Report Number: WUCSE-2006-62

2006-01-01

Virtualization for a Network Processor Runtime System

Brandon Heller, Jonathan Turner, John DeHart, and Patrick Crowley

The continuing ossification of the Internet is slowing the pace of network innovation. Network diversification presents one solution to this problem, by virtualizing the network at multiple layers. Diversified networks consist of a shared physical substrate, virtual routers (metarouters), and virtual links (metalinks). Virtualizing routers enables smooth and incremental upgrades to new network services. Our current priority for a diversified router prototype is to enable reserved slices of the network for researchers to perform repeatable, high-speed network experiments. General-purpose processors have well established techniques for virtualization, but do not scale efficiently to multi-gigabit speeds. To achieve these speeds, we... Read complete abstract on page 2.

Virtualization for a Network Processor Runtime System

Brandon Heller, Jonathan Turner, John DeHart, and Patrick Crowley

Complete Abstract:

The continuing ossification of the Internet is slowing the pace of network innovation. Network diversification presents one solution to this problem, by virtualizing the network at multiple layers. Diversified networks consist of a shared physical substrate, virtual routers (metarouters), and virtual links (metalinks). Virtualizing routers enables smooth and incremental upgrades to new network services. Our current priority for a diversified router prototype is to enable reserved slices of the network for researchers to perform repeatable, high-speed network experiments. General-purpose processors have well established techniques for virtualization, but do not scale efficiently to multi-gigabit speeds. To achieve these speeds, we employ network processors (NPs), typically consisting of multicore, multi-threaded processors with asymmetric, heterogeneous memories. The complexity and lack of hardware thread isolation in NP's, combined with a lack of simple programming models, creates numerous challenges for effective sharing between metarouters. In this paper, we detail strategies for enabling NP virtualization at the link, memory, and processor levels, to better enable a research infrastructure for network innovation.

2006-62

Virtualization for a Network Processor Runtime System

Authors: Brandon Heller, Jonathan Turner, John DeHart, Patrick Crowley

Corresponding Author: bdh4@arl.wustl.edu

Abstract: The continuing ossification of the Internet is slowing the pace of network innovation. Network diversification presents one solution to this problem, by virtualizing the network at multiple layers. Diversified networks consist of a shared physical substrate, virtual routers (metarouters), and virtual links (metalinks). Virtualizing routers enables smooth and incremental upgrades to new network services. Our current priority for a diversified router prototype is to enable reserved slices of the network for researchers to perform repeatable, high-speed network experiments. General-purpose processors have well established techniques for virtualization, but do not scale efficiently to multi-gigabit speeds. To achieve these speeds, we employ network processors (NPs), typically consisting of multicore, multi-threaded processors with asymmetric, heterogeneous memories. The complexity and lack of hardware thread isolation in NP's, combined with a lack of simple programming models, creates numerous challenges for effective sharing between metarouters. In this paper, we detail strategies for enabling NP virtualization at the link, memory, and processor levels, to better enable a research infrastructure for network innovation.

Type of Report: Other

Virtualization for a Network Processor Runtime System

Brandon Heller, Jonathan Turner, John DeHart, Patrick Crowley
Washington University in St. Louis
Campus Box 1045
St. Louis, Missouri 63130-4899
bdh4@arl.wustl.edu, jon.turner@wustl.edu, jdd@arl.wustl.edu, pcrowley@wustl.edu

ABSTRACT

The continuing ossification of the Internet is slowing the pace of network innovation. Network diversification presents one solution to this problem, by virtualizing the network at multiple layers. Diversified networks consist of a shared physical substrate, virtual routers (metarouters), and virtual links (metalinks). Virtualizing routers enables smooth and incremental upgrades to new network services. Our current priority for a diversified router prototype is to enable reserved slices of the network for researchers to perform repeatable, high-speed network experiments. General-purpose processors have well-established techniques for virtualization, but do not scale efficiently to multi-gigabit speeds. To achieve these speeds, we employ network processors (NPs), typically consisting of multi-core, multi-threaded processors with asymmetric, heterogeneous memories. The complexity and lack of hardware thread isolation in NP's, combined with a lack of simple programming models, creates numerous challenges for effective sharing between metarouters. In this paper, we detail strategies for enabling NP virtualization at the link, memory, and processor levels, to better enable a research infrastructure for network innovation.

Categories and Subject Descriptors

C.2.6 [**Computer Systems Organization**]: Internetworking – routers, D.3.4 [**Programming Languages**]: Processors - run-time environments C.1.3 [**Processor Architectures**]: Other Architecture Styles – heterogeneous (hybrid) systems

General Terms

Performance, Design, Experimentation.

Keywords

Multi-core processor, diversified network, metarouter, runtime system, programming model, network processor.

1. INTRODUCTION

An ideal testbed for network architecture research would be freely accessible, provide realistic user traffic, process packets at high speeds, and enable deployment of networks not based on the Internet Protocol (IP). The leading testbed for network research is currently PlanetLab, a series of PC-based routers connected over the Internet [3]. Its use of general-purpose processors limits network speeds and its ability to provide hard resource guarantees for repeatable experiments, but its distributed-service model is flexible and useful for network research. Moreover, PlanetLab is built on IP, making it difficult or impossible to investigate non-IP networks. *Diversified networking* is our vision for a network research testbed, as shown in Figure 1, and described in more detail in [1] and [2]. Diversified networks define a new substrate layer between the datalink and transport layers to enable flexibility both above and below. *Metarouters* (MRs) are software-based virtual routers which communicate through *metalinks*, or virtual links. Metarouters and metalinks are hosted on a shared physical substrate, which includes physical routers and physical links; the combination of these elements forms a *metanetwork*. The diversified router represents one candidate backbone routing node for the upcoming GENI project, a planned large-scale National Science Foundation network research platform [1]. In this paper, we discuss the design of a diversified router, starting with relevant high-level design goals:

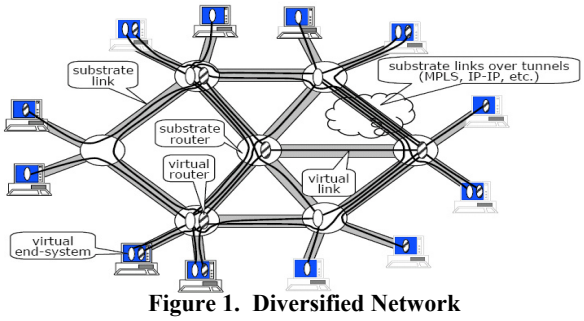

Figure 1. Diversified Network

- **Isolation.** *Virtualization* should be provided, where each metarouter has an exclusive, protected view of hardware resources such as link, processing, and memory bandwidth. The system must provide real-time guarantees to provisioned metarouters and fairly share excess system capacity.
- **Speed.** The router should handle 10 Gb/s links, over minimum- and average-size packets, with average delay of less than 1 ms.
- **Scalability.** The router should support around 100 metarouters, through both additional processors and a runtime system that enables single-processor sharing.
- **Generality.** Metarouters with a range of processing, memory, and synchronization requirements should be supported.
- **Openness.** To be an effective research platform, all system details must be open to experimenters, and most functionality must be modifiable. This requirement prevents the use of commercially available routers.

Our goal to deliver an operational system motivates the choice of Network Processors (NPs), the only **commercially available** option to support flexible processing at 10Gb/s rates. General-

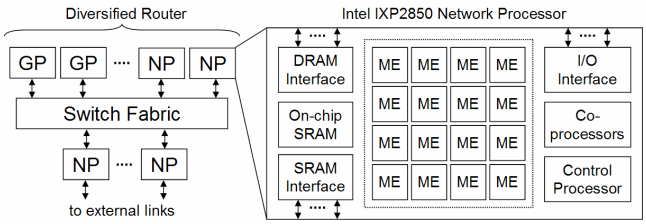

Figure 2. Diversified Router

purpose processors (GPs), while slower, enable virtually unlimited code complexity and can run a standard PlanetLab OS. We expect most metarouters to split their processing work in a way that plays to the strengths of each processor type, by performing fast-path, common-case tasks on an NP and exception and control processing on a GP. The router will use the Intel IXP2850 NP, shown in Figure 2, which has features representative of most NPs, including 16 simplified data processors called microengines (MEs), a high-bandwidth memory system, and on-die coprocessors to accelerate common networking operations. In the sections to follow, we explain the importance and difficulty of sharing an NP, summarize related work, and present runtime system designs for a diversified router.

2. CHALLENGES IN SHARING AN NP

A one-to-one assignment of metarouter to NP is the simplest design option, but has a number of disadvantages. For example, such a system would limit the total number of metarouters. Each NP would have far-from-optimal utilization, and the resulting system would be impractically expensive for large numbers of MRs. We argue that consolidating multiple MRs onto one NP is **essential** for a flexible, cost-effective diversified router.

Unfortunately, characteristics of NP hardware and high-speed networking present challenges to successful NP sharing. The first issue is the **lack of hardware mechanisms** to provide isolation between MRs. Features such as page protection and kernel/user separation would be required to support a typical OS, and the NP's shared memory architecture cannot enforce fair bandwidth use. Even with these features, a typical OS would be optimized for interactivity, rather than the hard resource guarantees and aggregate throughput we desire. The **limited code space** available for each ME restricts scalability. The presence of cooperative rather than preemptive multitasking complicates sharing a single ME between MRs, especially for metarouters with different processing and memory patterns. High-speed networking creates **demanding real-time constraints**. At full line rates, small packets leave few processing cycles available, yet next-generation network services carry higher processing requirements. Finally, existing IXP programming models expose architectural details and assume a single application. Using these models, unmodified, adds complexity to the runtime system. Fortunately, new work has confronted many of these challenges.

3. RELATED WORK

Recent IXP runtime systems have demonstrated a number of the properties desirable in a diversified router. A group of runtime systems support multiple virtual routers on a single NP [10][11][12][13][14][15]. Another project showed that programs dynamically loaded on NPs could have low overhead [16]. The performance of scheduling techniques on NPs has been investigated [6][7]. Other projects have demonstrated the benefits of adapting the processing configuration to the current packet distribution [8][9]. Memory access isolation and bandwidth management schemes have been shown [5], and there has been research into NP-specific programming models as well [17][18].

The good news is that techniques to achieve the design goals exist; the bad news is that no single runtime system fits our goals. Most significant is the **lack of scalability** with existing runtime systems. Another issue is isolation; none of the systems can provide per-metarouter **resource guarantees** for metarouters with varying requirements. Few of the systems have any provision for safely loading **dynamic code** onto a running system. Finally, and most crucially, many of the systems are "simulation-only," and have not been run on **real systems** with hardware constraints, yet system architecture can have unforeseen performance effects. **Our design will combine existing NP sharing techniques, in a new way, for a different set of design goals.**

4. TECHNIQUES FOR SHARING AN NP

Three main categories of resources that must be shared include (1) link resources, both incoming and outgoing, (2) memory resources, both space and bandwidth, and (3) processing resources. The design space of a system to share these resources is imposingly large, so we only provide a brief discussion of important design options. For more details, see [4]. The following virtualization techniques ensure that multiple threads of execution on multiple processing cores can safely and fairly access shared resources.

- **Memory Access.** The framework must ensure that memory access is safe. Static analysis is the lowest-overhead option, but cannot be used to prove the safety of any address computed at run-time. Dynamic checks with segments (legal memory ranges) or pages (reserved memory regions), implemented through macros, enable safe indirect access [5].
- **Resource Ordering.** Packets in one flow on a router must leave in the same order they enter. In a multi-core processor like the IXP, ordering can have a significant effect on efficiency, since the time required to access shared memory may be longer than the time allocated to process a packet. Multithreading can hide this latency, but only for a small number of shared memory accesses. Memory-resident locks are one option, but can have hard-to-bound costs. Atomic operations, e.g. add and increment, require hardware support, but offer a bounded cost. Ordered threads use signals to force a round-robin ordering between the threads of a single MR, but require reserving an entire ME. Another option is out-of-order execution, where a reorder buffer ensures that packets exit in order.
- **Bandwidth Regulation.** Bandwidth regulation ensures metarouters receive resource guarantees and fair use of excess resources. One option, static analysis, looks at all possible code paths, and provisions metarouter speeds based on worst-case resource costs. Static assumptions can be overly pessimistic; techniques that track actual use can provide better performance over a range of packet distributions and taken code paths [6]. Dynamic techniques can be distributed, where each metarouter yields when it exceeds resource requirements, or centralized, where a single entity controls the dispatch of packet-processing work.

The programming model is the interface through which software interacts with a system; it refers to language, libraries, architectural exposure, and supported hardware features. We detail more design options here:

- **Parallelism.** The pipelined model connects the cores in an NP in a chain. It can use fast direct connections to pass data between adjacent stages and use less code space, but scalability is reduced and adaptation is difficult. The alternative model, where a pool of MEs is available for processing, permits more sophisticated scheduling and greater scalability.
- **Modularity.** The programming model can implicitly find parallelism in an application [17], or require the programmer to explicitly decompose the application into modules [18].
- **Architectural Exposure.** The framework can hide architectural details for greater simplicity, or expose them for greater speed and flexibility.
- **Portability.** Portability can be desirable across NPs, between GPs and NPs, and between platform hardware revisions, but standardization can reduce opportunities for optimization.

The ideal programming model enables an experienced MR developer to program sophisticated tasks, while a new MR developer can quickly and simply program useful tasks.

5. FRAMEWORK DESIGNS

To make the discussion concrete, we present four framework designs that represent distinct points in the design spectrum. The best framework design will be fast, general, open, scalable, and provide isolation.

5.1 Static Code

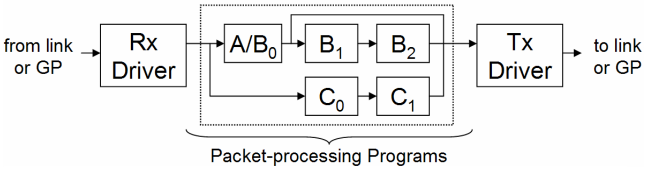

Figure 3. Static Code

In the Static Code design, shown in Figure 3, all NP functionality comes from a set of programs for handling common fast-path processing. Diversified Router developers create these programs to serve expected user needs, and bear the responsibility for mapping, debugging, and speed-testing them. Metarouter developers choose a program to handle their traffic and add custom packet-handling code to the GPs. Programs are organized as a pipeline of fixed driver stages and program-dependent stages. Within each processing stage, ordered thread execution ensures end-to-end ordering and safe critical section access. The design scales well; each added MR requires only a small section of SRAM for storing unique packet counters. Disadvantages include poor ME utilization, inability for researchers to load new NP code, and the need to remap and prove the performance for every added program. Despite those issues, the model does present an **immediately useful** demonstration of the diversified router concept. A PlanetLab developer only needs to choose the NP program closest to their experiment and make minor modifications to their GP code. For little effort, they gain a drastically higher (estimated 10-50x) packet processing rate. This design is based on the GENI Cooked Mode described in [1]; see the reference for other design options. We fully intend to deliver the Static Code design for an initial demonstration of the diversified router concept, as it is fast, provides isolation, and scales to large numbers of MRs.

5.2 Periodic Remapping

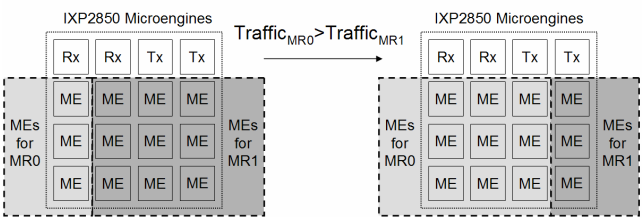

Figure 4. Periodic Remapping

This design, shown in Figure 4, is a summary of the runtime component of the Shangri-La project from UT Austin [11][12]. The distinctive feature is a scheduler component that periodically changes the processor mapping based on traffic conditions. The system is programmed in Baker, a language that represents packet-processing functions as a dequeue, process, and enqueue block. Dynamic adaptation enables higher aggregate throughput, better power consumption, and automatically adjusts to changing packet distributions. Unfortunately, the **fixed mapping limits scalability** to the number of processing cores, the adaptation time limits support for highly variable packet distributions, and it is unclear if and how the framework could provide guarantees to metarouters. These issues prevent the design from being a viable long-term Diversified Router framework, but we do expect its programming model and dynamic code loading work to be useful.

5.3 Time-slicing

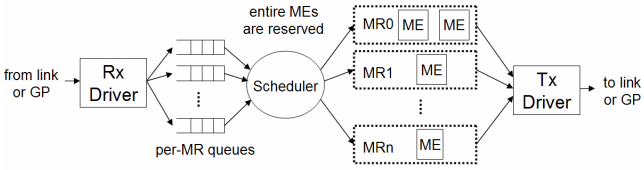

Figure 5: Time-slicing

Figure 5 shows the third framework design, which treats the MEs as a symmetric multi-processor system with an ME-resident real-time scheduler. The job of the scheduler is to decide which MRs should be provided resources and handle the dispatch of work to MEs. The primary resource to be scheduled is processor cycles, which can be as simple as deficit round-robin scheduling with a worst-case cycle cost assumption, or a more sophisticated algorithm specific to NPs that leverages the predictability of packet processing times [6]. Each MR has its own input queue; the scheduler can reserve MEs for a fixed time period, or for a time based on the number of packets in its input queue. Buffering helps the design cope with short-term variation in packet distribution and processing requirements, and ordered thread execution again provides simple resource ordering. The disadvantage is that each MR can no longer claim permanent access to ME resources, and its entire persistent context and code space must be saved and restored every time the MR is scheduled. This fact creates a trade-off between the number of MRs to be supported and sustainable performance levels. However, the central scheduler enables proper guarantees, and the lack of a

fixed mapping of processing to resources provides **exceptional scalability**.

5.4 Fine-grained Scheduling

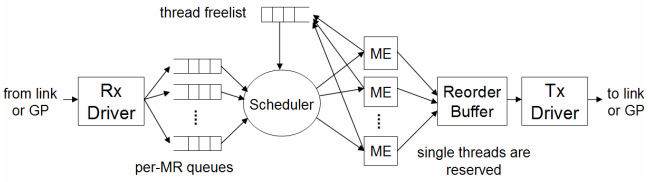

Figure 6: Fine-grained Scheduling

The last framework design, shown in Figure 6, enables concurrent processing of packets associated with different MRs. The MEs reserved for processing are treated as a set of independent threads, to which the scheduler can assign work. For simplicity, we assume that the code for every MR is on every ME. When the scheduler has selected an MR packet, it pulls a thread off a *freelist*, or queue of idle threads, to process that packet. When the thread has finished, it sends the packet to a reorder buffer and enqueues itself back onto the freelist. The reorder buffer is necessary to ensure end-to-end-ordering, as the load on each microengine varies. Shared data between threads of one MR must be safely accessed with locks or atomic operations, since sharing individual MEs eliminates the possibility of using ordered thread execution. The design more gracefully handles a mix of MR processing requirements, by providing much greater latency tolerance, minimal runtime overhead, and supporting packet reordering. The design does, however, add potential bottlenecks, such as the reorder buffer, additional accesses to shared memory, and lock acquisition times. Care must be taken to keep MR code sizes reasonably small, or the scheduler must take into account where the code for an MR is located. The primary advantage of this design is **maximum utilization.**

6. SUMMARY

This paper has demonstrated the importance of an NP runtime system to a flexible, cost-effective diversified router. We desire improved scalability, isolation, and performance over current NP runtime systems, and will deliver an open system to meet these goals. We have presented a variety of representative runtime frameworks for a diversified router, focusing on their differences from existing systems, and have shown that a combination of static verification, dynamic checks, scheduling and programming model techniques will likely meet all of our design goals. Designing with PlanetLab compatibility in mind leads to a system with immediate usefulness for a large research community. Our framework, designed for the multicore NP architectures of today, may be applicable to future multicore general-purpose processors. Finally, we assert that sharing an NP is an interesting research topic, one that fuses runtime systems, language research, and hardware design, and forms a useful component for a future network research infrastructure.

7. ACKNOWLEDGMENTS

Our thanks to Ben Wun for editing assistance.

8. REFERENCES

[1] Turner, J. *A Proposed Architecture for the GENI Backbone Platform*. GENI Design Document 06-09, http://www.geni.net/documents.php, March 2006.

[2] Kuhns, F., Wilson, M., and Turner, J. Diversifying the network edge. In *Proc. of INFOCOM 2005*.

[3] Bavier, A. et al. Operating System Support for Planetary-Scale Network Services. In *Proc. of Networked Systems Design and Implementation*, March 2004.

[4] Wolf, T., Weng, N., Tai, C. Design Considerations for Network Processor Operating Systems. In *Proc. of ANCS* 05. Oct., 2005.

[5] Wun, B., Turner, J., and Crowley, P., *Virtualizing Network Processors.* Technical Report WUCSE2006-12, Washington University, St. Louis, MO, 2006.

[6] Wolf, T., Pappu, P., and Franklin, M., Predictive Scheduling of Network Processors. In *Proc. of Comp. Networks,* 2003.

[7] Srinivasan, A., et al. Multiprocessor Scheduling in Processor-based Router Platforms: Issues and Ideas, In *Proc. of the 2nd Workshop on Network Processors*, 2002.

[8] Kencl, L. and Le Boudec, J.-Y., Adaptive load sharing for network processors, In *Proc. of INFOCOM 2002.*

[9] Kokku, R., et al. 2004. A Case for Run-time Adaptation in Packet Processing Systems. *SIGCOMM Computer Commuications Review.* 34, 1 (Jan. 2004), 107-112.

[10] Ruf, L., Keller, R., and Plattner, B., A. Scalable High-performance Router Platform Supporting Dynamic Service Extensibility On Network and Host Processors, In *Proc. of IEEE/ACS Intl. Conference on Pervasive Services (ICPS 2004).* July 2004.

[11] Kokku, R. *ShaRE: Run-time System for High-performance Virtualized Routers*. Ph.D. Thesis, University of Texas, Austin, Texas, 2005.

[12] Chen, M. K., et al. Shangri-La. In *Proc. of the 2005 ACM Conference on Programming Language Design and Implementation* (*SIGPLAN 05)*. Chicago, IL, USA, Jun 2005.

[13] Degioanni, L, et al. Network Virtual Machine (NetVM). In *Proc. of the 8th In..l Conf. on Telecommunciations (ConTEL).* Zagreb, Croatia, June 2005.

[14] Kumar, S., et al., S. C-CORE: Using Communication Cores for High Performance Network Services. In *Proc. of the Fourth IEEE Intl. Symposium on Network Computing and Applications*. Washington, DC, July 2005.

[15] Memek, G., and Mangione-Smith, W.H. NEPAL: a Framework for Efficiently Structuring Applications for Network Processors. In *Proc. of the 2nd Workshop on Network Processors*, 2002.

[16] Campbell, A.T., et al. NetBind: a Binding Tool for Constructing Data Paths in Network Processor-based Routers. In *Proc. of IEEE OPENARCH' 02*, New York City, NY, June 2002.

[17] Li, L., Huang, B., Dai, J., and Harrison, L. Automatic Multithreading and Multiprocessing of C Programs for IXP. In *Proc. of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP '05).* Chicago, IL, USA, June 2005.

[18] Shah, N., et al. NP-Click: A Productive Software Development Approach for Network Processors, *IEEE Micro*, 24, 5, (Sept.-Oct 2004), 45-54.