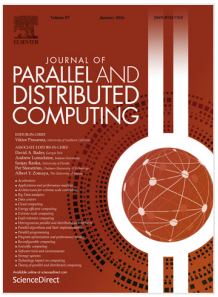
Accepted Manuscript



Heterogeneous packet processing in shared memory buffers

Patrick Eugster, Kirill Kogan, Sergey I. Nikolenko, Alexander V. Sirotkin

PII: S0743-7315(16)30087-9
DOI: http://dx.doi.org/10.1016/j.jpdc.2016.07.002
Reference: YJPDC 3527

To appear in: *J. Parallel Distrib. Comput.*

Received date: 10 November 2015
Revised date: 29 April 2016
Accepted date: 6 July 2016

Please cite this article as: P. Eugster, K. Kogan, S.I. Nikolenko, A.V. Sirotkin, Heterogeneous packet processing in shared memory buffers, *J. Parallel Distrib. Comput.* (2016), http://dx.doi.org/10.1016/j.jpdc.2016.07.002

Heterogeneous Packet Processing in Shared Memory Buffers

Patrick Eugster, Kirill Kogan, Sergey I. Nikolenko, Alexander V. Sirotkin

***Abstract*—Packet processing increasingly involves heterogeneous requirements. We consider the well-known model of a shared memory switch with bounded-size buffer and generalize it in two directions. First, we consider unit-sized packets labeled with an output port and a processing requirement (i.e., packets with heterogeneous processing), maximizing the number of transmitted packets. We analyze the performance of buffer management policies under various characteristics via competitive analysis that provides uniform guarantees across traffic patterns [10]. We propose the Longest-Work-Drop policy and show that it is at most 2-competitive and at least $\sqrt{2}$-competitive. Second, we consider another generalization, posed as an open problem in [19], where each unit-sized packet is labeled with an output port and intrinsic value, and the goal is to maximize the total value of transmitted packets. We show first results in this direction and define a scheduling policy that, as we conjecture, may achieve constant competitive ratio. We also present a comprehensive simulation study that validates our results.**

I. Introduction

The modern network edge is required to perform tasks with heterogeneous complexity, including, to list just a few, advanced VPN services, deep packet inspection, firewalling, and intrusion detection. This trend is further exacerbated by the advent of new network management models like Software-Defined Networking, which create new opportunities for advanced services. Each of these services may require to consider various characteristics (e.g., amount of processing or packet values) to achieve efficient implementation at the network processor and may present new challenges for traditional architectures, resulting in performance and implementation issues. Hence, the way how packets are processed may significantly affect throughput. For example, increasing per-packet processing requirements for some flows can increase congestion even for traffic with relatively modest burstiness characteristics.

This work is an extended version of [16]. This version contains rewritten and extended Sections I, II, and V, new Figure 4, Tables I and II, new examples that illustrate the proposed algorithms.

P. Eugster is with Departments of Computer Science of Purdue University and TU Darmstadt; e-mail: peugster@cs.purdue.edu. His work was partially supported by Cisco Systems through grant "A Fog Computing Architecture", the German Research Foundation through project "Multi-Mechanism Adaptation for Future Internet" and the European Research Council through project "Lightweight Verification of (Distributed) Software".

K. Kogan is with IMDEA Networks Institute; e-mail: kirill.kogan@imdea.org.

S.I. Nikolenko is with the National Research University Higher School of Economics, St. Petersburg, and with Steklov Mathematical Institute at St. Petersburg; e-mail: sergey@logic.pdmi.ras.ru. His work was partially supported by the Government of the Russian Federation grant 14.Z50.31.0030.

A.V. Sirotkin is with National Research University Higher School of Economics, St. Petersburg; e-mail: avsirotkin@hse.ru.

A. Shared Memory Switch

The main functionalities of a network element include receiving packets on ingress ports, applying specific policies to them, identifying their destination ports, and sending them out through egress ports. When application-induced traffic bursts create an imbalance between incoming and outgoing packet rates to a given port, packets must be queued in the switch packet buffer. The available queue size on a port determines the port's ability to hold the packet until the egress port can emit it. When buffer queue entries are exhausted, packets are dropped, resulting in poor performance. The allocation and availability of the switch's buffer resources to its ports — determined not only by the buffer's size but also by the buffering architecture of the network element.

Overprovisioning in terms of buffer capacity at each network node to absorb bursty behavior is not viable, as networks do not have unlimited resources; quite conversely, cloud data centers can only scale out as fast as the effective per-port cost and power consumption. These factors, in turn, are driven by the chosen buffering architecture. The shared memory switch allows to absorb traffic bursts in the best way since the whole buffer can be utilized by a same output port if needed. Being an actual choice in practice [14] we focus our efforts in this paper on this type of buffer architecture.

The paradigm of shared memory switch in general is very flexible since it can support both complete sharing, where the same buffer serves all output ports, and complete partition, where each output port gets a *de facto* dedicated buffer. Complete sharing utilizes the entire buffer space but can hamper fairness: a single output port may monopolize shared memory and drop packets dedicated to other output ports. On the other hand, complete partitioning ensures fairness but may lead to significantly underutilized buffer space, losing more packets in case of congestion. But how to get the best of both worlds?

B. Throughput Optimization

There are two major tasks performed by network elements: (1) changing the properties of single packets (e.g., recoloring, encapsulation) and (2) changing the properties of packet streams (rate-limiting, shaping); the latter may require buffers. Devising a buffering architecture and its management is a fundamental problem in network switch design. To accommodate for network size, number of transport requests, and efficient reuse of network infrastructure, the Internet currently implements "best-effort" servicing that during congestion prioritizes some types of traffic over others to optimize desired objectives.