Preserving Pacing in Real Networks - An Experimental Study Using NetFPGA

Hamed Tabatabaei
Department of Computer Science
University of Toronto
hamedty@cs.toronto.edu

Yashar Ganjali
Department of Computer Science
University of Toronto
yganjali@cs.toronto.edu

ABSTRACT

Bursty traffic can lead to poor performance in networks with small buffers such as all-optical, or data center networks. Pacing flows at their sources can remove bursts and make the traffic smooth. However, even if the traffic is paced at the source, it can become bursty as it passes through multiple routers. *Bounded Jitter Policy* (BJP) has been proposed as a light-weight queue management scheme to address this problem. BJP tries to keep the traffic smooth by preserving packet inter-arrival times. To the best of our knowledge, the only existing evaluation of BJP is purely theoretical and makes various simplifying assumptions on network and input traffic. The question is whether BJP can achieve the desired properties in a real network and in the absence of simplifying assumptions on network traffic and router internals?

In this paper, we evaluate BJP using simulations and testbed experiments. For simulations, we have developed a flexible and accurate framework that allows arbitrary interconnection of BJP routers. For testbed experiments, we have implemented BJP on NetFPGA. Programmability and high accuracy in timing make NetFPGA the perfect platform for our experiments. Our BJP implementation can operate at line rate, and we use the same system for highly accurate inter-arrival time measurements. Our simulations and testbed experiments show that BJP works as promised: packet inter-arrival times (and thus the smoothness of traffic) are preserved regardless of the number of routers on the path. This is quite opposite to what happens in a network with round-robin routers where packets become bursty after passing through a few routers.

1. INTRODUCTION

Today, TCP is the dominant transport layer protocol in the Internet [2]. TCP's behavior has a major impact on the overall throughput and latency of flows in the network. There has been a significant amount of work on different variations of TCP and their dynamics [7]. However, most of these results focus on how congestion window sizes should evolve over time, i.e. how many packets a TCP flow should inject to the network during each RTT, and ignore the details of how packets should be injected during sub-RTT time scales.

In practice, most variations of TCP inject a burst of packets at the beginning of each RTT, and then wait for acknowledgements from the receiver. This bursty injection can have a significant negative impact on packet drop rates and therefore the overall throughput of the system especially in the context of networks with small buffers [4] like data center environments. Paced-TCP, proposed by Savage and Anderson [1], overcomes these problems by uniformly distributing packet injections over the entire RTT, thus making the packet injections smoother.

The benefits of using pacing have been studied theoretically, using simulations, and also in the context of real networks [3, 8]. Unfortunately, when a TCP flow passes through a network of routers[1] and switches the inter-arrival times of packets can change significantly. Therefore, even if packets of a given flow are paced initially, they can become arbitrarily bursty as they go through the network [5], which means the benefits of pacing will diminish.

In order to solve this problem and preserve pacing of packets throughout the network, Beheshti *et al.* have introduced *Bounded Jitter Policy* (BJP), a simple traffic shaping scheme which aims at retaining the smoothness of traffic no matter how many routers it passes [4]. They have analyzed BJP and shown that it can preserve pacing in the context of networks with tiny buffers. Their analysis is purely theoretical however: they use a simple mathematical model which ignores the internal details of a real router, as well as many other simplifying assumptions on packet arrivals and TCP flows.

One major question here is whether BJP can still preserve pacing (as promised) in the absence of the simplifying assumptions on network traffic and router internals. In other words, does a real implementation of BJP have the same level of performance as predicted in theory?

In this paper, we answer this question through numerous simulations, and real network experiments on BJP. In both cases, we accurately measure packet inter-arrival times for all flows in the network to see how the burstiness of traffic changes after going through multiple routers, and if BJP can preserve pacing of flows.

For our simulations, we have developed a framework that models a BJP router in detail. As for real network experiments, we have implemented BJP using NetFPGA - a PCI-based board with four GigE interfaces and controlled by a programmable FPGA designed at Stanford University [9].

[1]In this paper, we do not distinguish between routers and switches and use them interchangeably. All the relevant details remain the same in both contexts.

The high level of accuracy of NetFPGA boards (clock frequency of 125MHz) compared to ordinary Network Interface Cards (NICs), as well as their flexibility and simplicity of design make NetFPGAs the perfect platform for implementing and testing BJP. The low level structure of the board, makes it possible to emulate BJP with high accuracy in timing. Also, we can measure packet arrival and departure times with an error of 8ns, which is sufficient for our experiments (packet lengths in our experiments are about 12 μs). Prior work has shown that unexpected (and sometimes uncontrollable) delays in commercial NICs can make them unattractive for high accuracy experiments like this [3].

Our simulations and real network experiments show that BJP can preserve pacing in networks of routers as promised by the theoretical model. We observe that when smooth traffic passes through ordinary (round-robin) switches, the traffic becomes more and more bursty. However, with BJP switches the smoothness of traffic is preserved and no matter how many nodes the traffic goes through it remains smooth. In other words, if we use BJP in a real network, we can have the nice properties associated with paced traffic regardless of the topology, and multiplexing/demultiplexing effects.

2. BOUNDED JITTER POLICY

The basic idea behind BJP is to try to add a fixed delay to each packet at each router. Doing so will preserve the inter-arrival times of packets and therefore if the flows are smooth initially, they will remain smooth throughout the network. To do this, whenever a packet P arrives at a given router at time t, BJP will try to send the packet out at time $t+\delta$, for a predefined constant δ. Clearly, this is not always feasible due to contention: different packets arriving at different ingress ports at a given time t, and destined to the same egress port need to depart at the same time $t+\delta$, which is not possible.

To solve this problem, BJP slightly reduces the delay for one of the incoming packets so that there is no contention[2]: the first packet departs at time $t+\delta-1$ and the second one departs at $t+\delta$. At the same time, BJP marks the outgoing packets so that the reduction in delay is compensated later in subsequent routers. The goal is to keep the expected delay of a packet which has gone through N routers as close as possible to $N \times \delta$ despite of minor set backs resulting from contention. If a packet cannot be scheduled within the time interval $[t+\delta-\alpha, t+\delta]$ (where α is another predefined parameter) BJP simply drops the packet (Figure 1).

3. BJP DESIGN ON NETFPGA

Commodity NICs do not have the accuracy and flexibility required to implement BJP. Previous work has shown that ordinary NICs can lead to significant errors in packet injection times [6]. NetFPGA boards, on the other hand, have the flexibility required to implement BJP, and have sufficient timing accuracy for packet injections as well as for measurement of inter-arrival times. As a result, we have used NetFPGA for our implementation and real network evaluations.

[2]Here, we are assuming that time is slotted, and it takes one unit of time to transfer each packet.

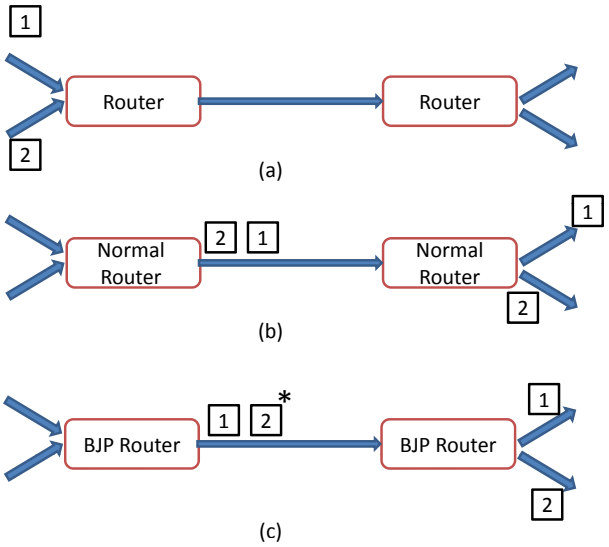

Figure 1: An example for a simple network with ordinary routers and BJP routers. (a) Two packets arrive simultaneously at two input ports. (b) Regular routers send packets out back-to-back immediately. (c) BJP routers try to delay both packets for δ units of time. Since there is congestion, packets are sent out back-to-back, but one of the is marked so that another router on the path delays it with $\delta-1$ rather than δ units of delay.

The reference router design for NetFPGA provides basic functionalities for receiving packets, passing them through a standard pipeline, forwarding packets to the appropriate egress port, and finally transmitting the packet (Figure 2(a)). We have started with this reference model, and have modified parts of it to enable BJP. The major differences between our design (Figure 2(b)) and the standard pipeline of NetFPGA (Figure 2(a)) are as follows.

1. In our design every packet is stored in the SRAM immediately after arrival. We use a pointer to the location of the packet as well as some extra information from the header for the rest of the process, i.e. routing and queueing in the switch is done through the pointer. In the NetFPGA reference design however, the entire packet passes through the pipeline. This simple modification gives us more time for the computation we need and allows us to keep up with the line rate.

2. In our design we log switch activities and send them to a measurement daemon through the PCI slot.

The first module in our design is the *input arbiter*: it continually checks ingress ports and as soon as a new packet arrives, the input arbiter records the exact arrival time and copies the payload of the packet to SRAM. Here, we can handle simultaneous arrivals on all four ingress ports, i.e. no incoming packet needs to wait for packets from other ports to be served. The input arbiter also creates an entry for the packet which includes a pointer to the payload, and passes it to the *scheduler* of the destination port.

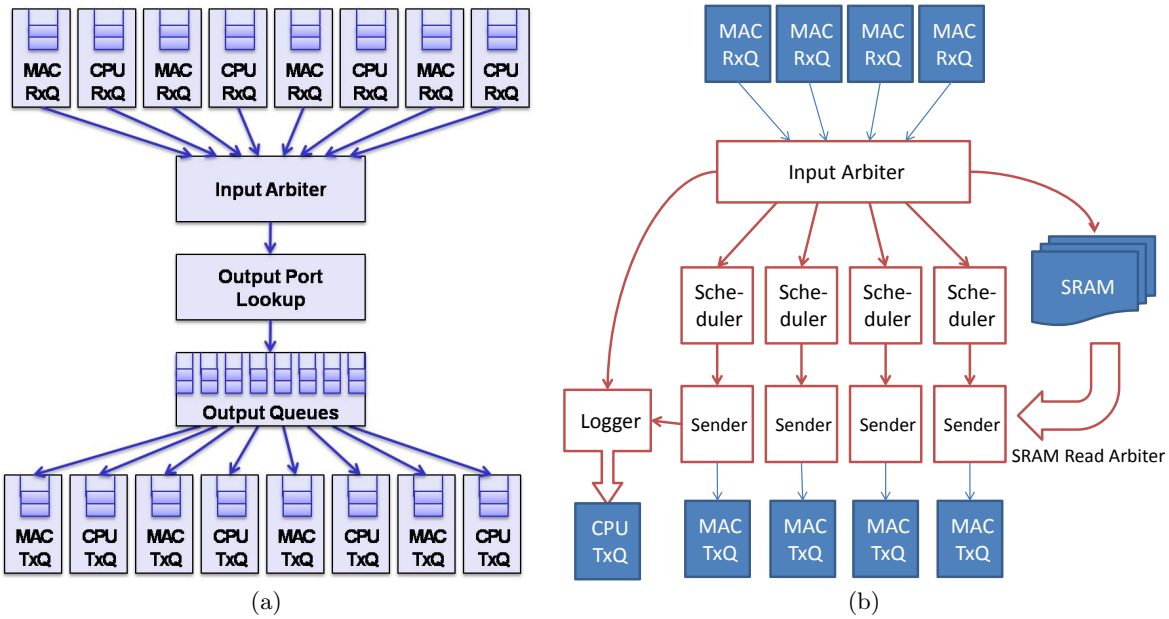

Figure 2: (a) NetFPGA reference pipeline for packets. (b) BJP design on NetFPGA

For each egress port there is a *scheduler* which handles the details of BJP procedure. Each scheduler keeps a schedule for packet departures. When a new packet arrives at the scheduler, it runs the BJP algorithm to find when the packet should depart, and if needed drops the packet. Well before the actual departure time of each packet, the scheduler sends the corresponding entries of departing packets to the *sender*. Each sender receives a packet pointer, reads the payload of the packet from SRAM and transmits the packet exactly at the scheduled time[3].

In our design we also have a *timer* module for slotting the time domain. It is connected to all other modules and provides current time and time-slot numbers for them.

For any packet arriving to the NetFPGA board, we create a log of their exact arrival and departure (in input arbiter and the sender module). These log entries are collected in the *logger* module. The logger module sends these logs to a measurement daemon through PCI.

Finally, we note that BJP needs to keep track of current lag for each packet. In our design, we use four bits of the destination MAC address for this purpose. In a real-world implementation we need to use some other fields in the header (including the options or some rarely used entries in the header).

4. SIMULATIONS

As we mentioned before, BJP has been studied theoretically only. In order to evaluate BJP in a more realistic setting, we have conducted numerous simulations as well as testbed experiments. For simulations we have developed a flexible framework in Matlab. Using this framework we can create an arbitrary topology of routers with and without BJP. We can also inject paced and non-paced traffic to the system, and study the evolution of packet inter-arrival times as flows pass through multiple routers. Even though this platform has various simplifications compared to real network components (routers, links, etc.), in Section 5.2 we show that there is no significant difference between our simulation results and what we observe in a real network of NetFPGA-based BJP routers.

4.1 Setup

We consider N routers which are connected back-to-back (as shown in Figure 3). Each router is assumed to have two ingress and two egress ports (we make this assumption to be able to compare our results with NetFPGA routers which have four ports). We consider a flow, which we call the *main flow*, going through all of these routers (entering the network at router 1, and leaving it at router N) and study inter-arrival times of packets after passing through each of the routers. As for the cross traffic, we consider $N-1$ flows: flow i enters the network at the ingress port of router i and leaves at egress port of router $i+1$. In all of the experiments, the main flow is a paced flow and cross flows are non-paced. Our goal is to study the impact of multiplexing/demultiplexing of these non-paced flows on the main paced flow.

Nodes: We use two types of routers for our simulations: BJP routers, and normal round-robin routers. BJP routers implement the BJP procedure explained before. Round-robin routers have unlimited input buffers, and serve packets in the input queues in a round-robin manner.

Links: In our simulations we use a link bandwidth of 1Gbps (the same as NetFPGA physical layer). Also, we set the time

[3]Since there are four senders in this design, we need a *SRAM read arbiter*, so all of the senders can read from SRAM simultaneously. Since the throughput of our SRAM memory is 8Gbps the BJP design can operate at the link rate for four 1Gbps Ethernet ports.

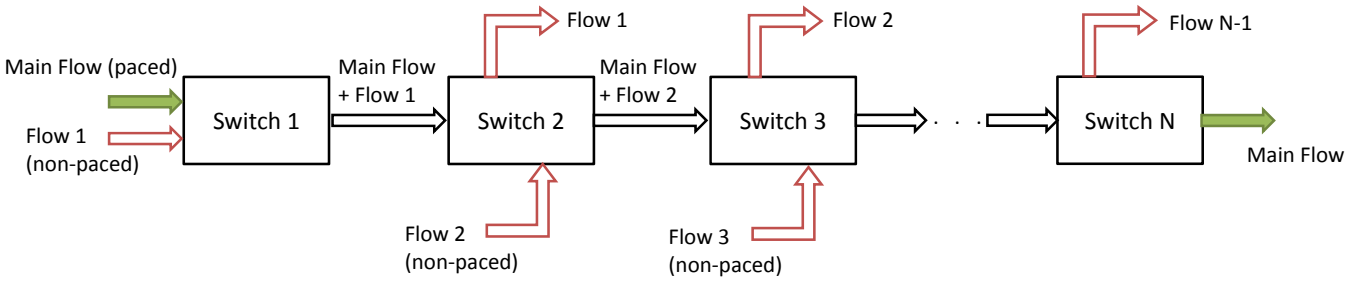

Figure 3: We use N routers in series for both simulations and testbed experiments. The main flow goes through all routers and is paced. The cross traffic is non-paced.

guard between two back-to-back Ethernet packets to 160ns[4].

Flows: In our simulations we use both paced and non-paced traffic. For paced flows, we set the inter-arrival times of all packets to a predefined constant. The average throughput of a paced flow is:

$$Throughput = \frac{PacketLength}{InterArrivalTime} \quad (1)$$

For non-paced flows, the inter-arrival times of packets are selected uniformly at random between one and a given constant γ. In this case, the mean inter-arrival time would be $(\gamma+1)/2$ time slots, and the throughput of the flow is:

$$Throughput = \frac{1}{(\gamma+1)/2} \quad (2)$$

We set packet lengths to 1504 bytes (18 bytes less than Maximum Ethernet packet size) equal to 12192ns[5]. Each time slot in BJP routers is set to 12192ns.

Figure 4(a) illustrates the inter-arrival times of the packets at the source for paced and non-paced traffic. Both paced and non-paced flows have a throughput of 43% in our simulations.

4.2 Simulation Results

To evaluate BJP we run two sets of simulations with and without BJP routers on the topology presented in Figure 3. We measure packet inter-arrival and delay times for all of the packets in both cases and compare the results to see the impact of BJP on keeping the traffic smooth.

As predicted theoretically, when the paced flow passes through the network of round-robin routers, it loses its paced shape. Figure 4(b) depicts the CDF of inter-arrival times of packets at the source and after going through the network (i.e. at the destination). We can see that packets have a smooth injection (uniform inter-arrival times) at the source, but going through multiple round-robin routers, inter-arrival times change significantly and the main flow is no longer smooth.

[4]For Ethernet packets we have 7 bytes preamble + 1 byte start of frame delimiter + 12 bytes inter-frame gap which is 20 bytes, or 160ns on 1Gbps link.

[5]1504 bytes plus the guard time.

When we use BJP routers however, the step-shape in the CDF of inter-arrivals is preserved (Figure 4(c)). The small change in the shape of the step function is due to the assumption that time is slotted, and the fact that the source traffic generator is not synchronized with the BJP routers. In other words, packet arrivals at the source do not necessarily align with the boundaries of time slots. To better illustrate this, imagine a case which the space between two packets at the source is equal to 2.33 time slots. Looking at the packet arrival times from BJP's perspective, some of the packets have a space of two slots and some have three slots between them. This error in measuring packet arrival times can happen only at the first BJP enabled router. Since the slot sizes at all BJP routers in the network are equal, there is no change in measured space between packets throughout the network[6].

The conclusion is that other than minimal changes due to synchronization, the inter-arrival times of packets are preserved under BJP. In other words, BJP preserves the smoothness of flows.

We also take a closer look at the end-to-end delay of packets. In a network with round-robin routers, the end-to-end delay varies significantly for different packets (Figure 4(d)). This delay depends on the state of other flows in the network and the throughput of links on the path, and the variations grow as we go through more routers. However, in the network of BJP routers, packets have a very small jitter in their end-to-end delays. The differences in the end-to-end delay here are due the synchronization issue explained before, and are bounded by the length of one time slot. Clearly, having a small jitter is a very desirable property for various applications, and our simulations show that BJP routers have this property.

5. TESTBED EXPERIMENTS

In order to evaluate BJP in a real setting, we created a testbed of BJP routers. In order to have a meaningful comparison, we have kept the setup the same as simulations.

5.1 Setup

Topology: For our testbed experiments, we use the same topology as Figure 3. Each router has two input ports and two output ports. As in the case of simulations, we have

[6]Except for the error causing by clock frequency mismatch in network router, which is negligible.

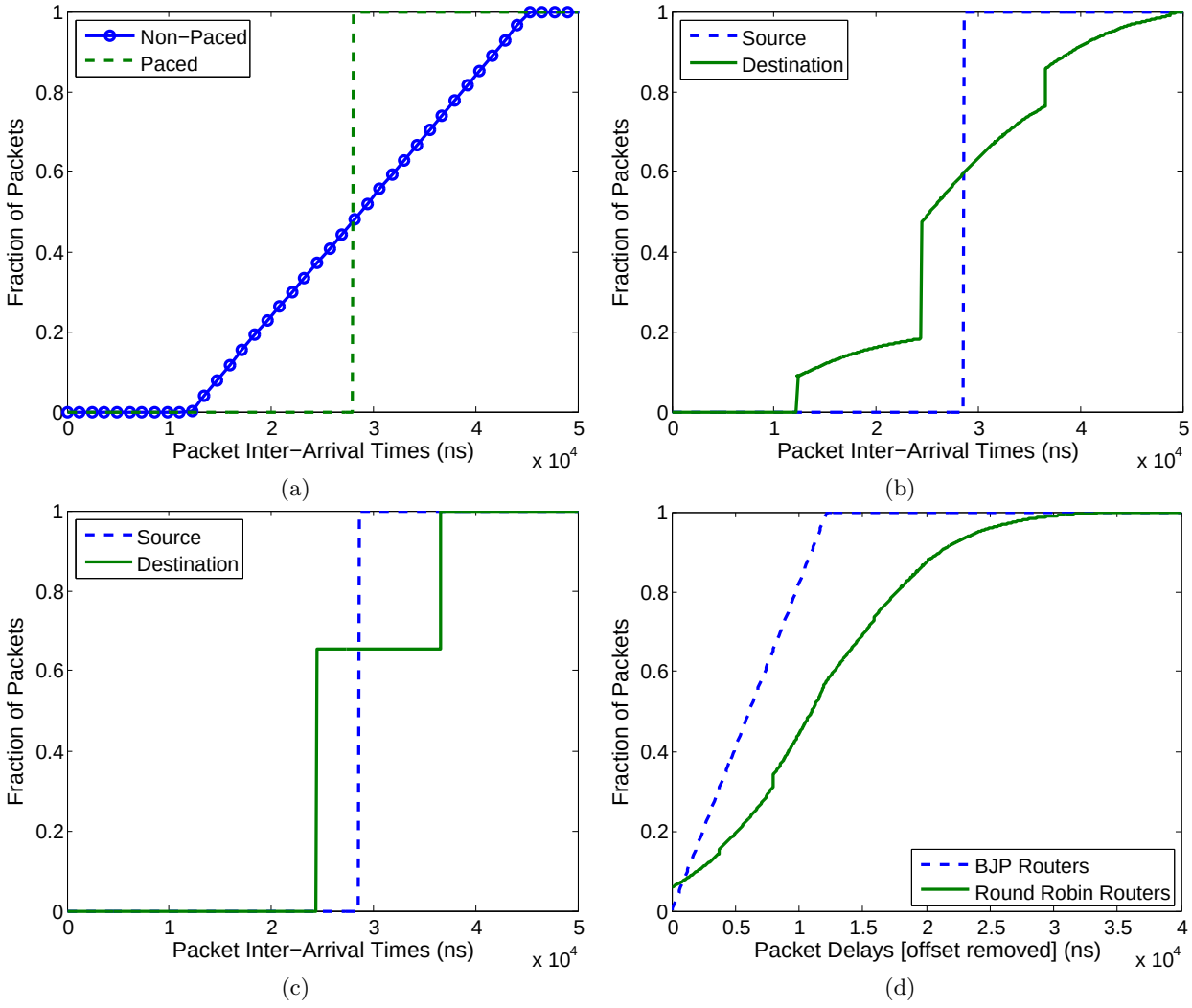

Figure 4: Simulations: (a) CDF of packet inter-arrival times for paced and non-paced traffic at the source. (b) CDF of packet inter-arrival times at the destination with round-robin routers. (c) CDF of packet inter-arrival times at the destination with BJP routers (d) End-to-end packet delays for both BJP and round-robin routers.

a main flow which is paced, and cross traffic which is non-paced.

BJP Switches: We Use Dell 2951 servers each with one NetFPGA PCI card. We load the BJP design which is discussed above on NetFPGAs. On each board, two ports are used as input and two are used as output ports.

Normal Switches: To run the emulations using normal switches, we use commercial 1Gigabit switches (NETGEAR ProSafe GS105 [10]). Since our emulation is open-loop, we need to initialize forwarding tables in all switches. To this end, we send a packet with the desired source MAC address to a non-existent destination MAC address. The initializer packet propagates among all the switches in the network, setting up the appropriate forwarding table entries on all the switches.

Traffic: We generate the traffic we need using NetFPGAs as well. We use Stanford Packet Generator (SPG)[9], which is more accurate than a normal NIC. We can set packet inter-arrival times with a resolution of 8ns (packet length is $12\mu s$).

Similar to the simulations we need one paced flow with constant space between packets, and several non-paced cross traffic flows. We measure the inter-arrival times of packets and the CDFs are illustrated in Figure 5(a).

Measurements: We also use NetFPGA as our accurate measurement tool. In networks with BJP routers, we capture logs generated by the logger module, which contain accurate arrival and departure times of each packet, along with information required to identify each packet.

For networks without BJP routers, we use a modified version of BJP design as a measuring tool. By changing routing table in a BJP router, BJP scans all incoming ports, generates the appropriate log entries for arrivals, and then drops all the packets. We use this modified design as our generic measurement tool.

5.2 Experiment Results

Similar to simulations, we setup a network with BJP routers, and a network using commodity switches in our testbed. We inject paced and non-paced traffic for each case. We record packets inter-arrival times and delays as they pass through switches, and we study the evolution of packet inter-arrival times.

For commodity switches, we plot the CDF of packet inter-arrival times for the pace flow at the source and after passing through one and three switches (Figure 5(b)). We can see that going through ordinary switches the shape of the traffic changes. In fact, as the flow passes through more switches, the traffic becomes more bursty and the shape of the CDF spreads over a wider range.

We repeat the same experiment with BJP routers. Here, we plot the inter-arrival times of packets at the source, and the destination after going through four BJP routers (Figure 5(c)). We can see that the CDF of inter-arrival times for source and destination remains intact other than for the changes made due to time slot alignments (as described in previous section). Here, if we look at inter-arrival times after aligning packets with time slots at the source, and the inter-arrival times at the destination, the two curves match very closely. In other words, BJP preserves the inter-arrival times to a very good extent.

Finally, as in the case of simulations, we study the end-to-end delay of packets going through the system. Figure 5(d) presents the end-to-end delay for BJP routers and commodity switches. Similar to simulations, we have linear variations in the end-to-end delay observed by BJP packets, which is due to alignment of packets with time slots. This variation is bounded by the length of one time slot, and does not grow with the number of routers on the path. However, in the case of ordinary switches, the variations in end-to-end delay increases as the flow passes through more routers.

6. CONCLUSION AND FUTURE WORK

In this paper, we studied the impact of deploying BJP in routers using simulations as well as testbed experiments. We use a framework that we have created for our simulations, and we use an implementation of BJP on NetFPGAs for our experiments. In both cases, we show that unlike commodity switches, BJP preserves packet inter-arrival times – and therefore traffic smoothness – regardless of the number of switches on the path. Given the simplicity of BJP and its good performance as observed in this paper, we believe BJP is a viable solution for preserving traffic shape in networks with a lot of multiplexing and demultiplexing.

7. REFERENCES

[1] A. Aggarwal, S. Savage, and T. Anderson, *Understanding the performance of TCP pacing,* Proceedings of IEEE INFOCOM, 2000.

[2] L.L.H. Andrew, S.V. Hanly, and R.G. Mukhtar, *CLAMP: Maximizing the performance of TCP over low bandwidth variable rate access links*, Technical Report, 2004-02-01, CUBIN, University of Melbourne, 2004.

[3] N. Beheshti, Y. Ganjali, M. Ghobadi, N. McKeown, and G. Salmon, *Experimental Study of Router Buffer Sizing,* Proceedings of IMC, Greece, 2008.

[4] N. Beheshti, Y. Ganjali, A. Goel, and N. McKeown, *Obtaining High Throughput in Networks with Tiny Buffers*, Proceedings of IEEE IWQoS, Netherlands, 2008.

[5] M. Grossglauser, S. Keshav, *On CBR Service*, Proceedings of IEEE INFOCOM, 1996.

[6] G. Salmon, M. Labrecque, M. Ghobadi, Y. Ganjali, and G. Steffan, *NetFPGA-based Precise Traffic Generation,* Proceedings of NetFPGA Developers Workshop, Stanford, CA, 2009.

[7] R. Srikant, *The Mathematics of Internet Congestion Control,* Birkhauser Boston, 2003.

[8] D.X. Wei, P. Cao, S.H. Low, C. EAS, and S. CS, *TCP pacing revisited*, Proceedings of IEEE INFOCOM, 2006.

[9] http://www.netfpga.org/

[10] http://www.netgear.com/Products/Switches/DesktopSwitches/GS105.aspx

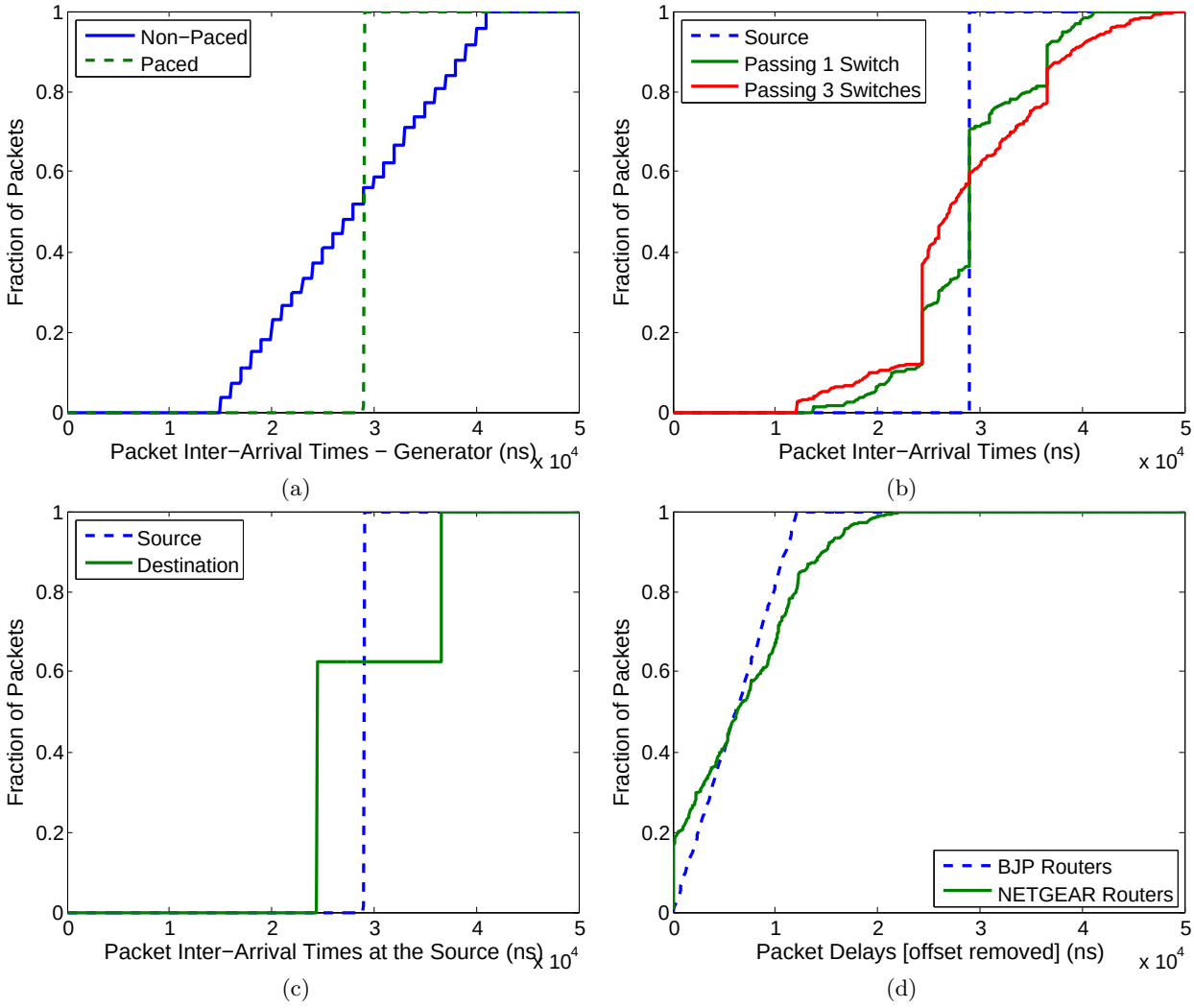

Figure 5: Testbed experiments: (a) CDF of packet inter-arrival times for paced and non-paced traffic at the source. (b) CDF of packet inter-arrival times at the destination with commodity switches. (c) CDF of packet inter-arrival times at the destination with BJP routers. (d) End-to-end packet delays for both BJP and regular switches.