Path Based Network Modeling and Emulation

Pramod Sanaga, Jonathon Duerig, Robert Ricci, and Jay Lepreau
School of Computing , University of Utah
Salt Lake City, Utah, USA
pramod@cs.utah.edu, duerig@cs.utah.edu, ricci@cs.utah.edu, lepreau@cs.utah.edu

1. ABSTRACT

Most of the commonly used network emulators are fundamentally *link* emulators, not *path* emulators: they concentrate on realistic emulation of the transmission and queuing behavior of individual network hops. To accurately model a realistic Internet-like multi-hop path, one must combine multiple link-emulator instances to create a detailed router-level topology.

In our work, we aim to develop a model of Internet path behavior, targeted at path emulation. Our work abstracts an Internet path, modeling the relevant characteristics that dominate the path's observed behavior. In addition to traditional path characteristics like bandwidth and latency, it also models shared bottlenecks, the reactivity of background traffic and distinguishes between capacity and available bandwidth. This model simplifies the work involved in emulating realistic Internet paths and significantly decreases the resources needed for the emulation. We implemented the model and performed an evaluation, with promising results.

Categories and Subject Descriptors I.6.5 [**Computing Methodologies**]: Model Development

General Terms: Experimentation, Measurement

Keywords: Available Bandwidth, Background Reactivity, Path Modeling, Realistic Emulation, Shared Bottlenecks

2. INTRODUCTION

Popular emulators such as DummyNet [2], NISTNet [1], ModelNet [4], and Emulab (which uses Dummynet) [3] focus primarily on *link emulation*, meaning that they are concerned with realistic emulation of individual links and queues. In order to emulate a realistic Internet path with a link emulator, one must use multiple instances of a link emulator to create a router-level topology. The input parameters for a model with this level of detail, however, are often not available and are not observable from end hosts on the Internet.

Our goal is to model Internet paths as a whole rather than specific links in a router-level topology. Our emulation model takes as input a number of characteristics which describe the end-to-end behavior of paths, and then *abstracts* these paths as much as possible. The parameters of our model include both an extended model of the reactivity of background traffic on the network and relationships between paths.

This material is based upon work supported by the National Science Foundation under Grant Nos. 0335296, 0205702, and 0338785. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Foreground flows	Max. Deviation Error	
	Link Emulator	Path Emulator
UniDir	6%	5%
BiDir - Symmetric	22%	7%
BiDir - Asymmetric [1.5 : 1]	30%	6%
BiDir - Asymmetric [2.5 : 1]	50%	5%
BiDir - Asymmetric [3.0 : 1]	66%	10%

Table 1: Throughput benchmarks

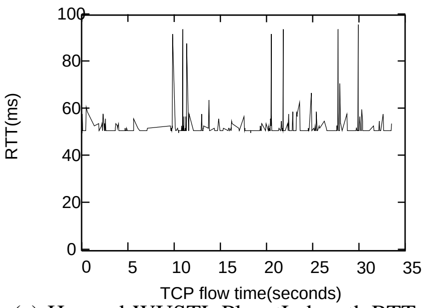
3. OUR APPROACH

We have identified four fundamental principles for modeling abstract, rather than router-level, Internet paths. Our contribution consists of identifying these principles, exploring them in depth, and implementing them in a path emulator.

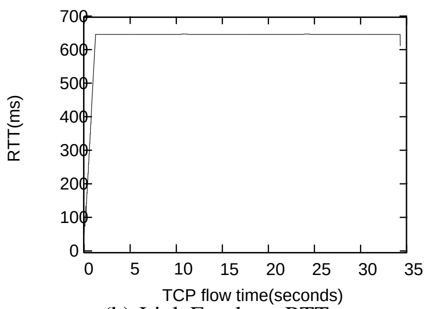
Model capacity and available bandwidth separately. In link emulators, the bandwidth parameter typically sets the emulated link's capacity. When this parameter is used to achieve a desired available bandwidth (ABW), subtle artifacts can cause incorrect behavior. The queue drains at a rate determined by the ABW we have set, which, on a real Internet bottleneck link, is typically a fraction of the capacity. Thus, the maximum queuing delay seen by a flow's packets is likely to be greater than the queuing delay that would be seen on an Internet link with the same ABW. This causes the total delay experienced by the flows in the emulation to be higher than it would be on a comparable shared Internet path.

This can cause not only errors in latency, but also errors in ABW, by artificially increasing the delay-bandwidth product of a link to a point where a TCP sender does not have sufficient buffer or window size to reach the full ABW of the link. This is exacerbated on paths with asymmetric ABW where there is a flow in each direction on the path. Suppose the ABW in the forward direction is much greater than the ABW in the reverse direction. If capacity is set to ABW, the reverse direction will have much longer queuing delays, and the ACKs of the forward high-bandwidth flow will be penalized greatly while waiting in the reverse queue. This additional delay on the reverse path can make the forward path window limited, preventing it from achieving the full target bandwidth.

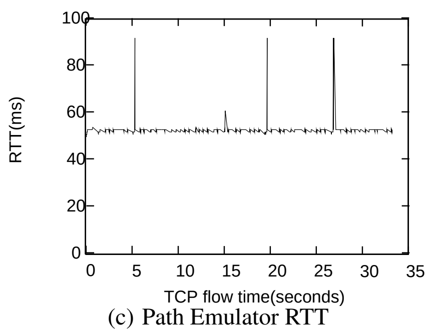
Let C be the capacity of the emulated path and ABW be the desired available bandwidth to be emulated on that path. To address this problem, we set $C > ABW$. In order to increase C without increasing ABW, we must fill the remaining capacity with other traffic. To get as close as possible to the target ABW in the emulation, we use constant bit rate, unresponsive traffic for this purpose. We combine this with the reactivity model described below to mimic the effects of responsive background traffic.



(a) Harvard-WUSTL PlanetLab path RTT



(b) Link Emulator RTT



(c) Path Emulator RTT

Figure 1: RTT Comparison

Pick appropriate queue sizes. Much work has been done in choosing "good" values for queue sizes in real routers, but the issues that apply to emulation are somewhat different and result in the following constraints.

1. The queue must be large enough that a TCP stream is able to get the full desired ABW; it should not drop bursts of packets.
2. The queue must not be so large that the queuing delay from a full queue causes excessive RTTs and could cause flows to become limited by buffer or window size.

The maximum RTT for a TCP flow f on path p with available bandwidth $ABW_{p,f}$ in the flow's direction, before it becomes window-size limited (assuming a buffer of at least the maximum window size allowed by the Operating System, w_{max}) and thus loses throughput, is given by:

$$RTT_{max,f} = \frac{w_{max}}{ABW_{p,f}} \tag{1}$$

If RTT is greater than $RTT_{max,f}$, the bandwidth-delay product exceeds the maximum window size, preventing TCP from having enough packets in flight to achieve the specified ABW. A path potentially has flows in the forward and reverse directions. The goal is to set the queue sizes in both directions so that TCP flows going either way do not become window limited. Without loss of generality, we define the "forward" direction to be the one with the higher ABW, and thus $RTT_{max,f}$ is always the smaller of the max. RTT tolerated by TCP flows in both directions.

The base RTT (RTT_{base}) denotes the sum of transmission, processing and propagation delays in both directions of the path. Assuming fully occupied queues,the total RTT for the path is given by:

$$RTT_{base} + \frac{q_f}{C_f} + \frac{q_r}{C_r} \tag{2}$$

By setting this less than or equal to $RTT_{max,f}$, which is given by Equation 1, setting the two capacities to be equal, and solving for the queue sizes, we get a limit on the queue sizes:

$$q_f + q_r <= C \cdot \left(\frac{w_{max}}{ABW_f} - RTT_{base}\right) \tag{3}$$

Abstract the reactivity of background flows. Rather than model competing background traffic directly, we model its reactivity by considering ABW to be a function of the number of foreground flows, adjusted in real-time as the application inside the emulator varies its traffic patterns. This gives a great deal of flexibility in the modeling of network reactivity. The ABW function can be created analytically or from measurements, and the emulation can provide—with high accuracy—exactly the desired ABW.

Model shared bottlenecks. In the Internet, some paths may share a bottleneck. If paths are modeled independently, this can cause the emulation to miss important interactions that occur on these shared bottlenecks. We model the relationships between paths that share bottleneck links by defining equivalence classes to place them into common bottleneck queues. Each equivalence class corresponds to a set of destinations that have a shared bottleneck from a given source, and all flows in an equivalence class share a common bottleneck queue.

4. RESULTS

We evaluated our path emulator and compared the results with a link emulator. We used a number of different ABW and round-trip time values observed on PlanetLab paths. Results for both uni-directional and bi-directional transfers are presented in Table 1. It is clear from the error margins in the tests that the link emulator performs poorly when emulating Internet path measurements, resulting in 50–60% lower throughput compared to the target available bandwidth. Our path emulation model makes the most difference on asymmetric paths, always being within roughly 10% of the target bandwidth. The graphs in Figure 1 show the round trip time experienced by a TCP flow on a PlanetLab path, versus the RTT in a link emulator and with our new path emulator. Our model produces more realistic delays.

We are evaluating our system further to quantify the gains in realism by modeling shared bottlenecks and reactivity. Our ongoing work includes considering the effects of destination-shared bottlenecks and improving the realism with which we can measure and generate reactivity models.

5. REFERENCES

[1] M. Carson and D. Santay. NIST Net: a Linux-based network emulation tool. *ACM SIGCOMM Computer Communication Review (CCR)*, 33(3):111–126, 2003.

[2] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *Computer Communication Review*, 27(1):31–41, Jan. 1997.

[3] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 255–270, Boston, MA, Dec. 2002.

[4] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker. Scalability and accuracy in a large-scale network emulator. In *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, pages 271–284, Boston, MA, Dec. 2002.