Peer-to-Peer Networking and Applications manuscript No.
(will be inserted by the editor)

ModelNet-TE: An emulation tool for the study of P2P and Traffic Engineering interaction dynamics

Dario Rossi · Paolo Veglia · Matteo Sammarco · Federico Larroca

the date of receipt and acceptance should be inserted later

Abstract In the Internet, user-level performance of P2P applications may be determined by the interaction of two independent dynamics: on the one hand, by the end-to-end control policies applied at the P2P application layer (L7); on the other hand, by Traffic Engineering (TE) decisions taken at the network level (L3). Currently available tools do not allow to study L7/L3 interaction in realistic settings, due to a number of limitations. Building over ModelNet, we develop a framework for the real-time emulation of TE capabilities, named ModelNet-TE, that we make available to the scientific community as open source software.

ModelNet-TE allows (i) to deploy real unmodified Internet P2P applications, and to test their interaction with (ii) many TE algorithms, as its design allows to easily integrate other TE algorithms that those we already provide, (iii) in a furthermore controlled network environment. Due to these features, ModelNet-TE is a complementary tool with respect to hybrid simulation/protoyping toolkits (that constrain application development to a specific language and framework, and cannot be used with existing or proprietary applications) and to other open testbeds such as PlanetLab or Grid5000 (lacking of control or TE-capabilities respectively). ModelNet-TE can thus be useful to L7-researchers, as it allows to seamlessly and transparently test any existing P2P application without requiring any software modification. At the same time, ModelNet-TE can be useful to L3-researchers as well, since they can test their TE algorithms on the traffic generated by real applications.

As a use case, in this work we carry on an experimental campaign of L7/L3 routing layers interaction through ModelNet-TE. As TE we consider the classic minimum congestion load-balancing, that we compare against standard IP routing. As example P2P applications, we take BitTorrent, one among the most popular file-sharing applications nowadays, and WineStreamer, an open source live-streaming application. We emulate BitTorrent and WineStreamer swarms over both realistic topologies (e.g., Abilene) and simplistic topologies that are commonly in use today (e.g., where the bottleneck is sited at the network edge) under a variety of scenarios.

Results of our experimental campaign show that user-level performance may be significantly affected by both the TE mechanism in use at L3 (e.g., due to interactions with TCP congestion control or P2P chunk trading logic), as well as scenario parameters that are difficult to control in the wild Internet, which thus testifies the interest for tools such as ModelNet-TE.

Dario Rossi · Paolo Veglia
Telecom ParisTech, Paris, France.
E-mail: firstname.lastname@enst.fr

Matteo Sammarco
Université Pierre et Marie Curie (UPMC), LIP6, France.
E-mail: matteo.sammarco@lip6.fr

Federico Larroca
Universidad de la República, Uruguay.
E-mail: flarroca@fing.edu.uy

1 Introduction

It is desirable that P2P algorithms and protocols are tested before they can be deployed at large scale. Simulation-based performance evaluation is often non representative of real-world dynamics, even when simulations are carried on exploiting the very same prototype code. As such, recent research on P2P networking embraced an experimental approach to assess P2P protocol performance. This usually involves either (i) the deployment of small to large-scale testbeds, such as Grid5000 [28], where the environment is fully under control but not representative of real world dynamics, or (ii) the use of large-scale testing facilities, such

PlanetLab [49] or OneLab [44], that benefit of the realism of the wild Internet, but lacks however of control.

Researchers face thus the following dilemma. On the one hand, their testbed results may be easily reproducible, but hardly representative of real-world performance: in this case, the large development and deployment effort invested in the testbed does not payoff, since the offered level of realism only slightly exceeds the one achievable by simulation. On the other hand, carrying on experiments over the wild Internet allows to gather realistic results, though in this case the experimental scenario is not under control and generally hardly reproducible. Loss of *control* means that it may be very hard to correlate the observed performance with their root cause, so that experimental results become hard to interpret. Loss of *reproducibility* –which has been a requirement of experimental science since Hipparchus (ca.190 BC – 120 BC) measurement on Earth axial precession– can further hinder cause-effect relationships, and is therefore not a favorable environment for beta testing.

Efforts such as ModelNet [62] offer a third way, enabling the control of the *core network topology*. Unlike simulative approaches, ModelNet uses a full networking stack, meaning that the ability to control the network does not come at the price of the performance evaluation realism. Notice that the control of the core network topology is not available in Grid5000, PlanetLab and OneLab: hence, ModelNet does not try to fully substitute to these existing experimental facilities, but rather to complement them. In a sense, ModelNet stands between these approaches for being more realistic than Grid5000 or smaller testbeds and, at the same time, more controllable than PlanetLab. Furthermore, experiments on ModelNet can be reproducible (L3 topology, traffic condition, etc.) as in Grid5000 and unlike in PlanetLab. These capabilities make it a valuable complementary tool for P2P application developer to test their systems. ModelNet is however only capable of shortest-path IP routing, which represents its major drawback. This limitation makes it is not suitable for research in Traffic Engineering (TE), nor completely realistic as emulation environment, since no source-routing or load-balancing techniques, though widely used as of today [8], are available for testing.

In this work, we present ModelNet-TE, an extension of ModelNet that enables TE emulation and experiments. Furthermore, we port the original ModelNet core code from BSD to Linux, making it available to the scientific community [40]. The ModelNet-TE tool is interoperable, scalable and flexible. Interoperability and scalability are directly inherited from the original ModelNet code, that allows to run possibly thousands of unmodified application instances (provided that certain constraints are met, which we detail in the following). Flexibility is instead a key of ModelNet-TE, as we took special emphasis in the design of a reusable toolbox, where researchers can easily integrate their own TE algorithms beside those that we already provide [25, 42].

We use ModelNet-TE to evaluate the uncoordinated interaction between Traffic Engineering (TE) at the network layer (L3) and end-to-end control policies applied by P2P systems at the application layer (L7). Indeed, though a number of work have studied the issue of selfish routing [30, 31, 35, 45, 50, 56] most of these work adopt a theoretical approach, which is especially true for the case of the uncoordinated interaction of routing dynamics at different levels [30, 31, 35]. On the other hand, while several experimental studies of popular P2P applications exists [15, 22, 48, 51, 57, 59, 65] they nevertheless neglect the interaction with the underlying network. While their approach is necessary to understand application dynamics, it does not allow ISPs to understand the impact of TE on the traffic of their users; nor it allows P2P developer to assess how do their algorithms perform over a reactive network.

Aiming at filling this gap, we study the L3/L7 interaction via ModelNet-TE. To prove the flexibility of ModelNet-TE, and to gather a full blown set of results, we carry on an experimental campaign that, as sketched in Fig. 1, considers a rich set of (i) L3 topologies and routing algorithms and of (ii) L7 applications and peer population models. At L3, we consider both a simplistic pure overlay model, where the bottleneck is only at the access, as well as the popular Abilene topology spanning across the US, in which any link can become a bottleneck (depending on the traffic matrix induced by the P2P application). As reactive L3 Traffic Engineering we implement a multi-path load balancing algorithm [25], that we compare to standard shortest path IP routing. At L7, we consider two reactive P2P applications, namely BitTorrent [14], the most popular file-sharing application nowadays, and WineStreamer [13,34], an open source live streaming application. Furthermore, we consider both a uniform peer population across the network, or a skewed population, that reflects the actual number of citizen in major US urban areas.

We point out that we use BitTorrent and WineStreamer as *examples* of filesharing and live-streaming P2P applications of our experimental campaign. At the same time, as ModelNet-TE is engineered to *transparently work* with any P2P application, it requires no modification or instrumentation to the P2P application. Similarly, while we limitedly consider an *example* of multi-path load balancing, other Traffic Engineering algorithms can be easily integrated in ModelNet-TE as we will show in the following. As such, the ModelNet-TE tool can be used by other researchers to perform similar experimental campaigns on completely different sets of P2P applications and TE algorithms. We therefore believe ModelNet-TE to be a valuable addition for the experimental evaluation of P2P systems – to which it adds the ability to *control the underlying network*, a feature that was missing

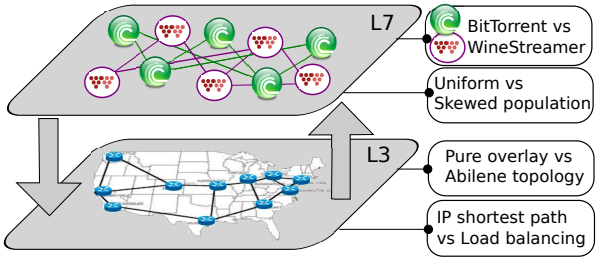


Fig. 1 Synopsis of the elements taken into account in this paper

in the other available tools (see Sec. 5 for a more thorough comparison).

Summarizing, this paper achieves two main milestones. First, we offer the scientific community a full blown, open source, customizable network emulator with real-time traffic engineering capabilities. The tool can be used with modest investment (i.e., few servers), to emulate medium to large scale overlays. Second, we carry the first thorough campaign, exploiting an experimental methodology, that focuses on the interaction of P2P dynamics with the underlying L3 network. Experimental results yield the following interesting insights: (i) bottleneck in the network (which recently arose in case of popular applications and content [17, 38]) may have a profound impact on the P2P application performance; (ii) the peer population model, other than shaping the traffic perceived by the L3 network, may significantly contribute in determining P2P performance; (iii) traffic engineering may ameliorate network-centric ISP performance (e.g., equalize traffic on links) to the detriment of user-centric P2P performance (e.g., due to unexpected interactions with TCP transfers or P2P trading logic).

The rest of this paper is organized as follows. In Sec. 2 we present a system-level view of the original ModelNet-TEemulator, describing the TE extension and the load balancing algorithm we implement, providing an initial assessment of its scalability as well. Sec. 3 provides a detailed description of the emulated scenarios, describing the P2P applications used, the different network and population models, and the evaluation metrics. Results of our experimental campaign are then reported in Sec. 4. Finally, related work are overviewed in Sec. 5, before conclusive remarks are drawn in Sec. 6.

2 ModelNet-TE Emulator

2.1 ModelNet Primer

The original ModelNet software [62] is an IP network emulator, which allows to run unmodified applications plugging them into realistic, large-scale networks. ModelNet implements emulated virtual topologies that are independent from the physical testbed interconnection. A synoptic of its architecture is sketched in Fig. 2. The ModelNet environment consists of two kind of machines, HOST and CORE, interconnected by a physical LAN (address 192.168.0.0/24 in the figure). The CORE machine emulates the virtual network with an arbitrary topology, while each HOST machine runs multiple instances of the application under test (in our case, BitTorrent or WineStreamer clients, see Sec. 3.2). Each instance is bounded to a Virtual Node (VN) and a virtual IP address belonging to a private subnet (typically the 10.0.0.0/8 network), dedicated to ModelNet emulation. While in the physical topology VNs runs on HOST machines, in the virtual topology each VN is attached to a Gateway node (GW), that constitutes its ingress/egress point in the emulated network.

Notice that, for the emulation to be successful, each packet generated by any VN application instance must be delivered to the CORE over the physical LAN: this is because, for each packet, IP network emulation takes place at kernel level in the CORE. Emulation tasks can be summarized as follow: using the source and destination virtual IP addresses of packets coming from applications running on HOST machines, the CORE determines a path through the virtual topology and handles the packets accordingly. Each hop on this path has a given bandwidth, queuing, propagation delay, and packet loss characteristics: thus, this hop-by-hop emulation lets IP traffic experience realistic wide area effects, possibly including congestion on core links. Notice that packet emulation occurs in real time, and packet delays are handled with millisecond accuracy.

For the sake of clarity, Fig. 2 depicts the case of an application instance bound to a virtual node VN having IP address 10.0.0.1, that wants to send a packet to a VN having IP address 10.0.0.4. Though both VNs are on the same physical HOST, packets are however delivered to the CORE over the physical LAN, through a kernel level hack happening at the interface of the machine hosting the source VN[1]. The CORE routes then the packet in the emulated topology: once the packet has crossed all path hops (in the emulated topology), it is delivered (again through the physical LAN) to the HOST to which the destination VN is bound.

2.2 ModelNet-TE Overview

To overcome the single-path limit, we have modified the original ModelNet kernel module to allow multiple parallel path to be used between any source destination pair: we call the improved emulator ModelNet-TE. We have ported

[1] ModelNet-TE flips a bit of the virtual destination address, which forces packets to exit the HOST (instead of being "captured" by the loop-back interface), and be directed to the CORE (which is set as HOST default gateway). The same bit of the IP destination address is then flipped again at packet reception in the CORE.

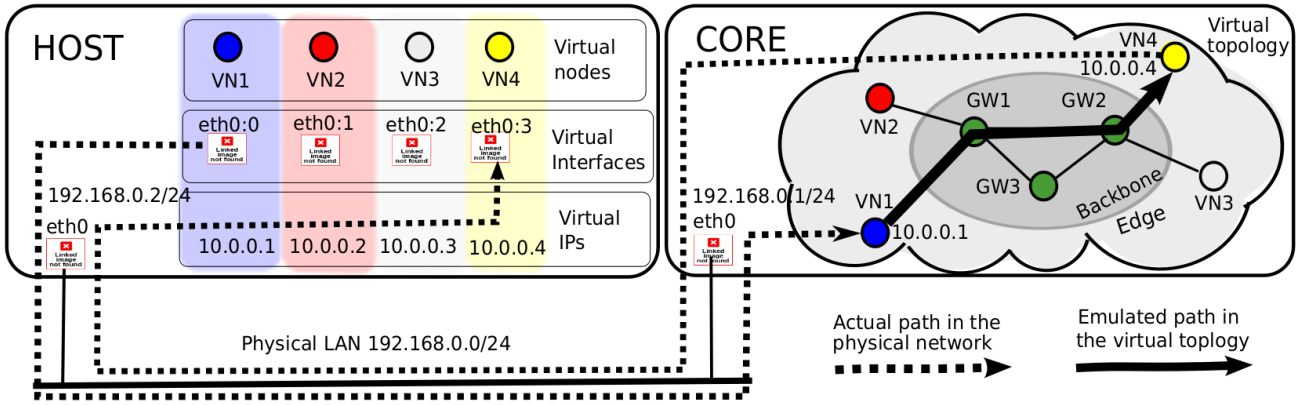


Fig. 2 ModelNet low-level architecture

the original BSD code to the Linux kernel, that we make available[2] at [40].

Notice that ModelNet-TE inherits from ModelNet its ability to seamlessly and transparently work with unmodified applications. This implies that ModelNet-TE can be used with any existing P2P application, as it requires no modification or instrumentation[3] to the P2P application.

We now briefly describe the improved internal ModelNet-TE structure. As can be seen in Fig. 2, the topologies emulated by ModelNet-TE can be logically divided in two sections: a *backbone* part that connects the gateways nodes GW and an *edge* part that comprises the set of access links interconnecting each VN to a single GW node. In practice, we can imagine that each GW to which VNs are attached, acts as WiFi hotspot or an ADSL DSLAM. We point out that the TE algorithms only apply to the *backbone* part of the network, acting thus on aggregated traffic demands coming from the network edge.

The high-level idea of the ModelNet-TE extension is depicted in Fig. 3, where all the relevant components are represented, as well as their relationship and their mean of interaction. Basically, the *emulated topology* is described through an XML file, as in the original ModelNet-TE. Topology definition consists in specifying both edge and backbone link, by fully defining the property of each link (such as bandwidth, delay, loss probability, queue size, etc.) and their topological interconnection structure.

Routing tables of nodes in the emulated topology are instead specified in a *source routes* file, representing the Forwarding Information Base (FIB). In ModelNet, FIB is a text file containing, for each VN couple, the list of hops that each packet coming from source VN_S and destined to VN_D has to cross. In ModelNet-TE, the FIB is extended in order to handle multiple routes between each VN pair: more specifically, a *probability* is associated to each of the multiple paths connecting each VN couple. The kernel-level forwarding module applies then this probability for each packet: i.e., on each new packet arrival, one of the multiple paths is chosen at random according to the specified probability.

Notice that the forwarding module only applies per-path probabilities, but expects an external *L3 TE routing module* to set them: this way, routing optimization is *decoupled* from low-level forwarding, making it easy to integrate new algorithms. Notice also that, in the context of this paper, the TE mechanism we are considering is limited to per-packet multi-path forwarding, though ModelNet-TE also supports per source-destination pair load balancing (as the FIB handles source-routed paths). We point out that studies such as [9] have shown that per-packet load balancing, though not predominant, is not rare at all in today ISPs. Furthermore, to prove the flexibility of ModelNet-TE and the extendibility due to the FIB/forwarding decoupling, ModelNet-TE already implements two different TE algorithms [25, 42]. In this work, for reasons of space, we use only one among [25, 42], that we briefly describe in Sec. 2.3; we instead refer the reader to our technical report [24] for an experimental performance evaluation of P2P systems with the other algorithm [42].

We point out that centralized TE algorithms can easily run on ModelNet-TE. Notice that, given that all GW traffic transits through the CORE machine, the TE optimization algorithms running on the CORE benefit of the knowledge of the Traffic Matrix (TM), and of the load on each link. TM is continuously updated by the CORE: more precisely, at a configurable periodic interval, the CORE exports TM information from the kernel, writing it in an output text file

[2] As our patch applies only to specific versions of the Linux kernel (namely `2.6.18` or `2.6.22`), and so as to reduce the startup time for new users, we directly provide full ready-to-use system *images* of the patched CORE and HOST machines, containing the source code as well.

[3] Clearly, instrumentation of the P2P application, whether possible, can bring a more detailed view of the QoE perceived by P2P users. At the same time, we provide basic QoS monitoring of end-point traffic (e.g., traffic volumes, delay, jitter, losses, etc.) that are general enough for any P2P applications.

containing the *load* of each link on the network (expressed as the sum of the PDU lengths that crossed each link during that timeframe). TM information can then be used as input by the TE toolbox running in the user space, so to compute the FIB to be used by the CORE in the kernel-level forwarding process, as illustrated in Fig. 3.

The routing/forwarding decoupling is not only a natural choice, as it follows from standard operation in IP networks, but also simplifies the integration of new modules by (i) providing a simple, clean and natural interface for the Linux environment (i.e., a file to read TM statistics from the kernel, a file to write FIB information for the kernel) and (ii) avoiding constraints on the time-scale of TE optimization module (which asynchronously runs in user space). To better grasp the advantages of this design choice, let us consider which one between the (i) TE optimization and (ii) FIB update process may constitute a bottleneck. Consider the ideal case of an instantaneous optimization algorithm: then, update rate could only be limited by the time it takes ModelNet-TE to read the new FIB from disk. As, in our experience, loading the update routing tables takes less than a second (for moderate size networks of 10-50 nodes), this poses no constraint on the choice of TE timescale. Indeed, the bottleneck in the FIB reconfiguration rate is more likely tied to the TE algorithm running time, that depends on the algorithm complexity, and is generally tied to the solution of an optimization problem.

With respect to our L7 vs L3 routing interaction study in a P2P vs TE scenario, notice that once the source routes are updated by the TE module, the CORE will use the updated routes in the emulated topology, possibly triggering in turn changes at L7 due to P2P traffic dynamics, as depicted in Fig. 3. This feedback happens naturally, i.e., without requiring any modification at the application level, which is thus unaware of the L3 dynamics. In turn, changes in the L7 traffic matrix translate into different loads at L3, which possibly triggers a new update of the source routes by TE, closing the feedback loop.

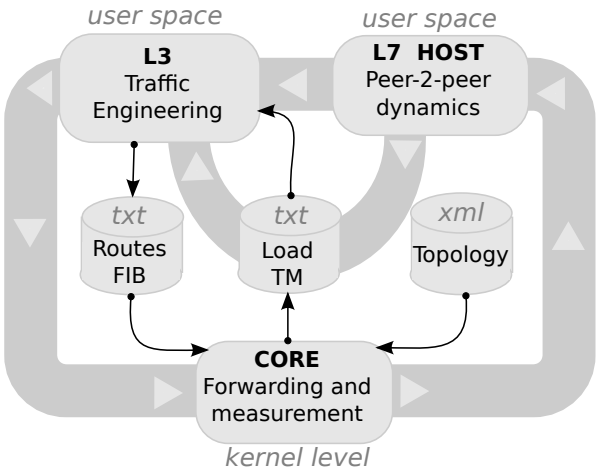


Fig. 3 ModelNet-TE high-level architecture

2.3 ModelNet-TE Minimum Congestion Load Balancing

The L3 TE algorithm we consider in this paper is the classic minimum congestion load balancing problem, probably first introduced in [19]. For each link l, we define a convex increasing function $f_l(\rho_l)$, where ρ_l is the load on link l, and the problem objective is to minimize the sum over all links of $\sum_l f_l(\rho_l)$. The rationale is that this function represents the congestion on the link, and that TE should strive to minimize the total congestion on the network. Convexity is intuitively justified by the fact that at higher loads, an increase in load generates more congestion than at lower loads. This objective function has become very popular, to the point that [66] defines TE as the procedure through which the network operator minimizes $\sum_l f_l(\rho_l)$.

Regarding the link congestion function $f_l(\rho_l)$ we chose the resulting mean queue size of a M/M/1 queue:

$$f_l(\rho_l) = \frac{\rho_l}{c_l - \rho_l} \tag{1}$$

The influence of the particular choice of $f_l(\rho_l)$ on different performance indicators is studied in [25]: as long as (1) is convex, increasing and diverges as ρ_l reaches c_l, the exact choice is unimportant in what regards path available bandwidth and link utilization.

The first input to the algorithm is the TM information. If every GW is ordered by an index, TE traffic matrix contains in its ij-th entry the mean traffic demand from node i destined to node j, usually called Origin-Destination (OD) pair. In addition to the TM, the algorithm also requires a set of paths that each OD pair may use. By specifying this set *a priori*, the resulting optimization problem is convex, which simplifies its solution (note that paths in this set are only *potentially* used in the solution, i.e., the amount of traffic sent along some paths may be zero). In particular, we bound the length of alternate paths $|A|$ with respect to the length of the shortest path $|S|$ found by Dijkstra, so that $|A| < |S| + 3$. In other words, we take alternate paths that exceed the shortest path by at most two hops, so to be able to route around a congested link or node, without incurring the load overhead of longer paths.

All in all, given the topology, the $f_l(\rho_l)$ associated to each link on the network, the traffic matrix and the paths that each OD pair may use, an algorithm is needed to find the amount of traffic that each GW should send along each path, so as to minimize $\sum_l f_l(\rho_l)$. With this respect, several choices are possible since the problem is convex. For instance, a classical approach to this kind of problems is the gradient descent method [18]. However, most of gradient

based algorithms include a parameter that controls convergence speed, which may be very tricky to assign. Although for each algorithm there exists a range for this parameter that makes the solution converge, in turn these values may result in slow convergence in certain situations. Conversely, larger values for the descent parameters may translate into faster convergence, but can possibly also trigger oscillations.

To avoid this reactivity-stability tradeoff, we resort to the use of so-called no-regret algorithms: in particular, ModelNet-TE implements the Incrementally Adaptive Weighted Majority (iAWM) algorithm [7], that presents the advantage of self-regulation. For instance, its convergence speed is automatically set, depending on previously observed values of $f_l(\rho_l)$. For lack of space, we invite the reader to [7] for a thorough algorithm description, and to [25] for extensive simulations results.

2.4 ModelNet-TE Scalability

It should not be forgotten that virtual nodes and virtual topology are ultimately emulated by the over a physical network, and that multiple virtual nodes are run on HOST machines. Hence, certain constraints must be met so to avoid that LAN capacity, CPU or RAM bottlenecks perturb the experiments. Our setups comprises 6 HOSTs and 1 CORE machines, each equipped with Intel Xeon CPUs (4 cores in hyper-threading running at 1.86GHz) and 4GB RAM, that are interconnected by an Ethernet Foundry EdgeIron 24G-A switch with 1 Gbps ports ports and a 4 Gbps back-plane.

Concerning CPU and RAM bottleneck, we can separately consider CORE and HOSTs machines. First, the CORE machine runs the forwarding and optimization engines: the first is highly efficient as implemented at kernel-layer, while the efficiency of the second depends on the TE algorithm implemented (and in our setup, it was never the bottleneck). Second, we experimentally verified that each HOST is able to run up to 35 P2P clients without incurring CPU penalties (i.e. CPU idle time was always higher than 20%): hence for instance, with 6 machines, we can build overlays whose size reaches $N_P = 200$ P2P clients (which is also a reasonable swarm size for both file-sharing and live-streaming, cfr. Sec. 3.2). Notice that we are considering much more conservative settings than, e.g., what usually considered in PlanetLab (for instance, [48] limits the overlay size on PlanetLab to 160 peers running on machines that report at least 5% idle CPU time). All in all, CPU bottleneck limitations are easy to get around, e.g., by increasing the number of HOST machines (to scale up the size of the L7 overlay), or by upgrading the raw computational power of the CORE (to scale up the size of the L3 network).

Rather, we point out that the number of P2P application instances that can be run on a single HOST machine also depends on the emulated VN uplink $C_{U,i}$ and downlink $C_{D,i}$ access capacities, as these translate into constraint on the physical HOST capacity (in our scenarios, we verify that such safety constraints are met, see Sec. 3.2).

An even more stringent constraint applies however to CORE capacity: indeed, in ModelNet-TE each packet needs to traverse the core twice, and although packets are sent on virtual interfaces, they enter the CORE through the same physical interface. As emulation happens in real time, at any time the overall traffic sent by all HOSTs in the physical network (or, equivalently, by all VNs in the emulated network) must not exceed the capacity of the CORE as otherwise unwanted queuing and drop effects may arise in the physical LAN, perturbing thus the experiments. In other words, it must be ensured that the sum of uplink and downlink traffic does not exceed the CORE capacity, translating into $C_U + C_D < 1$ Gbps, where C_U and C_D represent the aggregated uplink $C_U = \sum_{i=1}^{N_{VN}} C_{U,i}$ and downlink $C_D = \sum_{i=1}^{N_{VN}} C_{D,i}$ capacities respectively. To this extent, from our measures we derive that maximum aggregated throughput generated by BitTorrent is 377Mbps while for WineStreamer is 140Mbps.

Our setup can therefore be considered conservative also with respect to bandwidth bottlenecks, since both applications generate an aggregate traffic which is lower than the 1Gbps threshold discussed above. From the above data, we can also infer that theoretically our testbed settings could scale-up by a factor of 2.5 and 7 respectively for BitTorrent and WineStreamer, already without any change in the LAN speed. Even larger testbeds could be obtained upgrading the LAN interconnections between HOSTs and CORE to a 10Gbit Ethernet[4].

Finally, notice that ModelNet-TE does not allow to run experiments on parallel. This is however a design decision, as the primary tool usage is intended for individual research groups, that can easily decide a scheduling to run experiments on series. Notice that this limitation also applies to large dedicated experimental infrastructures, such as Grid'5000, whose aim is instead to be shared among different groups. Further, we point out that there could rather be more drawbacks in case mutual experiments would be run in parallel on the same infrastructure, since the traffic of the different experiments may have unwanted mutual influence, affecting and perturbing the experimental results.

3 Scenario and methodology

We now describe the scenarios emulated in our experimental campaign, providing motivation and detailed information concerning our choice of (i) network topologies, (ii) TE al-

[4] At the same time, care should be taken in this case, as [47] experienced degradation of ModelNet precision for aggregate traffic exceeding 600Mbps, so that further testing would be needed in this case.

gorithm details, (iii) population models, (iv) P2P applications.

3.1 L3 Network

3.1.1 Topology

Irrespectively of the P2P application, we consider two network topologies: namely, (i) a realistic Abilene topology and (ii) a simplified pure overlay model.

Often indeed, network topology is not considered due to studies such as [4], showing the bottleneck to be sited *at the edge* of the network. However, the above assumption holds for scenarios with a majority of low-capacity access technologies, such as e.g., ADSL lines, whose upload capacity is significantly limited. Conversely, the above assumption may no longer hold in case of fast FTTx Internet access [58] (i.e., Fiber To The Curb/Home), which is in the agenda of all major developed countries, and that reinforces the need of studying more realistic network scenarios.

Second, signals of the fact that P2P (or other user-level applications) are already causing congestion to ISPs can be inferred by recent issues such as (i) the throttling of BitTorrent connections by Comcast in the US [17] or (ii) the throttling of Megaupload by France Telecom in Europe [38]. The above examples show that, actually, ISPs are *already struggling* with the amount of data in their networks as of today, i.e., even when FTTx represent a minority of access technologies.

We take into account the above observation while building an emulation scenario. Due to the scale of our testbed, and to the physical limits of the interconnection (i.e., Ethernet transceivers, switches back-plane, number of HOST described in Sec. 2.4), it is however clearly impossible to emulate a full speed Internet core. Rather, we observe that problems may arise when the aggregated traffic generated by the user may cause congestion in the network, and decide thus to *jointly* scale access and core capacities so to produce situations similar to [17,38]. Notice also that while, the BitTorrent vs Comcast case has already hit the media, P2P-TV application may represent a similar threat due to the forecast ed growth of Internet video [16].

The realistic network scenario we design is thus as in Fig. 4, with core links interconnected according to the well-known Abilene topology [3], comprising $N_R = 11$ routers spanning over the US country. In our scaled setup, we consider core links capacities in $C = \{5, 10\}$ Mbps and we model peer access capacity as loose symmetric FTTH equal to $C_{D,i} = C_{U,i} = 5$Mbps. Notice also that, though realistic, the Abilene topology is also a hard scenario for load balancing, since the level of path diversity may not always allow to route *around* congestion.

To better grasp the impact of the network topology, we compare the Abilene scenario with a simplified model (not shown in the picture) where all peers are interconnected in a star topology to a single network core router. No capacities or delay are emulated in the network core (but only at the access): hence, due to our physical setup, the backbone runs at 1 Gbps switched Ethernet speed (which is much faster than the Abilene case, and where congestion never arises). Still, in this scenario we may enforce realistic access latencies, depending on the population model (see Sec. 3.2.1).

3.1.2 Traffic Engineering

We now discuss some implementation details of the iAWM algorithm described in Sec. 2.3, notably the timescale at which the algorithm is run. Let us recall that one of the inputs to the algorithm is the traffic matrix (TM), defined as the amount of aggregated VN traffic each GW node exchanges with each other.

In ModelNet-TE, the TM is sampled over windows of w seconds ($w = 1$ in our case), and ModelNet-TE can perform simple operations[5] (e.g., average, standard deviation, maximum, etc.) over W consecutive time windows. Then, after W consecutive windows, these demands are exported from the kernel to the TE algorithm (see Sec. 2.2). For the experimental campaign reported in this paper, we set $W = 30$, and run iAWM periodically after W windows, to set the new routing tables. Notice that the resulting timescale of the L3 traffic engineering decisions is on the order of 30 seconds (which is comparable with the order of the L7 timescale, as we describe in the next section).

3.2 L7 P2P Applications

At L7, we build realistic scenarios by considering heterogeneity in the (i) class of P2P applications and (ii) peer population models.

We select two P2P applications, namely BitTorrent [14] and WineStreamer [13, 34], that offer heterogeneous services and have thus a rather dissimilar design. Indeed, BitTorrent and WineStreamer are rather diverse in their constraints (i.e., elastic file-sharing vs minimum rate live-streaming), architectural choices (i.e., TCP vs UDP) and trading logic (i.e., rarest-first vs playout-deadline based). Yet, these applications also share some similarities (i.e., both are built on an unstructured an mesh overlay, with each peer optimizing its neighborhood by preferring high-bandwidth peers) that are a natural result of the evolution of the Internet P2P ecosystem, following the good performance these choices have exhibited [20, 33].

[5] The support for different operations simplify the implementation of different algorithms, that may rely on different inputs (e.g., average for iAWM [7] or maximum for [42]).

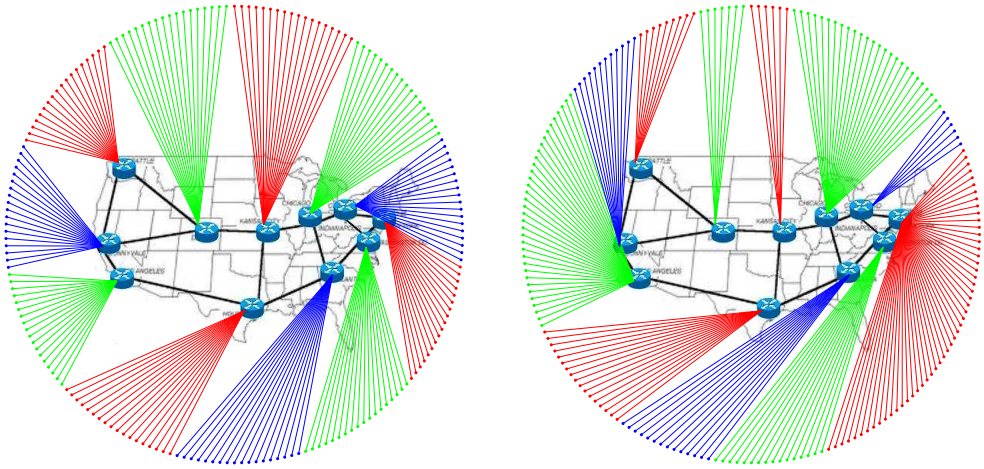


Fig. 4 L3 Abilene topology and L7 swarm: uniform (left) vs skewed (right) population models.

For both applications, we emulate a flash-crowd scenario in which a single source initially provides content (i.e., a file, or a TV channel) to a swarm of $N_P = 200$ peers. Notice that this is a reasonable swarm size for file-sharing applications, that furthermore trades off between observation in [46, 67]: more precisely, [67] observes that only about 1% of the torrents have more than 100 peers, while [46] reports typical sizes of BitTorrent swarms to be around 300-800 peers[6]. As far as live-streaming is concerned [29] observes that the swarm size for the same channel also depends on the application (i.e., which reflects the application popularity rather than the popularity of the content itself), with swarms ranging from 500 peers in TVAnts to about 180,000 peers for PPLive for the most popular content. Hence, $N_P = 200$ can represent an highly popular channel over a mildly popular application, or a mildly popular content over an highly popular application.

For the sake of simplicity, we consider homogeneous swarms capacities: notice that the effect of heterogeneous swarms with multiple capacities are well-known [32] from a pure L7 standpoint, and may be worth investigating from a joint L7/L3 viewpoint as future work.

3.2.1 Population model

Irrespectively of the P2P application, we may consider different swarm population models. In the Abilene topology of Fig. 4, each router acts as access router for several peers of the network: since the Abilene network comprises $N_R = 11$ nodes, and since we emulate $N_P = 200$ peers swarms, on average there are about 20 peers per node. In both cases, swarms initially have a single source located in Kansas City (in the middle of US).

[6] This may be due to the fact that [67] exhaustively explores *all* torrents, while observation in [46] are limited to a smaller torrents catalog.

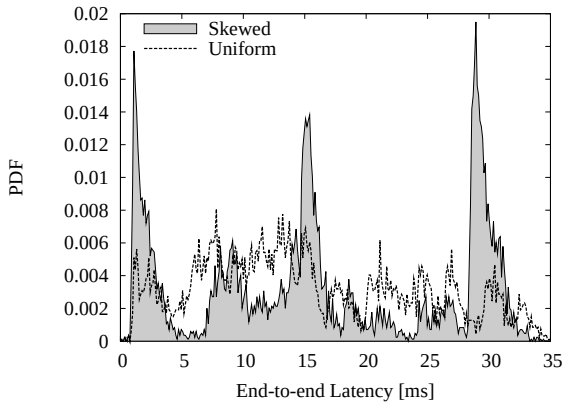


Fig. 5 Uniform vs skewed population models: end-to-end latency distribution.

However, while emulation studies usually uniformly distribute peers in the network (e.g., by spreading peers at random over PlanetLab nodes), we argue that peer population is more likely to reflect the actual human population in the real world. As the Abilene network spans across the US, we consider US cities of Abilene PoP and distribute peers to routers proportionally to the population of the corresponding urban area [63].

The skew in the population distribution translates into a more clustered swarm population, where several peers (users) may be found behind the same router (city). In turn, this also affect the distribution of the end-to-end[7] latencies, as peers are now more likely to be close.

[7] Notice that edge-to-edge latencies are measured between any pair of gateways GW (or IP routers), taking into account the physical distance between US cities. End-to-end latencies are emulated by additionally taking into account the local loop network beside the access GW [36].

The difference in the uniform vs skewed population models is pictorially represented in Fig. 4, and the corresponding distributions of edge-to-edge latencies are reported in Fig. 5. Latency distribution shows the impact of skewed peer population, as three peaks clearly arise: these correspond to (i) low delay for nearby communication (i.e., behind the same GW or confined in the east cost, west cost, or mid US), (ii) moderate delay for mid-range communication (i.e., between east cost and mid US, or west cost and mid US), and (iii) high delay for faraway communications (i.e., between both coasts). Notice also that high delay PDF peak is pronounced, as the majority of US inhabitants can be found along the eastern and western US coasts. Conversely, uniform peer distribution yields to a completely different latency distribution, which is unrealistic with respect to actual measurements carried on in measurement projects such as [64].

3.2.2 P2P Filesharing: BitTorrent

For file-sharing, we use the Python version of BitTorrent-4.0.0-GPL. In file-sharing, the main aim is to let all peers in the swarm download the content in the shortest possible time. Notice that the BitTorrent version we consider employs TCP at transport layer (L4). While we are aware that recently BitTorrent introduced a new application-layer transport protocol based on UDP at L4 [53], we choose TCP filesharing since the new protocol is for the time being implemented only in a specific client (namely, μTorrent, that is estimated to account for varying ratio of BitTorrent clients 15 [46]-60 [67]%. Besides, our attention is here more focused on the interaction of P2P applications and L3 network, rather than to the performance of BitTorrent under a new congestion control paradigm, which we instead investigate in [61].

We point out that providing a survey of BitTorrent is out of the purpose of this work, for which we refer the reader to [14, 32]. Here, we only mention that BitTorrent peers establish and maintain a limited number of connections, over which they download small portions (or chunks) of the file they are interested in obtaining. Periodically (every 20 seconds), peers rank their connections depending on the download rate, keeping only the best connections ("chocking" the least performing ones), and optimistically trying to discover new potential good peers (nicknamed as "optimistically unchoking" in BitTorrent, and performed every 30 seconds). To avoid free-riding, BitTorrent enforces reciprocation of content exchange (tit-for-tat) and, to avoid resources hotspot, BitTorrent peers try to equalize the chunk availability in the system by downloading the rarest chunk first. The timescale of the L7 application dynamics is on the order of 20 seconds, thus comparable with L3 dynamics.

In a flash crowd scenario, BitTorrent peers behave differently depending on whether they are leecher or seed. Initially, the seed is the unique source of a 100 MBytes file: hence, at the very beginning we expect most of the traffic to be originated from a single VN (i.e., the seed). However, as chunks start spreading in the swarm, exchanges between leechers become prominent, until the seed contribution is no longer necessary [32]. Hence, the traffic matrix offered at L3 by L7 will change during the whole experiment duration, so that the system evolves without ever reaching a stationary state.

3.2.3 P2P-TV: WineStreamer

For live-streaming, we use WineStreamer, an application developed in the context of the FP7 Strep Project on Network Aware P2P Applications over Wise Networks (Napa-Wine) [34]. In live-streaming the main aim is to let all peers in the swarm receive the minimum stream rate (similarly to video-on-demand), and to minimize the playout lag with the source (additionally to video-on-demand).

WineStreamer belongs to the last generation of live-streaming applications, and is able to take informed decisions with respect to the network state [21]. Knowledge of the network state is commonly nicknamed as "network awareness", and can either (i) be measured by the application or (ii) be achieved with ISP cooperation. Examples of (i) include preferring nearby peers to faraway ones based on RTT or IP hop-count measurements, or preferring high capacity peers by means of bandwidth measurements, etc. [21]). An examples of (ii) is represented by IETF ALTO [5], defining ISP servers that acts as "oracles" and participate in the P2P peer selection process with informed suggestion on good candidate peers.

In this work, we only consider L7 measurements performed by the application itself, and turn off WineStreamer ALTO capabilities. Notice that, due to chunk transmission duration over ADSL lines, we expect the bandwidth-aware [20] peer selection criterion to prevail over latency-aware [11] or power-aware [52] (i.e., the ratio of bandwidth over latency) peer selection criteria. In other words, as for slow ADSL peers the chunk transmission time exceeds the propagation delay, in order to keep the overall system latency low, the ability to find high-capacity peers prevails over the ability to find nearby peers [55].

In all of the following experiments we stream a 600 Kbps video at 25 fps encoded with H264. In the video diffusion, we map every video frame to a single chunk (while several audio frames are grouped together in a single chunk to reduce the overhead). Video stream is not decoded at destination, but is discarded to avoid too many concurrent blocking IO calls; however, we log chunk-level arrival patterns to later evaluate the quality of user experience.

Again, providing a survey of WineStreamer is out of the purpose of this work, for which we refer the reader to

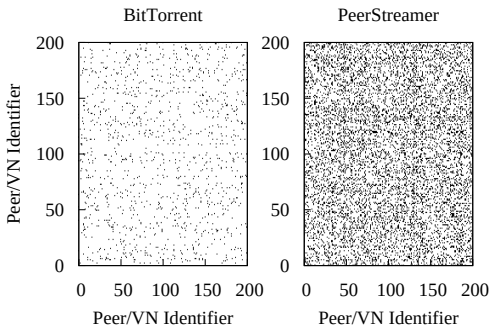


Fig. 6 L7 traffic demands (Abilene topology, skewed population, IP routing): Swarm adjacency matrix for BitTorrent (left) and WineStreamer (right)

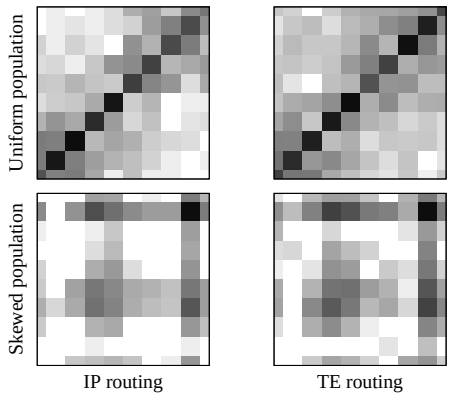


Fig. 7 L3 traffic demands (Abilene topology, BitTorrent file-sharing application). Plots represent uniform (top) vs skewed (bottom) population models, with IP shortest path (left) and TE multi-path (right) routing.

[13, 34]. Rather, here we highlight the complementarity of WineStreamer with respect to BitTorrent. On this regards, we point out that the application exploits UDP and, though it implements a simple retransmission mechanism, the version we use in the testbed does not implement any form of congestion or flow control – hence, it sends out chunks at full speed. Moreover, chunk size is smaller than the one normally used in BitTorrent: as the scheduler performs decisions at a higher rate, hence we expect the P2P neighborhood to be more dynamic. Due to the use of UDP and to the minimum stream-rate requirement, WineStreamer is therefore a non-elastic application, with stringent near real-time requirements, unlike BitTorrent. Also, differently from BitTorrent, WineStreamer source is always providing new content to the swarm, at the same average rate, so that the system tends to a stationary state (although with a varying neighborhood).

4 Experimental Results

In this section, we report results of our experimental campaign, adopting two complementary viewpoints. First, we analyze the traffic that the P2P applications induce on the L3 network. Then, we analyze the impact that each simplistic vs realistic parameter choice has on the quality that the user perceives.

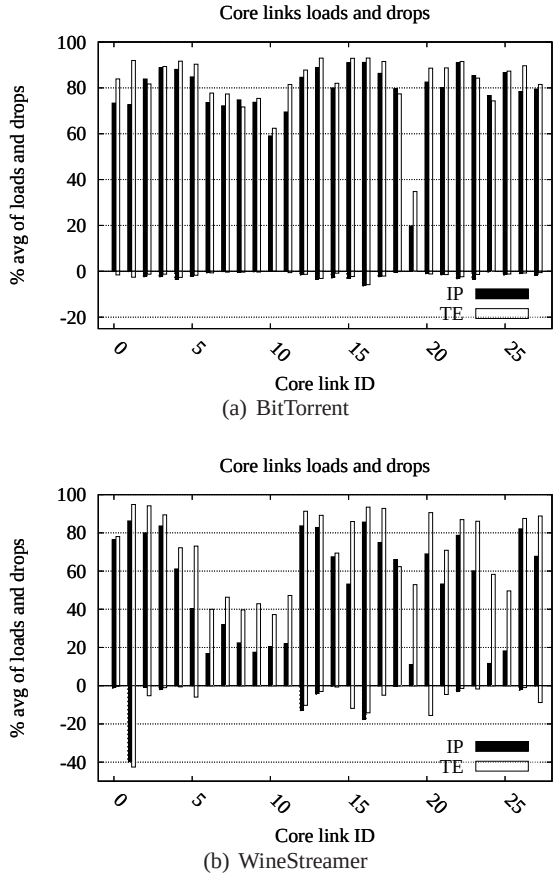
4.1 Traffic demands and link load

Let us investigate the traffic demands that P2P traffic induces over the whole network, and how these demands translate into individual link load. A pictorial representation of the traffic demands, at L7 and L3, is shown in Fig. 6 and Fig. 7 respectively.

Fig. 6 exploits a matrix representation to compare traffic demands of the applications, measured over the whole experiment duration, where black points indicate a chunk exchange between two peers. Already at a first glance we can observe the difference in matrix density: WineStreamer behavior is much more “loquacious”, while BitTorrent contacts a lower number of nodes.

Augmenting the same kind of representation with gray levels proportional to the volume of exchanged data, we analyze load on L3 induced by the L7 application. While Fig. 6 reported host-to-host communication with a binary semantic (i.e., the matrix representation reports a black dot if two peers communicate during the experiment), Fig. 7 reports the router-to-router traffic matrix (i.e., where the intensity of the traffic exchanged between any router is represented with gray colors). Considering for the sake of example only the BitTorrent application, we vary the L7 population model and the L3 routing, and depict the L3 TM in Fig. 7. Comparing top and bottom rows, one can gather the difference between uniform (top) and skewed (bottom) population models: data exchange in the skewed population is much more concentrated around few points (i.e. GW of large US cities as New York or Los Angeles) while in the uniform case, chunks are more evenly exchanged. Especially, for the uniform population the traffic concentrate on the matrix diagonal, so that peers prefer to exchange with peers behind the same router; conversely, in the skewed population cluster forms involving routers sited in regions where users (and, hence, peers) are more numerous.

Comparing instead left and right columns, one can gather the difference between IP shortest-path (left) and TE multi-path (right) routing: as expected, traffic is more spread out under TE load balancing, which is especially visible in the case of skewed population.



(a) BitTorrent

(b) WineStreamer

Fig. 8 Network link load (positive y-axis) and packet loss rate (negative y-axis) , BitTorrent (top) vs WineStreamer (bottom), in the IP vs TE cases, Abilene topology and skewed population

Performance of L3 network are reported in Fig. 8, showing link load and packet loss rate of individual links in the backbone, for both P2P applications and comparing IP vs TE routing. Notice that link load is reported on the positive y-axis, while packet loss rate is repored on the negative y-axis. For the sake of simplicity, we only consider a scenario with realistic Abilene topology, 5Mbps core links, and skewed population. The striking differences that the L7 traffic matrix exhibited in Fig. 6, also entails different impact on L3, which can easily be explained. Notice that while in this work we focus mostly on the performance implication on L7 application of the TE decision at L3, in our technical report [24] we provide further insights on the TE innerworking (such as paths and paths-probability) for the interested reader.

Consider first the BitTorrent case in Fig. 8(a): as the application version we use employs TCP at L4, peers attempt at fully utilizing their uplink bandwidth (provided that they have enough chunk requests). In turn, this yields to a significant utilization of core link, which are mostly above 70% average utilization. Notice also that important links, such as those serving the gateways where the source is located, are not facing severe congestion (i.e., higher load or losses), which is again due to TCP congestion control. Hence, TE only provides marginal changes in the traffic matrix, increasing by about 2% the fairness of the link utilization (measured with Jain fairness index $(\sum_{i=0}^{N} \rho_i)^2/(N \sum_{i=0}^{N} \rho_i^2)$ with ρ_i load on the i-th link).

Conversely, in the WineStreamer case of Fig. 8(b), we notice that load is unevenly distributed, with some links being lightly loaded and other carrying significant traffic amount, and experiencing non marginal losses. This striking difference is due to (i) chunk scheduling dynamics and (ii) the transmission of chunks as spurts of back-to-back packets over UDP. As for chunk scheduling, peers need to receive content over small time windows, and as new content is constantly being produced at the source, the source is possibly overwhelmed by chunk requests. Moreover, as no congestion control is implemented by the application, the chunk transmission process can be very bursty, so that aggregated traffic load is no longer smooth as in the TCP case, but is more likely to cause drops on some links. It must be said that Fig. 8(b) depicts a severe congestion scenario due to narrow 5Mbps link, that however let us better grasp some effects: notice indeed that while it can be seen that TE manages to equalize link level load to some extent (fairness increases by about 6%), TE efforts are not sufficient in this severe congestion scenario. Worse yet, use of multi-path TE can seldom overload links (that were only mildly loaded under IP routing), further inducing losses (that were absent under IP). This behavior can be induced by the two uncoordinated control policies at L3 and L7, that happen independently and at the same timescale, and that we can exemplify as follows. Assume that L7 application decides to route content toward a peer whose path is lightly loaded and has never experienced losses. Assume further that, roughly at the same time, L3 realizes that links along the same path are lightly loaded, and decides to reconfigure the FIB. Now, what happens is that links along that path will experience a sudden, unexpected, load increase – that in case of live-streaming will be exacerbated by the use of full-rate UDP chunk spurts.

Notice also that Fig. 8(b) suggests that not all the network capacity is fully utilized, while a swarm of the same size in Fig. 8(a) was able to use more resources. This hints to the fact that peers in the WineStreamer application could potentially serve more other peers, thus offloading the source and further ameliorating system performance. Similar observations lately led to the development in WineStreamer of a dynamic aggregated congestion control over UDP, named Hose Rate Control (HRC) [12], showing thus that ModelNet-TE can provide an invaluable help to P2P application developers[8].

[8] HRC was however not available at the time of the experimental campaign.

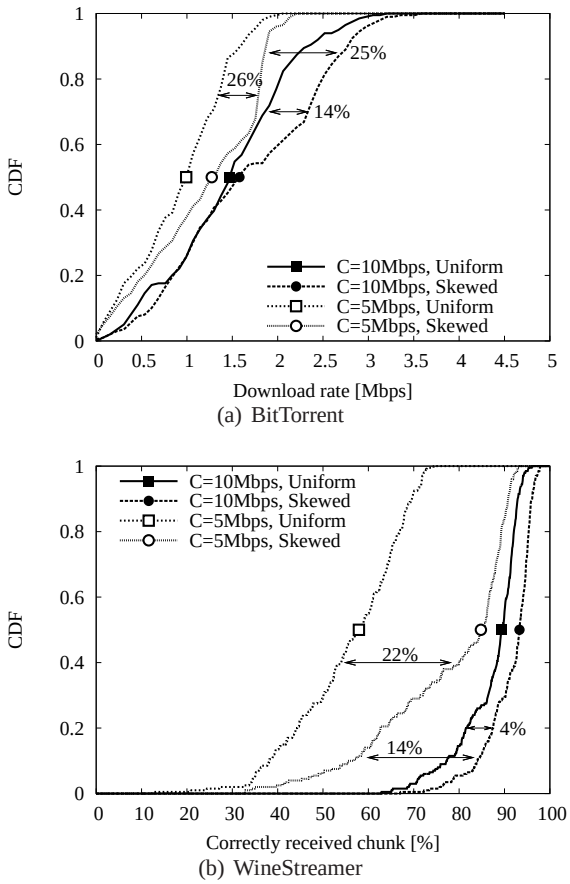


Fig. 9 Impact of (i) uniform vs skewed population model and (ii) core link capacities C=5Mbps vs C=10Mbps. Arrows are used to highlight the percentage of difference between the *mean values* of the corresponding CDF curves.

4.2 Impact of population model and network capacity

We now focus on the performance of L7 applications, considering (i) the download rate of BitTorrent peers and (ii) the percentage of correctly received chunks for WineStreamer. We argue these to be the most relevant metrics that furthermore intuitively express the quality of user experience: as for (i), download rate is tied to the system efficiency and to the time it takes peers to complete their download; as for (ii), the video quality is badly affected by chunks that are received after the playout deadline (e.g., due to queuing delay at L3) or that are only partially received (e.g., due to packet loss at L3 that WineStreamer retransmission mechanism failed to recover). Both metrics are evaluated over windows of 10 seconds (i.e., the same timescale employed by BitTorrent to rank the active peer set for the choke operation). In the following, we report results gathered over 5 different runs for any given experimental settings.

We first consider the impact that the capacity C on core links and the peer population model have, and depict the cumulative distribution function (CDF) of the download and chunk reception rates, measured over the whole swarm in Fig. 9 (for the time being, we fix the topology to Abilene and limitedly consider IP shortest path routing, whose impact we instead assess in Sec. 4.3).

Consider the population distribution first. The general consideration is that skewed population is beneficial in that, provided that the application is aware of latency or bandwidth[9], it can establish neighboring relationships with peers attached to the same GW router, thus confining traffic at the edge and avoiding narrow core links.

Focusing on BitTorrent clients, we see that lowest download rates are achieved by peers on the C=5Mbps uniform population scenario: then, notice that a roughly equivalent performance gain can obtained by either (i) doubling the capacity under the same population model or (ii) considering a skewed population model at the same capacity level. Notice indeed that the average download rate increases by 25% and 26% respectively, as reported in Fig. 9-(a).

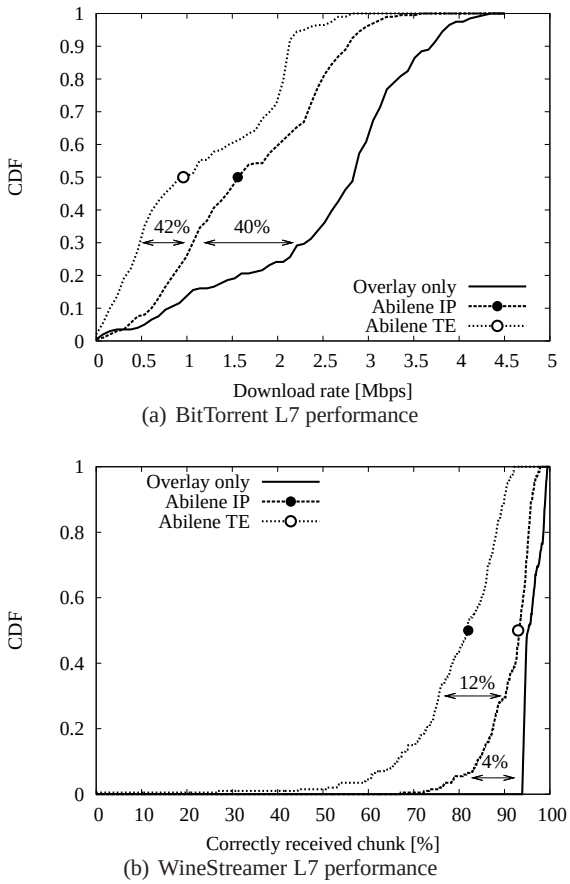
Considering WineStreamer clients, we see that the impact of the population model remains considerable, although in this case core links capacities plays a determinant role due to streaming constraints. Indeed, considering the underprovisioned C=5Mbps scenario in Fig. 9-(b), on average 22% more chunks are received under a skewed population model with respect a uniform one. Yet, change in the population model are not sufficient, as the percentage of received chunks for C=5Mbps is still low for some peers (those that were serviced by the underprovisioned link exhibiting up to 40% packet losses in Fig. 8), while situation improves considerably for C=10Mbps.

4.3 Impact of network topology and multi-path routing

Let us now consider the impact of the network topology and routing policies, where for the sake of simplicity, we only consider a skewed population model. To assess the impact of the topology, we compare a pure overlay model against a well provisioned Abilene network with core link capacity equal to C=10Mbps. While it is straightforward to foresee that on a pure overlay model both applications will perform better, as we removed any topological bottleneck, it would not be possible to quantify this gain without ModelNet-TE.

As expected, we see that in the case of BitTorrent the performance achieved on a pure overlay model can be significantly higher with respect to the Abilene case. This is because once in network capacity bottleneck are removed, TCP can make better use of the access capacity: on average

[9] Since TCP is advantaged by smaller RTT, application preferring high-bandwidth peers will also likely prefer nearby peers. Even for application such as PPLive, using UDP at L4 and measuring transfer rates at L7, we experimentally verified that bandwidth preference induces a clustering of nearby peers [54].



(a) BitTorrent L7 performance

(b) WineStreamer L7 performance

Fig. 10 Impact of (i) Abilene vs Pure overlay network topology and (ii) IP single-path vs TE multi-path routing. Arrows are used to highlight the percentage of difference between the *mean values* of the corresponding CDF curves.

BitTorrent can download at a 40% faster rate in a overlay-only scenario with respect to shortest-path IP routing. Conversely, as the video stream sent by WineStreamer has a fixed bitrate, and since the capacity of the network is provisioned to transport almost all that traffic, the difference between the overlay-only vs IP routing is limited to 4% (respectively, 95% vs 91% of chunks are received on average).

Finally we analyze the influence that TE techniques can have on P2P systems, by contrasting IP vs TE routing on the Abilene topology: counter-intuitively, we see that TE may lower the performance of both applications.

In BitTorrent, this can be explained by the fact that, recalling Fig. 8(a)-(a), nearly all links already operates at a regime close to their capacity. Hence, as TE reroutes the traffic along possibly longer paths, it extends the number of traversed link for each packet: thus, while TE balances the load more evenly across links, it may in turn raise the global network load. Second, since TE operates on a per packet basis, it may alter TCP congestion control: indeed, TCP transmission mechanism is self-clocked on the basis of RTT estimation. As each packet may traverse different paths, of different lengths, with different levels of congestion, this can significantly affects the RTT estimate. Second, as packets can now arrive out of order, this may possibly trigger spurious TCP retransmissions [39]

WineStreamer is a network aware application, that already executes measurement on the underlying L3 network, so to perform informed peer selection and scheduling decisions: in this case, periodical changes in the network topology due to TE are not beneficial to the already complex L7 algorithms. Recalling Fig. 8, we see that due to the highly bursty chunk transmission process, it seldom happens that independent L7 and L3 decisions increase the loads on some link. However, since this is the result of two uncoordinated decisions, it is impossible to blame a single actor between L3 or L7, as problems arise from the interplay of both. In fact, both L3 TE and L7 algorithm take decisions on the assumption that, respectively, traffic and network topology are static: thanks to ModelNet-TE we see that when this assumption no longer holds, unexpected phenomena may arise.

5 Related work

Two bodies of work are related to ours. On the one hand, there is work focusing on the experimental evaluation of P2P applications by means of testbeds [12, 15, 51] or large scale-experiments [37, 46, 65], possibly stemming from hybrid simulative/experimental approaches [1, 6, 10, 26, 27, 41]. On the other hand, work exists that focuses on the interaction of the overlay and network layers [30, 31, 35, 45, 50, 56, 68].

5.1 Experimental evaluation

Performance evaluation of P2P system has often involved simulation approaches, due to the relatively rapid prototyping of new applications on the one hand, and on possibility to inexpensively validate the application performance on of a controlled tool on the other hand. For a more in-depth discussion of simulators tools for P2P networks, we invite the reader to [43].

At the same time, as simulative environments forcibly incurr in a number of simplification of the reality, the performance gathered by means of simulation still need to be validated against experiments of real applications prototypes. In recent P2P research, we observe that basically three trends emerged: the first employs an hybrid simulative/experimental approach [1, 6, 10, 26, 27, 41], the second controlled testbeds [12, 15, 51], and the latter world-wide infrastructures [37, 46, 65].

In the *hybrid simulative/experimental approach*, researchers are offered a development toolkit that eases the transition from simulation to network testbeds, so that the same code can be reused for both simulation and experiments. Example of this line of effort include PlanetSim [27], PeerSim [41] and its latest evolutions, Protopeer [26], Kompics [6], ns3 [1] (whose scope goes beyond that of P2P performance evaluation) and Oversim [10] (the latter a P2P toolkit for the well known Omnetpp [2] simulator).

The issue is that [6, 10, 26, 27, 41] forces the user to develop new applications using a specific toolkit, that may sometimes be too constraining. Notice indeed that this means that existing applications should be re-implemented in the framework – a possibly overwhelming effort that may counter the success of the above tools. Also, such toolkits cannot clearly be used to evaluate very popular but closed-source and proprietary applications such as Skype (VoIP), PPLive or SopCast (P2P-TV) or uTorrent (filesharing the popular implementation of BitTorrent, that lately become closed-source). Finally, notice that while these toolkits offer the ability to simulate the system, they generally fall short in offering experimental evaluation capabilities as well.

Ultimately, this implies that a great effort is still required to evaluate the developed prototype in controlled testbeds [12,15,51] or world-wide infrastructures [37,46,65]. Hence, as far as experimental evaluation is concerned, this lead us to considering two main class of methodologies.

Controlled experiments are run in dedicated infrastructures, such as Grid5000 [15, 51] or ad hoc testbeds [12], where clusters of several coordinated machine, which are usually connected through LAN, run P2P clients. As these infrastructures are general purpose (i.e., not tailored for network experiments) experimental setup can be a burden. Besides, latency and packet drops must be artificially enforced by external tools and it is impossible to carry out studies on L3/L7 interaction. Nonetheless, [51] concludes that BitTorrent experiments performed on cluster are realistic and that wide area network latency and packet losses impact for less than 15% of the download time. If we agree that this precision is realistic enough for elastic file-sharing applications, such error margin cannot be tolerated for interactive live-streaming applications – where chunk losses or delayed arrivals heavily impact the quality of experience.

World-wide infrastructures such as PlanetLab [49] and OneLab [44], have long been used to test and benchmark distributed applications [37,46,65]. Currently, PlanetLab provides about 1000 nodes at 500 different sites scattered around the globe. Yet, one of the perception is that PlanetLab is not suitable for P2P experiments since it is composed mainly by high capacity nodes and few DSLs [60]. Another potential problem arises from the fact that access to nodes is shared among users, each of which is getting different "slices": yet, as multiple concurrent experiments can be run on the same infrastructure, there is no control on CPU load (for instance, [48] points out that "Since most Planetlab machines are usually over-loaded, we limit the overlay size to 160 peers running on machines that report at least 5% idle CPU time.") other traffic (e.g., two P2P experiments running concurrently, or measurement between PlanetLab nodes) that can both alter experiment results[10].

5.2 Routing layer interaction

The study of the interaction of several routing layers is instead motivated by findings in [45, 50, 56]. Briefly, while selfish routing may be highly unoptimal in general settings [45, 56], in practice it performs reasonably well in Internet-like environment [50], which justify and confirms its interest. At the same time, an important observation is that local optimization entailed by selfish overlay routing may counter actions taken by the underlaying network, overall resulting in poor system stability.

As a consequence, there has been a recently increased attention [30, 31, 35, 68] on the potential issues on uncoordinated, uncontrolled interaction of two routing paradigms. Different studies consider different levels of interaction such as a P2P overlay network and the underlying IP network routing [31,35], IP routing and the underlying MPLS/GMPLS network [68], multiple P2P overlays routing, coexisting on a given underlay network [30].

5.3 Advances with respect to the State of Art

This work extends and complements both bodies of work. On the one hand, though we use an experimental methodology, to the best of our knowledge P2P traffic has been studied with a pure overlay model [12, 15, 22, 32, 37, 48, 51, 57], neglecting thus the mutual impact with lower-layer network. On the other hand, most of the work focusing on interaction between different routing layers exploits a Game Theoretical approach [23, 35, 68], with a simulative approach limitedly used in [31].

As such, the research community still lacks more realistic and practical studies, which is precisely what we address in this work: thanks to the ModelNet-TE framework, we encompass both classes of work, by proposing the first study of routing layer interaction that exploits an experimental methodology. Notice instead, that our work is orthogonal to hybrid simulative/experimental approach [1, 6, 10, 26, 27, 41], in that ModelNet-TE transparently work with any working P2P application, irrespectively of how the prototype has been engineered/developed.

[10] Notice that this should change with the recent ability in OneLab to reserve resources similarly to what happens in Grid5000

Table 1 Scale of different P2P experiments (this work in bold)

Ref.	Testbed	Nodes	Nodes/ Machine
[15]	Grid5000	10000	100
[12]	Own testbed	1000	5
[22]	PlanetLab	400	1
[48]	PlanetLab	320	32
[51]	Grid5000	300	100
[57]	PlanetLab	280	1
	ModelNet-TE	**200**	**35**
[48]	PlanetLab	160	1
[47]	Modelnet	80	8
[32, 37]	PlanetLab	41	1

Moreover, ModelNet-TE tool sits between controlled and wild testbed infrastructures, trading off between the realism of PlanetLab, and the scalability of Grid5000, and adding the control over the network topology and routing algorithm. Yet, the scale of the experiments that can be performed is not compromised: to prove this, Tab. 1 reports a comparison of different closely related work, highlighting the scalability aspects for the methodologies discussed so far.

Notice that most works scale up to a few hundreds peers, i.e., the same order of magnitude of our ModelNet-TE experiments, confirming the validity and usefulness of the tool. Only two notable exceptions push the experiment scale to 1,000 [12] and 10,000 [15], trading off experimental scale with simplicity of the experimental setup.

6 Conclusions

This paper presents ModelNet-TE [40], a open source emulation tool with Traffic Engineering (TE) capabilities: building over the original ModelNet core, which only offers standard IP routing, we added the support of TE and implemented a multi-path load balancing algorithm. At the same time, our purpose was to design a flexible tool, that can be easily integrated with many other TE algorithms beyond the one that we provide.

As a case study, we use ModelNet-TE to analyze the interaction between Traffic Engineering at the network level (L3) and end-to-end control policies implemented at the application layer (L7) by P2P application such as BitTorrent (one of the most popular file-sharing applications) and WineStreamer (a mesh-based network-aware live-streaming application). We performed a thorough experimental campaign, considering several parameters (such as topology, core link capacities, IP vs TE routing, peer population models, etc.) in the scenario definition. To gather a comprehensive understanding of the system dynamics, we express performance in terms of both network-centric and user-centric metrics: at L3, we measure link load and losses and at L3 we measure the BitTorrent download rate and WineStreamer chunk reception rate.

Our results not only validate ModelNet-TE as a complementary tool to test P2P applications in realistic environment, but also yield several interesting insights on L7/L3 dynamics. Summarizing our main findings, we have that overlay-only models yield an overly optimistic evaluation of P2P application: while the overlay-only model applies to toady's ADSL access, we have increasing evidence [17, 38] that in the near future bottlenecks may no longer sits at the user access link. Second, we observed that the population model heavily impacts overlay performance, as its impact can be of the same order of magnitude of in-network capacity limitations: hence, the ability to localize part of traffic behind the same access gateway, e.g., by means of IETF ALTO servers, seen an interesting option to offload the network and ameliorate the user experience. Third, we see that TE can noticeably worsen L7 performance: this counter-intuitive results is due to the interplay of several factors, among which (i) the impact of per-packet load balancing on TCP performance, and (ii) the uncoordinated reconfiguration of the overlay and underlay networks for unelastic applications.

While this work attempts at analyzing a large spectrum of scenarios, it also leaves many points open. As far as the experimental results are concerned, for example, it would be interesting to assess whether the conclusions gathered in this paper are more general than the explored settings, i.e., if they continue to hold for different topologies, TE algorithms and P2P applications. As far as the tool itself is concerned, it would instead be interesting to further extend the scale of the achievable experiments, e.g., by allowing the newly introduced TE functionalities to work on multiple parallel CORES as supported by the original ModelNet core for shortest path IP routing.

Acknowledgments

This work was funded by FP7 STREP NAPA-WINE. Authors wish to thank Eugenio Alessandria and Luca Muscariello for their initial help on this work.

References

1. ns3 homepage. http://www.nsnam.org.
2. Omnetpp homepage. `http://www.omnetpp.org`.
3. Abilene network. `http://www.internet2.edu/network/`.
4. A. Akella, S. Seshan, and A. Shaikh. An empirical evaluation of wide-area internet bottlenecks. In *Proc. of the 3rd ACM SIGCOMM Conference on Internet Measurement (IMC'03)*, Miami, FL, USA, Oct. 2003.
5. Ietf ALTO. `http://datatracker.ietf.org/wg/alto/charter/`.
6. C. Arad, J. Dowling, and S. Haridi. Building and evaluating p2p systems using the kompics component framework. In *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*, pages 93–94. IEEE, 2009.

7. P. Auer, N. Cesa-Bianchi, and C. Gentile. Adaptive and self-confident on-line learning algorithms. *Elsevier Journal of Computer and System Sciences*, 64(1):48 – 75, 2002.
8. B. Augustin, X. Cuvellier, B. Orgogozo, F. Viger, T. Friedman, M. Latapy, C. Magnien, and R. Teixeira. Avoiding traceroute anomalies with paris traceroute. In *Proc. of ACM SIGCOMM Internet Measurement Conference, (IMC'06)*, Rio de Janeiro, Brazil, Oct. 2006.
9. B. Augustin, T. Friedman, and R. Teixeira. Measuring load-balanced paths in the internet. In *ACM SIGCOMM Internet Measurement Conference (IMC'07)*, pages 149–160. ACM, 2007.
10. I. Baumgart, B. Heep, and S. Krause. Oversim: A flexible overlay network simulation framework. In *IEEE Global Internet Symposium, 2007*, pages 79–84. IEEE, 2007.
11. R. Bindal, P. Cao, W. Chan, J. Medved, G. Suwala, T. Bates, and A. Zhang. Improving traffic locality in bittorrent via biased neighbor selection. In *Proc. of IEEE Distributed Computing Systems (ICDCS'06)*, Lisboa, Portugal, Jul. 2006.
12. R. Birke, C. Kiraly, E. Leonardi, M. Mellia, M. Meo, and S. Traverso. Hose rate control for p2p-tv streaming systems. In *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P'11)*, Kyoto, Japan, Sep. 2011.
13. R. Birke, E. Leonardi, M. Mellia, A. Bakay, T. Szemethy, C. Kiraly, R. L. Cigno, F. Mathieu, L. Muscariello, S. Niccolini, J. Seedorf, and G. Tropea. Architecture of a network-aware p2p-tv application: the napa-wine approach. *IEEE Communication Magazine*, 49, Jun. 2011.
14. Bittorrent home page. `http://www.bittorrent.com`.
15. S. L. Blond, A. Legout, and W. Dabbous. Pushing bittorrent locality to the limit. *Elsevier Computer Networks*, 55(3):541 – 557, 2011.
16. Cisco visual networking index: Forecast and methodology, 20102015. `http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf`.
17. Comcast discloses throttling practices bittorrent targeted. `http://www.wired.com/threatlevel/2008/09/comcast-disclos/`.
18. D. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.
19. D. Bertsekas and R. Gallager. *Data Networks*. Prentice-Hall, 1987.
20. A. da Silva, E. Leonardi, M. Mellia, and M. Meo. A bandwidth-aware scheduling strategy for p2p-tv systems. In *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P'08)*, Aachen, Germany, Sep. 2008.
21. L. Dai, Y. Cao, Y. Cui, and Y. Xue. On scalability of proximity-aware peer-to-peer streaming. *Elesevier Computer Communications*, 32(1):144 – 153, 2009.
22. C. Dale, J. Liu, J. Peters, and B. Li. Evolution and enhancement of bittorrent network topologies. In *In Proc. of ACM/IEEE International Workshop on Quality of Service (IWQoS'08)*, Twente, The Netherlands, Jun. 2008.
23. D. DiPalantino and R. Johari. Traffic engineering vs. content distribution: A game theoretic perspective. In *Proc. of IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009.
24. D. R. E. Alessandria, L. Muscariello. ModelNet-TE: An emulation tool for the study of P2P and Traffic Engineering interaction dynamics. Technical report, Telecom ParisTech, available at `http://www.enst.fr/~drossi/ModelnetTE/modelnet-techrep.pdf`, 2011.
25. Federico Larroca. *Techniques d'Ingénierie de Trafic Dynamique pour l'Internet*. PhD thesis, Telecom ParisTech, 2009.
26. W. Galuba, K. Aberer, Z. Despotovic, and W. Kellerer. Protopeer: Distributed systems prototyping toolkit. In *IEEE P2P'09*, pages 97–98. IEEE, 2009.
27. P. García, C. Pairot, R. Mondéjar, J. Pujol, H. Tejedor, and R. Rallo. Planetsim: A new overlay network simulation framework. *Software Engineering and Middleware*, pages 123–136, 2005.
28. Grid5000 home page. `http://www.grid5000.fr/`.
29. A. Horvath, M. Telek, D. Rossi, P. Veglia, D. Ciullo, A. G. da Rocha Neta, E. Leonardi, and M. Mellia. Network awareness of p2p live streaming applications: a measurement study. *IEEE Transactions on Multimedia*, 12(1):54–63, Jan. 2010.
30. W. Jiang, D.-M. Chiu, and J. C. S. Lui. On the interaction of multiple overlay routing. *Elesevier Performormance Evaluation*, 62(1-4):229–246, 2005.
31. R. Keralapura, C.-N. Chuah, N. Taft, and G. Iannaccone. Can ISPs take the heat from overlay networks? In *In Proc. of ACM Workshop on Hot Topics in Networks (HotNets-III)*, San Diego, CA, USA, November 2004.
32. A. Legout, N. Liogkas, E. Kohler, and L. Zhang. Clustering and sharing incentives in bittorrent systems. In *Proc. of ACM SIGMETRICS*, San Diego, CA, USA, Jun 2007.
33. A. Legout, G. Urvoy-Keller, and P. Michiardi. Rarest first and choke algorithms are enough. In *Proc. of ACM SIGCOMM Internet Measurement Conference, (IMC'06)*, Rio de Janeiro, Brazil, Oct. 2006.
34. E. Leonardi, M. Mellia, A. Horvath, L. Muscariello, S. Niccolini, and D. Rossi. Building a cooperative p2p-tv application over a wise network: the approach of the european fp-7 strep napa-wine. *IEEE Communications Magazine*, 46(4):20 –22, april 2008.
35. Y. Liu, H. Zhang, W. Gong, and D. Towsley. On the interaction between overlay routing and underlay routing. In *Proc. of IEEE INFOCOM*, Miami, FL, USA, Mar. 2005.
36. H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. Anderson, A. Krishnamurthy, and A. Venkataramani. iplane: An information plane for distributed services. In *Proc. of 7th USENIX Sympsium on Operating Systems Design and Implementation (OSDI'06)*, Seattle, WA, USA, Nov. 2006.
37. P. Marciniak, N. Liogkas, A. Legout, and E. Kohler. Small is not always beautiful. In *Proc. of 7th International workshop on Peer-To-Peer Systems (IPTPS'07)*, Tampa, FL, USA, Sep. 2008.
38. Megaupload accuse orange de ralentissements. `http://info.france2.fr/sciences-tech/megaupload-accuse-orange-de-ralentissements-66896673.html`.
39. M. Mellia, M. Meo, L. Muscariello, and D. Rossi. Passive analysis of tcp anomalies. *Elsevier Computer Networks*, 52(14), October 2008.
40. ModelNet-TE Web page. `http://perso.telecom-paristech.fr/~drossi/index.php?n=Software.ModelNet-TE`.
41. A. Montresor and M. Jelasity. Peersim: A scalable p2p simulator. In *IEEE P2P'09*, pages 99–100, 2009.
42. L. Muscariello, D. Perino, and D. Rossi. Do next generation networks need path diversity? In *Proc. of IEEE International Conference on Communications, (ICC'09)*, Dresden, Germany, Jun. 2009.
43. S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers. The state of peer-to-peer simulators and simulations. *ACM SIGCOMM Computer Communication Review*, 37(2):95–98, 2007.
44. Onelab home page. `http://www.onelab.eu`.
45. A. Orda, R. Rom, and N. Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Transactions on Networking*, 1(5):510–521, 1993.
46. M. Piatek, T. Isdal, T. Anderson, A. Krishnamurthy, and A. Venkataramani. Do incentives build robustness in bittorrent. In *Proc. of 4th USENIX Symposium on Networked Systems Design & Implementation (NSDI'07)*, Cambridge, MA, USA, Apr. 2007.

47. F. Picconi and L. Massoulié. Is there a future for mesh-based live video streaming? In *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P'08)*, Aachen, Germany, Sep. 2008.
48. F. Picconi and L. Massoulié. Isp friend or foe? making p2p live streaming isp-aware. In *In Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS'09)*, Montreal, Quebec, Canada, 2009. IEEE.
49. Planetlab home page. `http://www.planet-lab.org`.
50. L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. On selfish routing in internet-like environments. In *in Proc. of ACM SIGCOMM*, Karlsruhe, Germany, Aug. 2003.
51. A. Rao, A. Legout, and W. Dabbous. Can Realistic BitTorrent Experiments Be Performed on Clusters? In *IEEE International Conference on Peer-to-Peer Computing (P2P'10)*, Delft, Netherlands, Aug 2010.
52. D. Ren, Y. Li, and S. Chan. On reducing mesh delay for peer-to-peer live streaming. In *IEEE INFOCOM*, Phoenix, AZ, USA, Apr. 2008.
53. D. Rossi, C. Testa, S. Valenti, and L. Muscariello. Ledbat: the new bittorrent congestion control protocol. In *Proc. of International Conference on Computer Communication Networks (ICCCN'10)*, Zurich, Switzerland, Aug. 2010.
54. D. Rossi and P. Veglia. An hybrid approach to assess the network awareness of p2p-tv applications. *International Journal of Digital Multimedia Broadcasting, SI on Network-Aware Peer-to-Peer (P2P) and Internet Video*, 2010.
55. D. Rossi and P. Veglia. Assessing the impact of signaling on the QoE of push-based p2p-tv diffusion algorithms. In *Proc. of 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS'11)*, Paris, France, Feb. 2011.
56. T. Roughgarden and va Tardos. How bad is selfish routing. *Journal of the ACM*, 49:236–259, 2002.
57. J. Seibert, D. Zage, S. Fahmy, and C. Nita-Rotaru. Experimental comparison of peer-to-peer streaming overlays: An application perspective. In *In Proc. of IEEE Conference on Local Computer Networks (LCN'08)*, Montreal, Canada, oct. 2008.
58. P. Shumate. Fiber-to-the-home: 1977–2007. *Journal of Lightwave Technology*, 26(9):1093–1103, 2008.
59. T. Silverston, O. Fourmaux, K. Salamatian, and K. Cho. On Fairness and Locality in P2P-TV through Large-Scale Measurement Experiment. In *Proc. of IEEE GLOBECOM*, Miami, FL, USA, Dec 2010.
60. N. Spring, L. Peterson, A. Bavier, and V. Pai. Using planetlab for network research: myths, realities, and best practices. *SIGOPS Operating Systems Review*, 40:17–24, January 2006.
61. C. Testa and D. Rossi. The impact of uTP on bittorrent completion time. In *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P'11)*, Kyoto, Japan, Sep 2011.
62. A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and accuracy in a large-scale network emulator. In *Proc. of 5th USENIX Symposium on Operating Systems Design and Implementation (OSDI'02)*, 2002.
63. Table of united states metropolitan statistical areas. `http://en.wikipedia.org/wiki/Table_of_United_States_Metropolitan_Statistical_Areas`.
64. B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proc. of ACM SIGCOMM*, Philadelphia, Pennsylvania, USA, Aug 2005.
65. D. Wu, P. Dhungel, X. Hei, C. Zhang, and K. Ross. Understanding peer exchange in bittorrent systems. In *Proc. of IEEE International Conference on Peer-to-Peer Computing (P2P'10)*, Delft, Netherlands, Aug. 2010.
66. D. Xu, M. Chiang, and J. Rexford. Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering. *IEEE/ACM Transactions on Networking*, PP(99):1, 2011.
67. C. Zhang, P. Dhungel, D. Wu, and K. Ross. Unraveling the bittorrent ecosystem. *IEEE Transactions on Parallel and Distributed Systems*, 22(7):1164 –1177, july 2011.
68. H. Zhang, Y. Liu, W. Gong, and D. Towsley. On the interaction between overlay routing and mpls traffic engineering. In *In Proc. of ACM SIGCOMM, Poster Session*, Portland, OR, USA, 2004.