A Device-Independent Router Model

Roman Chertov, Sonia Fahmy, Ness B. Shroff
E-mail: {rchertov, fahmy}@purdue.edu, shroff@ece.osu.edu

***Abstract*—Several popular simulation and emulation environments fail to account for realistic packet forwarding behaviors of commercial switches and routers. Such simulation or emulation inaccuracies can lead to dramatic and qualitative impacts on the results. In this paper, we present a measurement-based model for routers and other forwarding devices, which we use to simulate two different Cisco routers under varying traffic conditions. The structure of our model is device-independent, but requires device-specific parameters. We construct a profiling tool and use it to derive router parameter tables within a few hours. Our preliminary results indicate that our model can approximate the Cisco routers. The compactness of the parameter tables and simplicity of the model makes it possible to use it for high-fidelity simulations while preserving simulation scalability.** [1]

I. Introduction

Critical properties of Internet forwarding devices such as intra-device latencies, maximum packet forwarding rates, policies, and queue sizes are not accurately incorporated into popular simulators and emulators such as ns-2 [4] and DETER [10]. These properties are crucial when modeling low-to-mid level routers. Compared to core routers, low-to-mid level routers are performance-limited. Yet, due to cost considerations, they constitute the majority of the forwarding devices in Internet edges and enterprise networks, which is where most packet losses in today's Internet occur. Accurately modeling these devices is especially important in experiments with high resource utilization, since resource consumption models used in simulators and emulators are not representative of today's commercial routers [5], [6]. Discrepancies between the simulated and deployment behaviors can be large for security experiments (e.g., denial of service), high bandwidth traffic (e.g., IPTV) scenarios, and network planning/dimensioning experiments (e.g., ISP upgrades). Our previous results with low-rate TCP targeted denial of service attacks (reported in [5]) demonstrate that seemingly identical tests on various testbeds and on the ns-2 simulator produce very different results. The discrepancies in the results arise because routers and other forwarding devices have complex architectures with multiple queues and multiple bottlenecks (e.g., buses, CPUs) [2] that change in complex ways according to the characteristics of the workload they are subjected to.

In commercial simulators such as OPNET [19] and OMNeT++ [1], detailed models of routers, switches, servers, protocols, links, and mainframes are provided. However, the model base needs to be constantly updated and validated, and using complex models significantly increases computational cost, hindering scalability.

We believe that in order for a model to be useful, it must meet the following requirements: (i) the model derivation process is the same regardless of the device; (ii) the model is dynamic, reflecting behavior under changing workloads; (iii) model parameters are derived without assuming any knowledge about device internals; (iv) the model is accurate, but is allowed to miss special cases for the sake of scalability; and (v) the model is not computationally expensive.

The only recent study we found that has attempted to model a router based on experimental observations was [14]. The authors created a Virtual Output Queuing (VOQ)-based model that added delays to the packets prior to placing them into a FIFO output queue. The delays were based on empirical observations of a production Tier-1 router. However, the VOQ-based model assumes no interactions among inputs, no backplane contention, and no loss (uses infinite queues). This was adequate for modeling a Tier-1 router under light and medium load, but the model does not generalize to other forwarding devices.

In this paper, we devise a novel model that differs from the VOQ-based model in a number of key aspects, most importantly the queue size and number of servers. Our model is the same regardless of the router type, but the model *parameters* make it unique for a specific type. Additionally, we devise a parameter derivation process based on simple measurements. To acquire the measurement data for each router, we leverage our *Black-Box Profiler* or BBP [6]. BBP is an SMP PC that serves as a traffic generator and a packet logger. We use our measurements to model two low-to-mid level Cisco routers: 3660 and 7206VXR. Our preliminary experiments reveal that our model is capable of capturing different router behaviors. We believe that this is a significant step forward toward creating high-fidelity scalable simulations.

The remainder of this paper is structured as follows. Section II explains our model and the model parameter derivation process. Section III gives an overview of our BBP system. Section IV describes a simple simulator that we developed to compare our model predictions with actual observations. Section V gives the details of our test setup. Section VI discusses our results. Section VII summarizes related work. We conclude in Section VIII.

II. Device-Independent Router Model

There is a multitude of router types with very different architectures and performance. Designing a completely different model for each type would be time-consuming and error prone.

[1]Roman Chertov and Sonia Fahmy are with the Department of Computer Science, Purdue University. Ness B. Shroff is with the departments of ECE and CSE, The Ohio State University. This research has been sponsored in part by NSF/DHS grant 0335247 and NSF grant 0523249.

However, regardless of the type, routers share a few critical similarities:

1) Packets may get dropped or delayed within the router.
2) Routers have a number of input and output interfaces.
3) Routers can have intermediate buffers/queues.
4) Packet flow in a router is complex [2], and there can be several queues and servers for each part of the path.
5) Packets can be split into parts while traveling between the input and output ports (as in several devices that use fixed-size "cells") [8].
6) Shared components such as the backplane, routing cache, and possibly a central processor can lead to interference among flows that do *not* share the same outputs.

The complexities of real router tasks introduce difficulties in developing an accurate and comprehensive model. A real router must deal with control packets such as ARP and ICMP, as well as routing packets such as BGP, OSPF, and RIP. The control/routing packets can have a profound impact on the forwarding of regular packets. For instance, ARP can lead to a significant delay until the right mapping between a packet's IP and MAC addresses is established. Routing packets can lead to delays or losses of regular packets as routes are removed or added. Routers can have interfaces with different speeds and hardware (e.g., Ethernet/FastEthernet/SONET, etc.). Hence, for the sake of simplicity, we will make a few assumptions to create a general packet forwarding model:

1) We do not model control traffic (OSPF/BGP/ARP/ICMP etc.).
2) We assume that all the interfaces have the same performance.
3) We assume that data packets are treated equally (no Quality of Service).
4) We assume full duplex operation.

We do not assume any knowledge of router internals or traffic statistics, however.

A. Virtual Output Queue (VOQ)-Based Model

Fig. 1 depicts the Virtual Output Queue (VOQ)-based router model suggested in [14]. The model is similar to the classical output queue model, except that there is a constant delay added to each packet based on its packet size. The extra delay signifies additional router overhead required for packet processing. This delay is derived from experimental measurements. Each output port is modeled in this fashion, ignoring any interactions at the inputs and the backplane. We believe that this model is sufficiently accurate for core routers which have a sophisticated switching fabric, but it can be *inaccurate* for lower-end devices. For example, there was no loss observed in the core router in [14], and hence the queues have unlimited capacities. The VOQ model is quite attractive due to its simplicity; however, it fails to account for details that can lead to large deviations in the results with other types of forwarding devices.

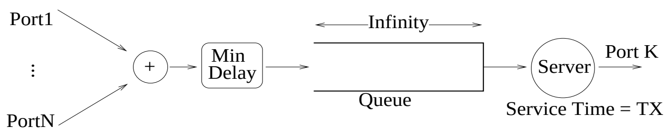


Fig. 1. Minimum delay queuing model with an unbounded queue *per output port*. The service time is based on the packet transmission (TX) time.

B. Multi-Server/Multi-Queue Model

We observe that traffic interactions in routers can play a significant role in causing packet loss and delay. Fig. 2 demonstrates our device-independent model. The additional complexity over the VOQ-based model allows modeling devices with limited performance characteristics, in addition to the Tier-1 router modeled in [14].

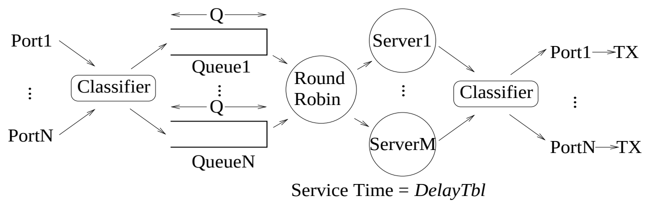


Fig. 2. N router inputs are served by M servers. There is one queue per port. Packets exit the router through one of the N output ports.

Based on the router similarities described above, in our model:

1) Each output port has a fixed queue of size Q slots. However, packets can occupy more than one slot in the case of byte-based queues. Hence, a table $QueueReq$ will be used to specify how many slots a given packet size occupies.
2) Traffic from N inputs is classified and queued by output port, served by M servers and proceeds to N outputs for transmission.
3) Servers process packets with the observed average processing delay. A table, $DelayTbl$, represents observed *router delays* (excluding transmission delay), as described in Section III, for various packet sizes. This is similar to "Min Delay" in Figure 1.
4) Packets can be served concurrently by different servers, but packet transmissions on the same output link do not overlap.
5) Since packets may be split into smaller units (cells) internally within a router [8], some packets may need more than one server to process them. Hence, another table, $ServReq$, gives the number of servers required to process packets of different sizes.

As previously stated, routers may have multiple queues on the packet path from the input to the output. Modeling the location of all the queues and their respective sizes would require detailed knowledge of each router internals. Since this is infeasible, we approximate all the internal queues as a *single aggregate queue* of size Q slots per output port. We infer Q and $QueueReq$ from our measurements. When there is backplane contention resulting in the slow drain of the input queues in a device, queues in our model overflow causing packet drops. If the contention occurs at the output queues of a device, then this contention also propagates from the servers to the queues in our model causing packet drop.

We utilize multiple internal *servers*. In a routing or forwarding device, input/output queues are served by the processors on the network cards, while intermediate queues might be served by a central processor(s) or specialized switching fabric processors. Since it is difficult to determine the exact number of servers, we derive the number of servers, M, based on measurements. Varying M from one to infinity allows us to model the entire range of routers from those with severe contention to those with none at all. The service delay is composed of two parts. First, there is the *router delay* due to internal processing, as discussed above. Second, there is delay due to the actual packet transmission. Multiple servers in our model allow packets on the same path to be served in parallel. Of course, transmission delays cannot overlap for packets going out of the same port. Fig. 3 demonstrates this scenario.

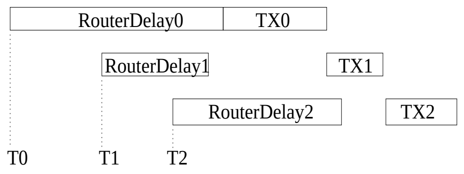


Fig. 3. Three packets destined to the same output are concurrently served by three servers. A packet is not transmitted on the output link until the previous packet is sent out.

Finally, based on the fact that packets can be split into small units within a router, we allow cases where a packet requires more than one server through the table $ServReq$.

C. Derivation of Model Parameters

As discussed above, our model has parameters that are unique to each routing or forwarding device: (1) M represents the number of servers. (2) Q represents the size of the aggregate queue per interface (recall that we assume equivalent interfaces). (3) $DelayTbl$ denotes the processing delays for various packet sizes. (4) $QueueReq$ denotes the number of queue slots occupied by a given packet size. (5) $ServReq$ denotes the number of required servers for a given packet size.

The model parameters can be derived experimentally by subjecting a router to a series of simple tests. Before describing the derivation process, we need to give some notation:

1) N is the total number of device interfaces.
2) $TX\_Capacity$ is the maximum transmission capacity of an interface.
3) $LowRate$ is a rate at which queuing delay does not occur but that allows sufficient (say 1000–2000) samples to be collected in a short time (e.g., 50 pps).
4) p is a packet path between two interfaces.
5) P is the set of all the possible paths p during the test.
6) $|P| = N(N-1)$ is the number of all paths.
7) S is a set of packet sizes s = $\{64, \ldots 1500\}$ bytes.
8) R is a set of packet rates r (in packets per second), including rates that induce packet loss.
9) $D_{s,r,p}$ is the measured average packet delay from input to output for packets of size s at rate r on the path p. The minimum of these average delay values for a specific packet size is denoted by d_s.
10) $DepartGap_{s,r,p}$ is the measured average gap between the packets when leaving the router, while $ArriveGap_r$ is the gap between arriving packets for rate r.

Algorithm 1 is used to compute $DelayTbl$, Q, M, $QueueReq$, and $ServReq$. The first step is to determine the values of $D_{s,r,p}$ in order to create the $DelayTbl$ for various packet sizes. Injecting Constant Bit Rate (CBR) flows through the router is a simple and efficient way to do so.

Algorithm 1 Parameter Derivation Algorithm

```
Input: Any forwarding device with $N$ interfaces.
Output: $DelayTbl$, $M$, $Q$, $QueueReq$, $ServReq$.
1: Determine $D_{s,r,p}$ for all packet size $s$ and rate $r = LowRate$ on path $p$
2: if $D_{s,r,p1} \approx D_{s,r,p2}$ $\forall s \in S$, and $\forall p1, p2 \in P$ then
3:     $d_s = avg(D_{s,r,1}, ..., D_{s,r,|P|})$
4: else
5:     exit
6: end if
7: for each $s \in S$ do
8:     $\forall N$, find max. rate $r$ s.t. no loss
9:     $ArriveGap_r = \frac{1}{Nr}$
10:    $NumServ_s = \frac{d_s}{ArriveGap_r}$
11: end for
12: for each $s \in S$ do
13:    find min. rate $r$ s.t. when $N-1$ ports send to one port at rate $r$ each, $rs(N-1) > TX\_Capacity$
14:    $QSize_s = \frac{D_{s,r(N-1),p}}{DepartGap_{s,r(N-1),p}}$
15: end for
16: $DelayTbl =$ set of $d_s, \forall s \in S$
17: $M = max(NumServ_1, \ldots, NumServ_{|S|})$
18: $Q = min(QSize_1, \ldots, QSize_{|S|})$
19: $ServReq = M/NumServ_s, \forall s \in S$
20: $QueueReq = Q/QSize_s, \forall s \in S$
```

The intuition behind the algorithm is as follows. *DelayTbl* is constructed by taking an average of the packet delays across different ports when the sending rate is very low. This is computed for a variety of packet sizes to construct a comprehensive table. If the delay differences between the interfaces are large, then we cannot continue, as our assumption that interfaces are approximately similar becomes invalid. Otherwise, we compute the maximum number of concurrent servers for each packet size *just before* loss starts occurring. This is done by utilizing all ports to transmit packets, such that flows do not create conflicts on the output ports. For example, suppose N is four, then flows port0-to-port1, port1-to-port0, port2-to-port3, and port3-to-port2 are non-interfering on their output. M is set to the largest number of servers estimated for all packet sizes. $ServReq$ can also be constructed based on the resulting values.

To estimate Q and $QueueReq$, we must create a high loss scenario to ensure queue build-up. We send flows from several ports into the output queue of another port. The $DepartGap$ between the packets in this case will indicate the maximum drain rate, meaning that the size of the queue can be estimated as the observed delay divided by $DepartGap$. The observed queue size is recorded in Q (we record the minimum size in units of maximum-sized packets).

III. Measurement System Overview

The model derivation in Section II-C is primarily based on packet delay and loss data. Acquiring such data at sufficiently high precision requires a high-fidelity measurement system. We have created such a system called the Black Box Profiler (BBP) [6]. In this section, we give a brief overview of BBP, and the updates we have made to it since [6] was published.

Fig. 4 demonstrates the layout of our BBP infrastructure connected to a 2-port commercial router. A Symmetric Multiprocessing (SMP) multi-NIC PC is used to emulate subnets that multiple flows can traverse. The router that is being profiled, e.g., a commercial Cisco or Juniper router or a programmable router, is configured to route between the subnets. To minimize the measurement error, the BBP system is directly connected to the router.

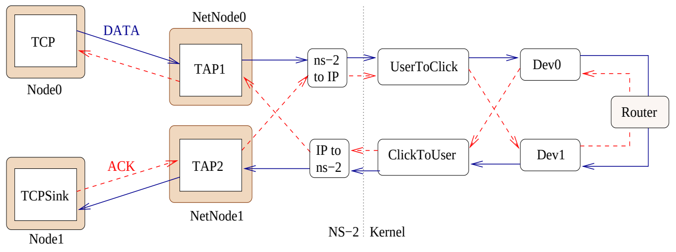


Fig. 4. Example of a single TCP flow from the simulator into the network and vice versa.

To create responsive (i.e., closed-loop) traffic, we leverage the ns-2 simulator which provides various TCP stacks and traffic workload models. We plan to extend our traffic generation capabilities by reproducing application workloads based on real-life traces as in [22], [23]. Our custom additions to ns-2 allow packets from ns-2 to be injected into the test network and vice versa. Since all packets originate and terminate on the SMP PC, we can embed arrival/departure timestamps into the packet payloads with micro-second precision, without worrying about clock skew/synchronization. The timestamping of packets occurs in the network device driver to obtain an accurate estimate of the delay. Additionally, we can provide highly accurate accounting per-packet and per-flow to determine delay, loss, reordering, and corruption.

Fig. 5 displays the components of measured packet delay. The measured delay is composed of the NIC send/receive overheads, two packet transmissions, and the router delay. A calibration phase is required to infer the NIC overheads and the packet transmission delay. Knowing the NIC overheads allows the computation of the router delay. During initial calibration, our setup is similar to Fig. 5, except that the NICs are directly connected with a cable and there is no router.

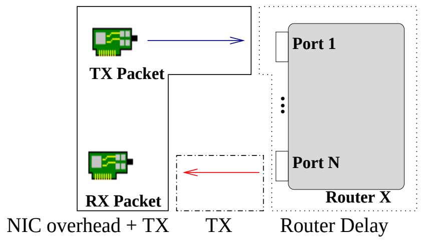


Fig. 5. Measured packet delay consists of NIC send overhead, NIC receive overhead, router overhead, and two transmit delays.

Traffic Generator. We use the ns-2 simulator [4] for traffic generation, since it provides several TCP implementations that have been validated by the research community. Further, ns-2 provides excellent capabilities for logging and debugging. We had to make several changes to ns-2, described in [6], to allow us to send and receive packets at extremely high rates, as well as send and receive spoofed IPs. These changes make it possible to have many flows with distinct IPs enter and depart from the simulator. From the router point of view, the BBP appears as a collection of various subnets with unique flows between them.

The version of BBP described in [6] had two shortcomings. First, we could not create a CBR UDP flow of more than 90 Kpackets/s as the scheduler and the TCL subsystem became limiting factors. To overcome this issue, we have created a stand-alone tool called udp_gen which is a stripped down version of the BBP ns-2. The new tool allows sending/receiving around 200 Kpackets/s, with disk logging now being the bottleneck factor.

Second, we reduced the size of the measurement payload to 12 bytes. This allows us to embed the measurement payload as a TCP option. This new arrangement allows creating correct TCP packets that do not have an extended payload when there should be none (e.g., SYN/ACK/FIN). Since there is limited space for TCP options, our measurement payload option can only be combined with a time-stamp or a three-block SACK option. For UDP and other IP packets, the measurement payload remains in the data portion.

Click Modular Router. The default Linux IP stack was unsuitable for our purposes for two reasons. First, the default stack was not designed to efficiently handle sending/receiving non-existent IPs to/from a user-level application. Second, the default stack has several features that we do not need, which add overhead. Hence, we use the Click modular router [16] kernel module. In Click, it is easy to create a mapping of IPs to real devices as shown in Fig. 4. In order to attach virtual subnets to a particular device, we use a *UserToClick* element per device, and the user application writes IP packets into the correct *UserToClick* element depending on the configuration.

In the latest version of BBP, we modified Click's *ToDevice* element to avoid transmit buffer drops. The default Click

ToDevice element can schedule packets faster than the device can transmit. Instead, we *hold* the packets until the transmit buffer starts draining.

Device Driver. Since we aim to measure packet delays in the router under test and not in our system, we must instrument the device driver. This is as close as we can get to the point where the packets get transmitted or received without requiring a specialty card. Fig. 6 demonstrates the steps we take to time-stamp packets in the device driver.

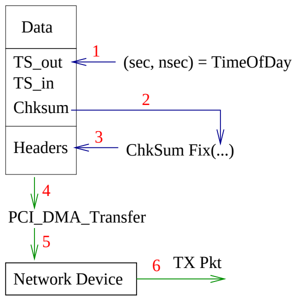


Fig. 6. Time-stamping of packets during a transmit. Time-stamping during a receive is similar, except the flow is reversed with checksum correction being the last step.

When a packet arrives, we time-stamp it just before it is sent to the device via a bus transfer. Since changing the packet payload will result in a corrupted TCP or UDP checksum, we recompute a new checksum. We compute incremental checksums to avoid computing an entire checksum from scratch. Packet reception is done in a similar fashion.

IV. The Queue Simulator

In order to evaluate the fidelity of our model, we have devised a simulator of the model, which we call *QSim*. The simulator mimics the Multi-Server/Multi-Queue model exactly as in Fig. 2. We use the derivation algorithm in Section II-C to compute the model parameters. The simulator then replays the packet trace and compares the delays and losses with the simulator against the observed delays and losses. If, for a particular packet size, there is no data in the $DelayTbl$, $QueueReq$, or $ServReq$ tables, the simulator performs a linear interpolation between the two adjacent packet sizes which are in the tables. Since the observed delays include the NIC overheads (Fig. 5), we subtract the delays we obtain during calibration runs from the observed delays to make an accurate comparison.

V. Experimental Setup

This section explains the network setup for our experiments as well as the BBP configuration.

A. Network Setup

Fig. 7 demonstrates our test setup. In the experiments, *Node0*, *Node1*, *Node2*, and *Node3* are logical nodes on the same PC, while the “Router” node is either a pair of cross-over cables that connect four cards on the PC running BBP, a Cisco 3660 router, or a Cisco 7206VXR router. The Cisco routers under test in our experiments have *four Fast Ethernet ports*. The Cisco 3660 has two identical cards on the main data plane and a dual port swappable module, while the Cisco 7206VXR has one main card and three swappable modules. The Cisco routers were configured with minimal settings to ensure that forwarding between the ports would happen on a *fast path* without special processing. Additionally, we have enabled the Cisco Express Forwarding (CEF) option [9] on both routers. The cross-over cable configuration is used solely for calibration, in order to determine the latencies due to the network cards. The queue size for all the links in the traffic generator was set to 50 slots. The queue sizes on the physical link are dictated by the particulars of the hardware.

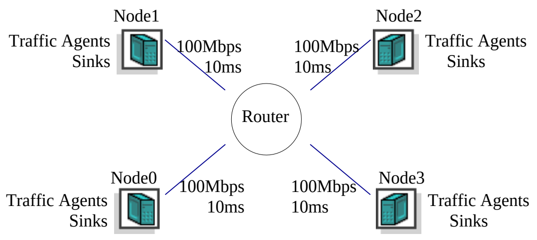


Fig. 7. Test topology with different subnets

B. BBP Configuration

We use a PC with two quad 1.8 GHz Xeon CPUs and PCI-E Intel Pro cards to run BBP on. Naturally, the measurement process must allow several tasks to execute concurrently to achieve the highest precision. Fig. 8 demonstrates the main tasks that must run concurrently in order for the BBP to give precise results.

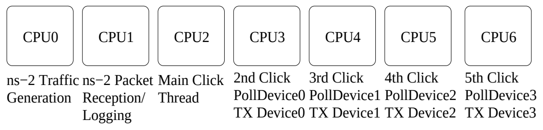

CPU0
CPU1
CPU2
CPU3
CPU4
CPU5
CPU6
ns−2 Traffic Generation
ns−2 Packet Reception/ Logging
Main Click Thread
2nd Click PollDevice0 TX Device0
3rd Click PollDevice1 TX Device1
4th Click PollDevice2 TX Device2
5th Click PollDevice3 TX Device3

Fig. 8. Concurrent threads of execution

The traffic generation component must have at least two threads of concurrent execution to achieve high packet rates. The main ns-2 thread runs all of the simulation agents and writes packets to *UserToClick* elements. Auxiliary ns-2 threads are required for reception of packets and logging data to disk; otherwise, the main simulation thread would become I/O blocked.

The Click modular router must also have at least two threads of concurrent execution. The main Click thread is responsible for all the elements in the Click configuration except for packet reception and transmission. If the elements in the main thread are delayed in scheduling, no measurement error will occur. This is because the main elements are not responsible for reading/writing packet timestamps. A problem arises during packet reception and transmission. If there is delay in element scheduling, then the packets will remain in the NIC’s send/receive buffer as time goes on. The variance in delay would increase in proportion to the packet rate increase. Hence, it is imperative to have a separate thread per *PollDevice/ToDevice*. Since we have four ports in our experiments, we need seven CPUs as depicted in Fig 8.

C. Calibration

Before we can proceed with data collection, we must determine which network device configuration would give the best performance and induce the least amount of noise into the measurements. This measurement noise results from the network card/bus specifics of our measurement machine. We had the best results with polling and 64-slot receive and transmit buffers. Fig. 9 demonstrates the measured delay between the two NICs compared to pure transmission delay. We used a constant flow of UDP packets to generate the results.

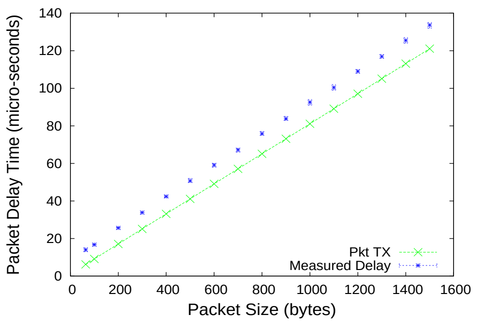


Fig. 9. NIC–to–NIC (mean, 5 and 95 percentiles) vs. pure TX delay

Tables I and II demonstrate the mean, 5th, and 95th percentiles for Ethernet frames of sizes of 64, 800, and 1500 at various rates. The percentiles indicate that variance is relatively small.

TABLE I
NIC–TO–NIC PACKET DELAYS FOR 64-BYTE ETHERNET FRAMES

	4 Kpps	8 Kpps	40 Kpps	110 Kpps	113.5 Kpps
mean	13.82	13.78	13.84	15.94	15.63
5th	13.00	13.00	13.00	13.00	13.00
95th	14.00	14.00	14.00	23.00	23.00

TABLE II
NIC–TO–NIC PACKET DELAYS FOR 800- AND 1500-BYTE ETHERNET FRAMES

	800 bytes		1500 bytes	
	4 Kpps	8 Kpps	4 Kpps	8 Kpps
mean	75.61	75.60	135.16	137.02
5th	75.00	75.00	135.00	135.00
95th	76.00	76.00	136.00	138.00

VI. EXPERIMENTAL RESULTS

This section outlines the model parameters derived for the Cisco 3660 and Cisco 7206VXR routers, and compares our model predictions to measurements.

A. Model Parameters

As discussed in Section II-C, we first compute delay tables. We vary the packet size from 64 to 1500 bytes, and keep the rate at a low 50 packets/s. The packet size includes Ethernet/IP/UDP headers. Fig. 10 compares the results for a *perfect router*, Cisco 3660, and Cisco 7206VXR. The results show the mean, 5th, and 95th percentiles. The *perfect router* is a hypothetical router that has zero processing and queuing latency, with packet transmission time being the only delay. We use the data from Fig. 9 to obtain the results for the perfect router: we add the NIC overhead to one additional packet transmit time (recall Fig. 5). The results indicate that the routers have significantly higher delays and variance than the perfect router.

Tables III and IV demonstrate the packet delays for 64, 800, and 1500-byte packets at different rates. In these experiments, only one port was used and there was no loss except for the 64-byte packets at 113.5 Kpackets/s on the Cisco 3660 router. At that setting, the queue did become full and losses occurred.

TABLE III
CISCO ROUTER PACKET DELAYS FOR 64-BYTE ETHERNET FRAMES

3660	4 Kpps	8 Kpps	40 Kpps	110 Kpps	113.5 Kpps
mean	72.52	78.04	70.21	331.13	7564.31
5th	45.00	46.00	45.00	58.00	2702.00
95th	99.00	99.00	98.00	1288.00	8290.00
7206	4 Kpps	8 Kpps	40 Kpps	110 Kpps	113.5 Kpps
mean	85.21	83.03	96.08	83.76	84.38
5th	38.00	40.00	41.00	43.00	44.00
95th	149.00	145.00	136.00	143.00	144.00

TABLE IV
CISCO ROUTER PACKET DELAYS FOR 800- AND 1500-BYTE ETHERNET FRAMES

	800 bytes		1500 bytes	
3660	4 Kpps	8 Kpps	4 Kpps	8 Kpps
mean	207.03	203.71	333.28	405.02
5th	179.00	179.00	307.00	346.00
95th	234.00	233.00	360.00	444.00
7206	4 Kpps	8 Kpps	4 Kpps	8 Kpps
mean	229.31	237.75	343.71	416.84
5th	182.00	190.00	296.00	346.00
95th	293.00	294.00	407.00	503.00

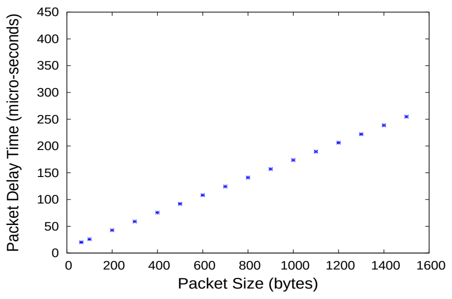
TABLE V
QUEUE SIZES FOR DIFFERENT PACKET SIZES

Router	64	200	400	600	800	1000	1200	1500
3660	1909	674	167	167	167	167	167	167
7206	272	294	295	167	167	167	125	125

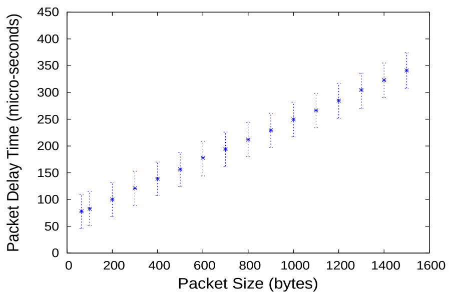
TABLE VI
NUMBER OF SERVERS FOR DIFFERENT PACKET SIZES

Router	64	200	400	600	800	1000	1200	1500
3660	6.027	6.080	5.661	4.653	3.969	3.389	3.126	2.786
7206	15.489	11.648	7.794	5.464	4.318	3.463	2.969	2.439

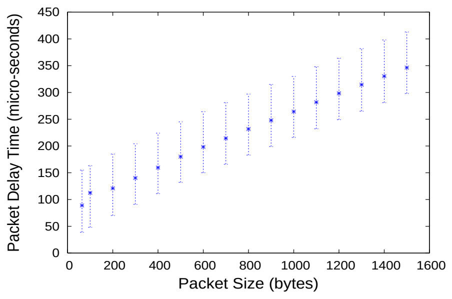
Using the model derivation process outlined in Section II-C, we compute parameter values for the two routers. Table V and Table VI demonstrate the measured values for $QSize$ and $NumServ$ respectively. The values of M, Q, $ServReq$, and $QueueReq$ for the two routers are computed from these tables, as given in Algorithm 1, lines 17–20. The $DelayTbl$s are omitted from the paper for brevity. As seen in the tables, there are 3 queue sizes (in terms of number of packets) for each of the routers for this set of packet sizes. This is consistent with the documentation in [7].



(a) Perfect Router



(b) Cisco 3660



(c) Cisco 7206VXR

Fig. 10. Observed minimum delays for different packet sizes

B. Model Fidelity

In our first series of experiments, we replay the simple CBR traces used in the model derivation experiments. We use *QSim* to model the Cisco 3660 and 7206VXR routers, and compare packet delay and loss values. *QSim* utilizes the time-stamps of when the packet departed the BBP device driver for the first time, as the time when the packet is injected into the simulation. Since the packet time-stamp reflects the time when the packet entered the device driver and not the router, we add a packet transmission time to the arrival time. This attempts to re-create the exact timing of when the packet entered the routing device.

During the validation runs, we noticed that loss creates a major complication for the accuracy of the trace replay in *QSim*. If the packet gets lost, time-stamp accuracy is compromised, since the device driver time-stamp is lost as well. The only available time-stamp is the one from ns-2 which may be a few milliseconds behind. Hence, the packet would appear in the simulation earlier than it would have in the original experiment. This can lead to inaccuracies between the observed and predicted data, as the events do not happen at exactly the same times in both cases. To make the rest of the comparisons accurate, we have removed the events when the actual packet or a predicted packet is dropped.

Our results (not shown due to space limitations) indicate that the model can accurately account for backplane contention. For instance, the model correctly predicts that Cisco 7206VXR cannot have all four interfaces forwarding 64-byte frames at more than 57 Kpps. Additionally, when multiplexing two flows into a single output port, the model correctly predicts the packet delays due to queue build-ups. The data also confirmed the need for $QueueReq$, as fixed size slot-based queues would have been insufficient for correctly modeling the routers.

To further investigate the accuracy of the model, we create a more complex traffic scenario. We use *FullTCP SACK* agents in ns-2 to represent the TCP agents. *FullTCP* agents in ns-2 mimic the Reno TCP implementation in BSD 4.4. *Node0*, *Node1*, and *Node3* in Fig. 7 generate 30 TCP and 5 UDP flows each, destined to *Node2*, for a total of 90 TCP and 15 UDP flows. TCP was configured to use up to 3 SACK blocks, 1400-byte payloads, and delayed ACKs. We use FTP and CBR agents at the application layer to drive the TCP and UDP traffic respectively. FTP agents were configured to send large continuous files, and UDP CBR agents were configured to send 100 256-byte messages per second. Certainly, such synthetic traffic is not representative of real-world traffic, but it subjects the routers to a high load. The experiment ran for 200 seconds on the Cisco 3660 and Cisco 7206VXR routers. We then replayed the traces in our simulator, *QSim*, using the model parameters we derived for Cisco 3660 and 7206VXR.

Table VII depicts the loss ratios in the actual and predicted data sets. The data indicates that the model has lower loss ratios than the routers. This implies that the model overestimates the forwarding capacity of the routers. We also ran the same traces on the VOQ model (which does not model backplane contention) and obtained similar loss ratios to our model. This implies that backplane contention resulted in almost two times the loss compared to output queue only losses. The replay fidelity problem due to losses, as mentioned above, may have contributed to this discrepancy.

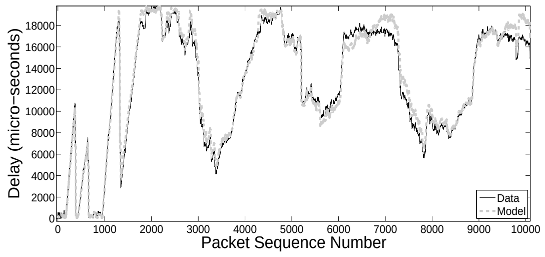
TABLE VII
AVERAGE TCP LOSS RATIOS

Test Type	Loss
Cisco 3660	0.0317
Predicted 3660	0.0182
Cisco 7206VXR	0.0445
Predicted 7206VXR	0.0184

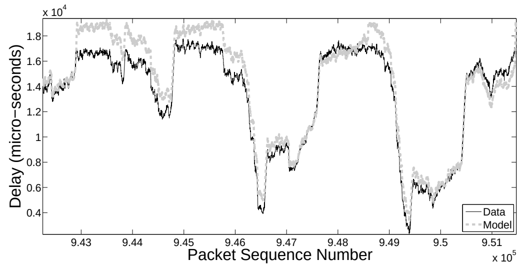
In addition to comparing the loss ratio, it is important to compare delay statistics. Table VIII reports the mean and Coefficient of Variation (COV) for the model predictions and the actual data set. It is interesting to note that even though the means for the predicted and observed delays are not far from each other, especially in the case of the 7206VXR, there is a mismatch in coefficients of variation except for $Node2$. More insight can be gained from a closer examination of the data in Fig. 11 and Fig. 12. In Fig. 11(a,b) and Fig. 12(a,b), the model tracks observed data quite well, because output port queuing delay is the dominant factor in the delay. The queuing delay reaches 18 ms and 14 ms on the 3660 and 7206VXR routers respectively. This is in contrast to core routers, which observe almost no queuing delay [13]. In Fig. 11(c) and Fig. 12(c), there is little output port queuing delay as only ACKs are being sent. In such a scenario, a VOQ model would predict only the minimum delay per packet size. Our model captures the dynamics of backplane contention that cause increased packet delays due to extra processing delays. The modeled contention is evident from the rising peaks on the graphs, though the observed data exhibits higher variation than the model predicts.

TABLE VIII
MEAN AND COEFFICIENT OF VARIATION OF CISCO 3660 AND 7206VXR PACKET DELAYS

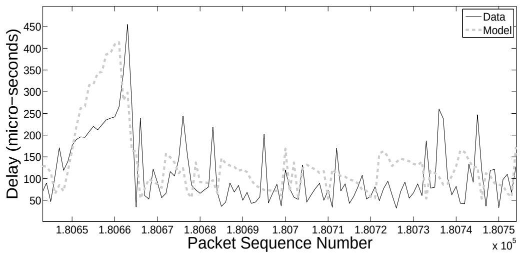
		3660		7206VXR	
Dst Node	Type	Mean	COV	Mean	COV
$Node0$	Predicted	111.095	0.506	71.795	0.326
	Observed	89.005	0.674	76.574	0.513
$Node1$	Predicted	110.957	0.495	72.460	0.345
	Observed	90.677	0.737	78.297	0.508
$Node2$	Predicted	15658.462	0.184	10822.451	0.251
	Observed	14636.507	0.176	11327.927	0.207
$Node3$	Predicted	113.013	0.536	71.925	0.329
	Observed	99.626	0.700	78.709	0.503



(a) Start of the experiment, destination $Node2$



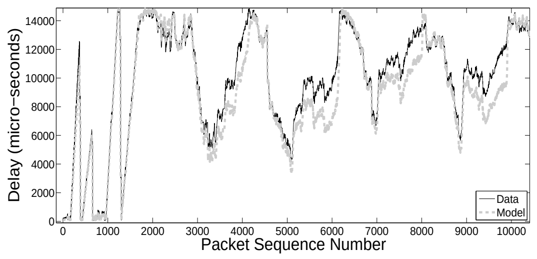
(b) Middle of the experiment, destination $Node2$



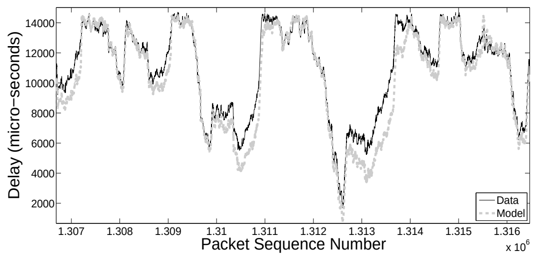
(c) Middle of the experiment, destination $Node3$

Fig. 11. High rate TCP/UDP traffic scenario on the 3660 port 3

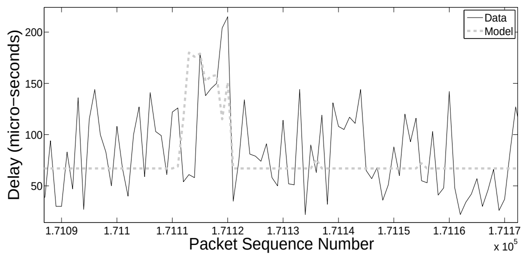
The high COV values for ports that do not have output queuing in Table VIII imply that processing delays are significant,



(a) Start of the experiment, destination $Node2$



(b) Middle of the experiment, destination $Node2$



(c) Middle of the experiment, destination $Node3$

Fig. 12. High rate TCP/UDP traffic scenario on the 7206VXR port 3

and induce a high degree of packet delay variance. To further analyze the accuracy of the model, we use the Kolmogorov-Smirnov (K-S) statistical test (maximum deviation of CDFs). For brevity, we do not show CDF plots and only report the resulting K-S values. Table IX reports the results, which demonstrate that the predicted values are a close approximation for 7206VXR on the heavily congested port. For the 3660, the congested port statistic indicates that the model is not as accurate as in the 7206VXR case, but it is better than the case of non-congested ports. With a VOQ model, the statistics for both routers on non-congested ports are 1, meaning that the model completely fails to capture backplane contention.

TABLE IX
KOLMOGOROV-SMIRNOV STATISTIC

Dst Node	3660	7206VXR
$Node0$	0.286	0.496
$Node1$	0.287	0.482
$Node2$	0.306	0.094
$Node3$	0.207	0.478

Our experiments reveal that the model predictions are reasonably accurate, but not always perfect, especially in cases when backplane contention is dominant. Our predictions are, however, *much closer* to a variety of routers than the VOQ-based model presented in Section II-A, or the simpler model in ns-2.

VII. Related Work

Black box testing and traffic generation are required for measurements and modeling. Hence, we summarize related work on these topics in this section.

A. Black-box Testing

Router modeling based on empirical observations is discussed in [14]. The model is called the Virtual Output Queue, and is described in detail in Section II-A. The work derived a simple queuing model, but was not designed to handle loss events, and ignored interactions at the input ports. A production Tier-1 router was used in their work. While this ensures that the router configuration and traffic are highly realistic, repeatability is not possible in a production setup. Time-stamping was performed with GPS synchronized DAG cards [11]. Such devices are highly accurate, but increase the setup cost and complexity.

Black-box router measurement is also described in [20], [3], [18]. In [20], a router is profiled with a focus on measuring its reaction times to OSPF routing messages. RFCs 2544 [3] and 2889 [18] describe the steps to determine the capabilities of a router (e.g., forwarding rate). The RFCs only discuss using homogeneous traffic for profiling, and do not discuss creating models based on measurements.

B. Traffic Generation

The Harpoon [21] traffic generator uses flow data collected by Cisco routers to generate realistic traffic. Creating highly configurable *live* (i.e., *closed-loop*) traffic is important for our purposes. One of the earliest network simulation-emulation tools was VINT [12] – a part of ns-2. We did not directly use the ns-2 emulation code as it does not support sending/receiving spoofed IPs (required for subnet emulation on a single node), and it is data-rate limited. A recent effort to extend emulation in ns-2 was reported in [17]. However, the system was not built to handle very high data rates and extensive packet logging with micro-second precision, which are important for our measurements. A commercial alternative to generating live TCP traffic is the IXIA-400T traffic generator [15]. IXIA devices use a proprietary OS and do not allow changing the types of the TCP stacks, however.

VIII. Conclusions and Future Work

In this paper, we have devised a device-independent model for forwarding devices, and outlined a general model parameter derivation procedure. We also developed a router profiling system that we refer to as BBP. Our model is router-agnostic and only requires a few parameter tables to mimic a specific router. The model tables are small, which makes the model highly portable and easy to compute. The model attempts to not only replicate packet delays due to router processing, but also packet losses.

We have derived model parameters for two low-to-mid range Cisco routers: 3660 and 7206VXR, and compared real observations to the model predictions. The comparison has revealed that the model is capable of closely matching the real data when output congestion is more dominant than backplane delays. We believe that incorporating such a model in simulators is an important step in increasing fidelity of network simulations, while preserving scalability.

Our future work plans include using a wider variety of router types for fine-tuning the model. We also plan to investigate more sophisticated traffic scenarios, and investigate backplane contention. For example, we would like to utilize *tmix* [23] so that traffic can be generated based on realistic application workloads. Finally, we plan to integrate our models with the ns-3 and emulation testbed development efforts.

References

[1] OMNeT++. http://www.omnetpp.org/.

[2] F. Baker. Re: [e2e] extracting no. of packets or bytes in a router buffer. Message to "end2end" mailing list, December 2006.

[3] S. Bradner and J. McQuaid. Benchmarking methodology for network interconnect devices. RFC 2544, 1999.

[4] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59–67, May 2000.

[5] R. Chertov, S. Fahmy, and N. B. Shroff. Emulation versus simulation: A case study of TCP-targeted denial of service attacks. In *Proc. of TridentCom*, February 2006.

[6] R. Chertov, S. Fahmy, and N. B. Shroff. A black-box router profi ler. In *Proc. of Global Internet*, May 2007.

[7] Cisco Systems. Basic system management. http://www.cisco.com/en/US/products/sw/iosswrel/ps1835/products_confi gu%ration_guide_chapter09186a008030c799.html#wp1009032.

[8] Cisco Systems. Cisco 12000 series internet router architecture: Packet switching. http://www.cisco.com/en/US/products/hw/routers/ps167/products_tech_note%09186a00801e1dc1.shtml.

[9] Cisco Systems. How to choose the best router switching path for your network. http://www.cisco.com/en/US/tech/tk827/tk831/technologies_white_paper091%86a00800a62d9.shtml.

[10] DETER. A laboratory for security research. http://www.deterlab.net.

[11] Endace. http://www.endace.com/.

[12] K. Fall. Network emulation in the vint/ns simulator. In *Proc. of ISCC*, pages 244–250, July 1999.

[13] C. Fraleigh and et. al. Packet-level traffi c measurements from the sprint ip backbone. *IEEE Network*, 17(6):6–16, Nov-Dec 2003.

[14] N. Hohn, D. Veitch, K. Papagiannaki, and C. Diot. Bridging router performance and queuing theory. In *Proc. of SIGMETRICS*, 2004.

[15] IXIA. http://www.ixiacom.com.

[16] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[17] D. Mahrenholz and S. Ivanov. Real-time network emulation with ns-2. In *Proc. of DS-RT*, pages 29–36, October 2004.

[18] R. Mandeville and J. Perser. Benchmarking methodology for LAN switching devices. RFC 2889, 2000.

[19] OPNET. Network modeling and simulation environment.

[20] A. Shaikh and A. Greenberg. Experience in black-box OSPF measurement. In *Proc. of IMW*, pages 113–125. ACM Press, 2001.

[21] J. Sommers and P. Barford. Self-confi guring network traffi c generation. In *Proc. of IMC*, pages 68–81. ACM Press, 2004.

[22] K. V. Vishwanath and A. Vahdat. Realistic and responsive network traffi c generation. In *Proc. of SIGCOMM*, 2006.

[23] M. C. Weigle, P. Adurthi, F. Hernandez-Campos, K. Jeffay, and F. D. Smith. Tmix: A tool for generating realistic application workloads in ns-2. *ACM Computer Communication Review*, 36:67–76, July 2006.