

A Network Emulator for High-Precision Latency Forwarding over Packet-Programmable Networks (Demo Paper)

Alexander Clemm, Toerless Eckert
Futurewei
Santa Clara, California/USA
{alex | toerless.eckert}@futurewei.com

Abstract—In a companion paper [1], we introduced Latency-Based Forwarding (LBF) as a new approach to achieving stringent end-to-end latency objectives with precisely specified lower and upper bounds. In order to accomplish this, a new forwarding protocol is used that can carry latency objectives as part of packet metadata and take differentiated actions at network nodes depending on a packet’s remaining latency budget, its destination, and the latency encountered so far. LBF is intended to enable new emerging network applications such as Industry 4.0, haptic applications, and holographic-type communications which require support for corresponding high-precision network latency services that are not possible with current best-effort Internet or QoS mechanisms.

This demo shows a near real-time, lightweight network emulator for LBF. Individual network nodes are emulated by separate cores on a host. As part of the demo, we vary the end-to-end latency objectives of multiple competing simultaneous flows and compare minimum, maximum, average packet latencies across different scenarios. This demonstrates the behavior and effectiveness of LBF and how it affects flows and packets.

Keywords—High precision networking, latency, SLOs, QoS, LBF, BPP, New IP

I. INTRODUCTION

Many future networking applications will place high demands on service levels that networks must deliver, specifically precise end-to-end latency objectives that must be met [2][3].

For example: (1) Industry 4.0 applications include industrial and robotic automation, in which robotic machinery is controlled remotely. This requires very precise synchronization and spacing of control and telemetry data with near-deterministic latency in which packets not only do not exceed a certain latency but are also not delivered sooner. (2) Haptic applications of the Tactile Internet require end-to-end networking latencies with well-defined upper bounds on the order of 5 ms or less in order to provide the illusion of remotely “touching” something. Without it, the ability to remotely operate machinery on the basis of haptic feedback is lost. (3) Holographic-Type Communications (HTC) and VR applications all require the ability to rapidly adapt streamed contents based on changes in user position or viewing angle.

Well-defined end-to-end network latency is required to guarantee certain levels of user experience.

Current Internet technology is based on the Best Effort principle. QoS mechanisms provide for the ability to prioritize traffic, reserve resources, and apply admission control in order to provide better service levels for applications that need it. However, while these mechanisms allow for optimizations of latency, they do not change the fact that fundamentally end-to-end latency still needs to be accepted as delivered by the network, as opposed to have the network deliver on a given end-to-end latency as required by an application. Also, existing mechanisms that attempt to optimize latency typically focus on constant bit rate (CBR)-type of applications and do not accommodate variable bit rates (VBR) without incurring significant waste in terms of reserved resources. Finally, the focus in the past has been on reducing maximum latency and “not-to-exceed” objectives, not on delivery of packets that is also ensured to be no sooner than a given minimum latency.

In order to address these gaps, we have introduced a novel approach called Latency-Based Forwarding (LBF) in a companion paper [1]. LBF is based on the idea that for any given packet with a given end-to-end latency objective, a networking device takes a differentiated action that can “slow down” or “speed up” the packet as needed (within certain bounds). To deliver this functionality, each node that receives a packet computes a local latency budget (maximum and minimum) based on the end-to-end latency objective, on the latency that was incurred by the packet up to this point, and on the expectation of the remaining path latency and number of hops to the packet’s destination. The local scheduling / forwarding decision for the packet is then based on this budget. Latency metadata, including the latency objective, can be carried as part of the packet itself. In cases where a requested latency is physically impossible to meet, packets can be dropped early to reduce congestion, instead of forwarding them all the way to the destination only to have them dropped there because they are late.

Advantages of LBF include the fact that it can be used to support both “in-time” (not to exceed) and “on-time” (i.e. right on target) latency objectives, that it does not require admission control, and that it supports applications with variable bit and packet rates. In addition, LBF is very elastic in the sense that it is able to react to unforeseen variations in latency along a path

for packets that are in flight, adjusting per-hop latency as needed. Finally, LBF allows stateless forwarding, in which the forwarders do not need to maintain per-flow-state, unlike other solutions that require awareness of per-flow awareness on forwarders, which may impact scalability.

LBF relies on Big Packet Protocol (BPP) [4], a novel network programming framework part of a new framework for packet-based networking called New IP. BPP allows to program custom flow behavior from the edge of the network using metadata and commands that are carried as part of packets themselves. This allowed us to not only prove the viability of LBF, but to demonstrate that the network programming framework can indeed be used to support applications with real-time, high-precision latency networking demands solely with more intelligent per-packet forwarding algorithms, but without the need for custom control software in networking devices or controllers.

II. PROOF-OF-CONCEPT IMPLEMENTATION AND DEMO SETUP

To validate the LBF algorithm, we have built a lightweight Latency-Based Forwarding Emulator (LBFEM) as a Proof-of-Concept (PoC), supporting a subset of BPP as required for LBF. The PoC allows quick experimentation with different algorithms in small topologies through implementation and test runs; its code is publicly available [5] and can be easily adopted and expanded.

The implementation consists of one Linux application for the forwarder and one for the sender. The demo runs the entire network within a single server. UDP ports are used as link-layer interfaces. UDP payloads represent the BPP packet. It consists of the required parts of IP headers to hop-by-hop forward a packet and the BPP Block to perform LBF processing. Forwarders use separate threads for each ingress and egress interface. An ingress thread receives and routes a packet to an output interface and enqueues it into that interface's LBF queue. An egress thread dequeues and sends packets, emulating outgoing interface speeds. By assigning threads for each interface to separate CPU cores, the code can as necessarily use busy loops (so called poll-mode) for higher timing accuracy. At the end of each test run (e.g.: 10 seconds), logs for each packet that passed through an interface are written for analysis.

The emulation performs near real-time at 100 Mbps line rate speed with μsec accuracy for LBF processing and measurement. This is achieved through elimination of almost all Linux system call interference. It constitutes a lightweight starting point for fast experimentation with forwarding plane functionality when high accuracy time measurements are required.

Fig.1 shows a screen shot of the demo for a very simple validation case for LBF and the emulator: 5 competing flows are passed through a single emulated LBF forwarder. Logs with detailed latency data for each flows packet are captured and analyzed at the end of the test run to calculate per flow average, minimum, and maximum packet latency. Latency of SLOs are easily changed for test runs, allowing to show LBF forwarding conforms to desired SLOs, how it impacts fairness

between flows, and how it compares with best effort forwarding latency.

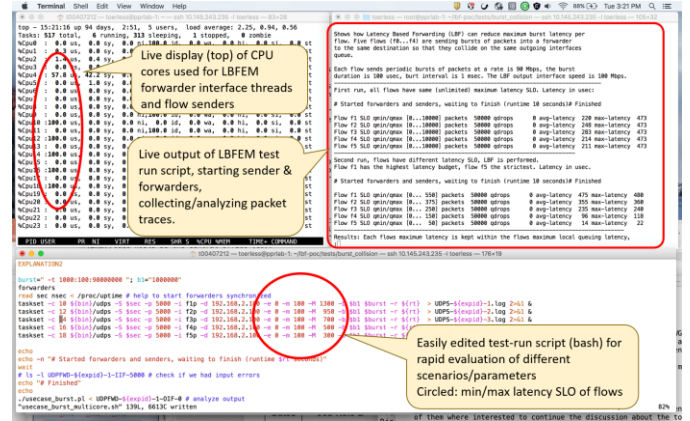


Fig. 1. Single forwarder congestion, best effort and LBF flows compared

Fig. 2 shows the topology as validated for companion paper [1]. This too can be shown in the demo. It demonstrates how the emulation supports multiple forwarders, and how LBF introduces “elastic” multi-hop latency management. Flow 1 from Sender 1 passes through 4 LBF queues 1, 2, 3, 4. On each queue, it competes with traffic from another flow that will be routed by the following forwarder to a different interface than Flow 1, so that for each hop the queue buildup can be controlled separately by setting up the competing traffic flow sender parameters. The receivers are shown with dashed lines because no actual receiver applications are started as only the per-packet statistics generated by the forwarders are of interest.

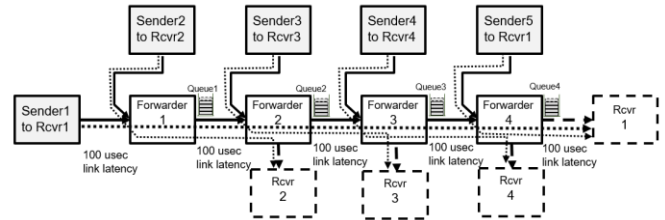


Fig. 2. Emulated network topology

REFERENCES

- [1] A. Clemm, T. Eckert: “High-Precision Latency Forwarding over Packet-Programmable Networks.” IEEE/IFIP Network Operations and Management Symposium (NOMS 2020), Budapest, Hungary, April 2020.
- [2] “Network 2030 – A Blueprint of Technology, Applications and Market Drivers Towards the Year 2030 and Beyond.” White Paper, accessible at <https://extranet.itu.int/sites/itu-t/focusgroups/net-2030/SitePages/White%20Paper.aspx>, ITU-T FG-NET-2030, May 2019.
- [3] ITU-T FG-NET2030: “New Services and Capabilities for Network 2030: Description, Technical Gap and Performance Target Analysis”. FG-NET2030 document NET2030-O-027, November 2019.
- [4] R. Li, A. Clemm, U. Chunduri, L. Dong, K. Makhijani: “A New Framework and Protocol for Future Networking Applications.” ACM SIGCOMM Workshop on Networking for Emerging Applications and Technologies (NEAT), Budapest, Hungary, August 2018.
- [5] <https://github.com/network2030/lbf-poc>