230 ms	250 ms	400 ms	400 ms
AS'_5	210 ms	230 ms	250 ms	400 ms	400 ms

TABLE III: RTT between the seed and the leechers in the torrent. *AS'_i indicates that seed is placed in as AS with latency values similar to AS_i. RTT between the seed in AS'_1 and a peer in AS_1 is 20 ms.*

in Table I for the ethernet and loopback device while using a machine to emulate an AS.

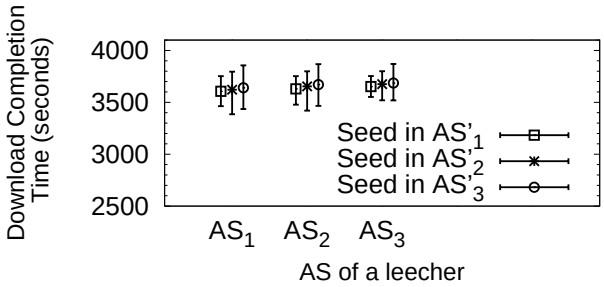
We now give an example to show how Table I can be used to find the RTT between a pair of leechers. The RTT between a leecher in AS_1 and a leecher in AS_2 is 40 ms (5+15+15+5) as the leechers use the ethernet device to communicate with each other. The RTT between a pair of leechers in AS_1 is 8 ms (2+2+2+2) as the leechers use the loopback device to communicate with each other. Table II gives the RTT values between all such pairs of leechers.

For our experiments, we assume that the seed and the tracker are placed in a dedicated AS with no leechers. We use AS'_i to denote that the seed and the tracker are placed in an AS with the same latency values as AS_i. For example, AS'_1 implies that the seed and tracker are placed in an AS having the same latency values as AS_1. Table III gives the RTT values between the seed and the leechers.

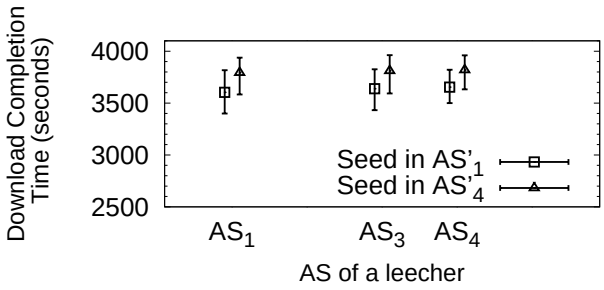
B. Presentation and Discussion of Results

Figure 8 show the impact of heterogeneous latency on the download completion time of a 50 MB file when the upload rate of the peers is limited to 20 kB/s. The impact of network latency when the upload rate limit is 50 kB/s is presented in Figure 9. In Figure 8 and Figure 9, the X-axis represents the AS of the leechers present in the torrent, and the Y-axis represents the download completion time in seconds. The figures present the average download completion time over 10 iterations; the error bars indicate the minimum and maximum download completion time observed in 10 iterations.

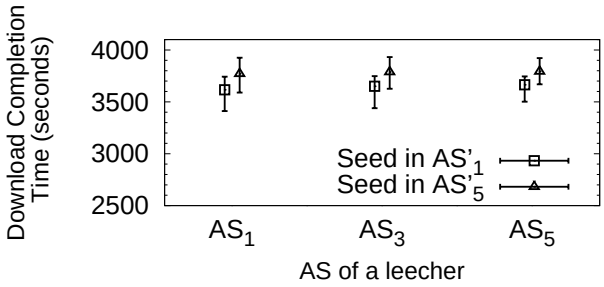
Figure 8a shows the outcome of three experiments with leechers in AS_1, AS_2, and AS_3. For a given experiment, the seed was either in AS'_1, AS'_2, or AS'_3. Despite the different RTT values between the peers, in the three experiments presented in Figure 8a, we observe that the difference in the average download completion time is less than 15%. According to Table II, the RTT between any two peers in these experiments was less than 120 ms. For the experiments presented in Figure 8b and Figure 8c, a peer in AS_1 or AS_3 and another peer in AS_4 or AS_5 have an RTT greater than 120 ms. Despite the wide range of RTT values, we observe that the average download completion time in Figure 8b and in Figure 8c is not more than 15% of the average download completion time observed in Figure 8a, i.e., when the RTT between a pair of peers is less than 120 ms.



(a) Download completion time for leechers present in AS_1, AS_2, and AS_3. The difference in the average download completion time is less than 15%.



(b) Download completion time for leechers present in AS_1, AS_3, and AS_4. An RTT of 400 ms between a leecher in AS_4 and the seed in AS'_4 does not increase the average download completion time by more than 15%.



(c) Download completion time for leechers present in AS_1, AS_3, and AS_5. An RTT of 400 ms between a leecher in AS_4 and the seed in AS'_4 does not increase the average download completion time by more than 15%.

Fig. 8: Download completion time of a 50 MB file by leechers in a given AS when the maximum upload rate of all the peers is 20 kB/s. *Despite the wide range of RTT values emulated, the difference in the download completion time is less than 15%.*

According to Table II, a peer in AS_5 and the seed in AS'_5 have an RTT of 400 ms. We observe that the average download completion time in Figure 8c is smaller than the average download completion time observed in Figure 5 for an upload rate limit of 20 kB/s and an RTT of 400 ms. This shows that *the scenario of homogeneous latency can be used to give a worst case impact of network latency for a given upload rate*. When the upload rate limit is set to 50 kB/s, in Figure 9 we observe that the difference in the average download completion time is less than 15%.

C. Summary

In this section, we relax the condition of fixed latency between any two peers in a torrent. We observe that an RTT of up to 400 ms has a marginal impact, less than 15%, on

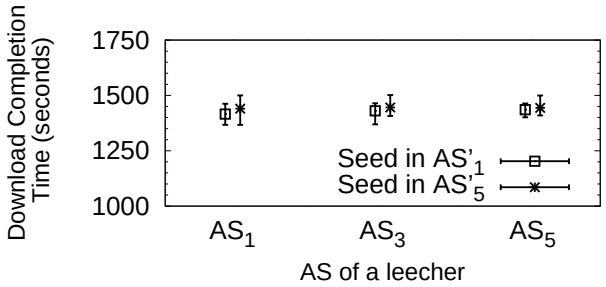


Fig. 9: Download completion time of a 50 MB file by leechers present in a given AS when the maximum upload rate of all the peers is 50 kB/s. *Despite the wide range of RTT values emulated, the difference in the download completion time is less than 15%.*

the average download completion time. These observations show that the *upload process at the peers is not sensitive to the variations in the TCP throughput due to the change in latency*. These observations also confirm that, for a given upload rate among the peers, *the scenario of homogeneous latency provides an upper bound on the download completion time of a file when the maximum latency between any two peers in a torrent is known*.

V. Impact of Packet Loss

We now present the impact of packet loss on the download completion time of a file. For our experiments, we emulate a 5% packet loss on the ingress interface of the loopback and ethernet devices of the machines.

In Table IV, we present the average download completion time of a 50 MB file for a given upload rate limit and a given RTT between any two peers. As in the case of homogeneous latency, we consider a torrent consisting of a one tracker, one seed, and a flash crowd of 300 leechers. We observe the download completion time for the following network conditions.

1) *Homogeneous latency and TSO is disabled*. We do not emulate packet losses in this scenario. As packet losses are not emulated, this setting gives the outcome of experiments performed on clusters that do not support TSO. The results of this scenario are discussed in detail in Section III.
2) *Homogeneous latency and TSO is enabled*. We do not emulate packet losses in this scenario. The results present the outcome of experiments performed without emulating packet loss on clusters that support TSO.
3) *Homogeneous latency with a loss rate of 5% and TSO disabled*. We use this scenario to emulate network conditions present in the Internet.

In Section III, we observed the impact of network latency when TSO is disabled and packet loss is not emulated. We observe that for an RTT of up to 1000 ms, TCP ramp-up and the delays in receiving the control messages have a marginal impact, less than 15%, on the average download completion time of a file.

		Average Download Completion Time		
Upload Rate	RTT	TSO disabled Loss Rate 0%	TSO enabled Loss Rate 0%	TSO disabled Loss Rate 5%
10 kB/s	0 ms	7314.7 s	7268.4 s	7359.1 s
	400 ms	8006.0 s	7823.4 s	8183.6 s
	1000 ms	8274.19 s	8060.6 s	8827.6 s
20 kB/s	0 ms	3634.9 s	3728.3 s	3711.2 s
	400 ms	4023.9 s	3985.3 s	4034.3 s
	1000 ms	3768.3 s	3796.6	4102.7 s
50 kB/s	0 ms	1437.96 s	1433.7 s	1432.7s
	400 ms	1457.2 s	1463.9 s	1476.9
	1000 ms	1466.9 s	1470.5 s	1638.2 s
100 kB/s	0 ms	838.9 s	828.9 s	832.7 s
	400 ms	863.4 s	860.4 s	940.20 s
	1000 ms	844.5 s	865.4 s	1619.87 s

TABLE IV: Impact of RTT and loss rate on download completion time. *An RTT of up to 400 ms and the loss rate of up to 5% does not increase the average download completion time by more than 15% of the average download completion time observed when the RTT is 0 ms and the loss rate is 0%.*

When TSO in enabled and packet losses are not emulated, we observe that the impact of network latency on the download completion time is similar to that observed in the case of homogeneous latency, which has been presented in Section III. These results show that *BitTorrent experiments performed on clusters that support TSO shall produce results that are similar to those performed on clusters that do not support TSO.*

We now discuss the scenario of homogeneous latency with packet loss when TSO is disabled. While emulating a packet loss, each packet en-queued by NetEm is dropped with a probability controlled by the loss rate. A packet loss result in retransmission of the packet by the source. The source reduces the congestion window in response to the loss and then ramps-up its congestion window to attain the desired upload rate. We observe that for an upload rate limit of 20 kB/s and 50 kB/s, an RTT of 1000 ms and a loss rate of 5% between any two peers does not increase the average download completion time by more than 15% of the average download completion time observed when *neither latency nor packet loss was emulated*. However, when the upload rate limit is 10 kB/s or 100 kB/s, the RTT between any two peers is 1000 ms, and when the loss rate is 5%, we observe that the average download completion time is more than 15% of the download completion time when the RTT is 0 ms and the loss rate is 0%. An RTT of 1000 ms between any two peers is unrealistic as 99.8% links probed by iPlane have an RTT less than 1000 ms. However, *an RTT of up to 400 ms is realistic* as 98% of the links probed by iPlane have an RTT less than 400 ms. For the upload rate limit from 10 kB/s to 100 kB/s, when the RTT between any two peers is 400 ms and the loss rate is 5%, we observe that the download completion time is not more than 15% of the download completion time observed when the RTT is 0 ms and loss rate is 0%. In Section IV we observe that the scenario of homogeneous latency gives an upper bound on the impact of a given network latency. Therefore, the results presented in Table IV show that, for an upload rate limit from 10 kB/s to 100 kB/s, *a loss rate of up to 5% and an RTT of up to 400 ms*

between any two peers in the torrent has a marginal impact, less than 15%, on the average download completion time of a file.

VI. Concluding Remarks

In this paper we present the impact of network latency and packet loss on the download completion time of a file distributed using BitTorrent. We use the download completion time as the metric for evaluation because the BitTorrent users are primarily interested in the download completion time of a file.

We first studied the impact of network latency on the download completion time. For a given upload rate limit, we emulated the same latency among the peers to give a worst case impact of network latency on the download completion time. The download completion time can be affected by TCP ramp-up and the delays in receiving the BitTorrent control messages. We therefore studied the impact of network latency on the download completion time by studying the impact of network latency on TCP ramp-up and the delays in receiving the control messages. For our experiments, we varied the upload rate limit from 10 kB/s to 100 kB/s and the RTT from 0 ms to 1000 ms. We observe that the TCP ramp-up and the delays in receiving the BitTorrent control messages only have a marginal impact, less than 15%, on the average download completion time. The high RTT values used in our experiments also emulate torrents with peers that are not only geographically apart but also connected with high capacity links that support the BitTorrent upload rate without causing congestion. Our results show that *experiments performed on Grid'5000 give results similar to those performed on testbeds such as PlanetLab that have geographically distributed hosts that are connected by high capacity links*. We also study the impact of network latency on the delays in receiving the control messages; this impact cannot be captured using the traditional models for TCP throughput.

We then studied the impact of packet loss by emulating a loss rate of 5%. For the upload rates seen in public torrents, from 10 kB/s to 100 kB/s, we observe that realistic RTT values of up to 400 ms and a packet loss rate up to 5%, have a marginal impact, less than 15%, on the average download completion time. We performed our experiments over a wide range of RTT values, from 0 ms to 1000 ms, and a wide range of packet loss rates, from 0% to 5%, to study the impact of network latency and packet loss. Our results show that *experiments can be performed on dedicated clusters, such as those present in Grid'5000, without explicitly emulating latency and packet loss between the peers in a torrent*.

We also studied the impact of using devices that support TSO on the outcome of BitTorrent experiments performed on clusters. Our results show that BitTorrent experiments performed on clusters that support TSO produce results that are similar to those performed on dedicated clusters that do not support TSO.

Our main conclusion is that, for upload rates seen in public torrents, network latency and packet loss have a marginal impact on the download completion time of a file, hence, dedicated clusters such as Grid'5000 can be safely used to perform realistic and reproducible BitTorrent experiments.

VII. Acknowledgment

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, being developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr).

References

[1] S. Floyd and V. Paxson, "Difficulties in Simulating the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 4, pp. 392–403, 2001.
[2] S. Floyd and E. Kohler, "Tools for the Evaluation of Simulation and Testbed Scenarios draft-irtf-tmrg-tools-05.txt," 2008.
[3] D. R. Choffnes and F. E. Bustamante, "Pitfalls for Testbed Evaluations of Internet Systems," *Computer Communication Review*, vol. 40, no. 2, pp. 43–50, 2010.
[4] "http://www.planet-lab.org/."
[5] M. Dischinger, A. Haeberlen, I. Beschastnikh, P. K. Gummadi, and S. Saroiu, "SatelliteLab: Adding Heterogeneity to Planetary-Scale Network Testbeds," in *SIGCOMM '08: Proceedings of the ACM SIGCOMM 2008 conference on Data Communication*. New York, USA: ACM, 2008, pp. 315–326.
[6] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using PlanetLab for Network Research: Myths, Realities, and Best Practices," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 17–24, 2006.
[7] "https://www.grid5000.fr."
[8] B. Cohen, *The BitTorrent Protocol Specification*, Jan 2008.
[9] D. Qiu and R. Srikant, "Modeling and Performance Analysis of BitTorrent-Like Peer-to-Peer Networks," in *SIGCOMM '04: Proceedings of the 2004 conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. New York, NY, USA: ACM, 2004, pp. 367–378.
[10] A. R. Bharambe, C. Herley, and V. N. Padmanabhan, "Analyzing and Improving a BitTorrent Networks Performance Mechanisms," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications.*, April 2006, pp. 1 –12.
[11] Y.-M. Chiu and D. Y. Eun, "Minimizing File Download Time in Stochastic Peer-to-Peer Networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 253–266, 2008.
[12] M. Allman, V. Paxson, and E. Blanton, "RFC5681: TCP Congestion Control," 2009.
[13] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm," *SIGCOMM Computer Communication Review*, vol. 27, no. 3, pp. 67–82, 1997.
[14] "http://bt-instru.gforge.inria.fr/."
[15] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-friendly High-speed TCP Variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.
[16] A. Norberg, *uTorrent Transport Protocol*, Jan 2010.
[17] S. Shalunov, "Low Extra Delay Background Transport (LEDBAT) draft-ietf-ledbat-congestion-01.txt," Mar 2010.
[18] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and Sharing Incentives in BitTorrent Systems," in *Proceedings of the 2007 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. ACM, 2007, pp. 301–312.
[19] S. Hemminger, "Network Emulation with NetEm," in *Proceedings of the 2005 Linux Conference Australia, Canberra, Australia*, 2005.
[20] "http://iplane.cs.washington.edu/."
[21] H. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iPlane: An information plane for distributed services," in *Proceedings of OSDI*, 2006, pp. 367–380.
[22] J. C. Mogul, "TCP offload is a dumb idea whose time has come," in *HotOS*, 2003, pp. 25–30.
[23] D. Freimuth, E. Hu, J. LaVoie, R. Mraz, E. Nahum, P. Pradhan, and J. Tracey, "Server Network Scalability and TCP offload," in *ATEC '05: Proceedings of the USENIX Annual Technical Conference*. Berkeley, CA, USA: USENIX Association, 2005, pp. 15–15.
[24] "http://mawi.nezu.wide.ad.jp/mawi/."