Deliverable FI3-D1.5.3
High-performance SW directed packet routing

Matias Eli, Sami Ruponen, Markku Savela

VTT

Tivit Future Internet Program
(Tivit FI)

Periodi: 1.4.2011 – 30.4.2012

Tivit, Strategisen huippuosaamisen keskittymän tutkimusohjelma

Rahoituspäätös 1171/10, 30.12.2010, Dnro 2790/31/2010

www.futureinternet.fi

www.tivit.fi

This work was supported by TEKES as part of the Future Internet programme of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT).

Executive summary / Internal release

Title: High-performance SW directed packet routing

Software routing promises to offer more adaptive and easily programmable network nodes. Until now, however, it has been considered too inefficient for production environments in terms of both performance and cost. Several recent studies have shown that software routers are now capable of scaling up to the high-performance routing tasks even with commodity personal computer (PC) hardware. In this deliverable, we have explored the possibility of running a software router on a network processing unit (NPU). The focus was to study if this approach provides any advantage over traditional off-the-shelf PC hardware with a comparable price point. The work was conducted by modifying Click Modular Router software to better utilize Cavium OCTEON network processor hardware.

Content: This deliverable explores the possibility of running a software router on a network processing unit (NPU). The deliverable introduces a modified hardware accelerated version of Click Modular Router. The performance of the system is measured using simple raw packet forwarding tests.

Impact: With minor modifications, we were able to significantly improve the performance of Click software router on an OCTEON network processor. This was achieved by modifying user space Click to take advantage of the available OCTEON hardware accelerator units. Stability and throughput test results have proved to be promising and the maximum reached throughput is comparable to current-day high-end commodity PC hardware.

Contact info: Matias Elo, matias.elo@vtt.fi; Sami Ruponen, sami.ruponen@vtt.fi; Markku Savela, markku.savela@vtt.fi

1 Introduction

Traditionally, software routing has been considered too slow and inefficient for high-performance routing tasks. During the last few years, however, software routers have started to emerge as a viable alternative to hardware routers in certain routing tasks. This has been made possible by the constant increase of processing power in commodity PC hardware and the availability of efficient open-source operating systems and routing software.

Certain network processors enable the same possibility to run custom routing software as normal PC hardware. They can also provide hardware accelerated units, such as packet management accelerators and crypto engines for network packet processing related tasks. The overall architecture of network processors is optimized for packet processing making their performance comparable to specialized hardware implementations but with the added flexibility that comes with software. In addition, their performance per energy consumption ratio is often more efficient compared to traditional PC hardware.

This deliverable explores the possibility of running a Linux software routing application on one type of NPU. The work is carried out by modifying a popular open source routing application Click Modular Router (Click) [1] to take advantage of Cavium OCTEON NPU [2] features directly, bypassing the kernel, network stack and device drivers. With the modifications, very high raw packet forwarding performance was achieved compared to a standard vanilla Click running on the same hardware. The implementation also supports multithreading on OCTEON processors, which allows us to reach better performance with its multicore architecture. In our throughput tests, the implementation has reached forwarding rates comparable to high-end PCs.

2 System Components

Click Modular Router

Our router implementation is built on Click, which is a modular open-source software router platform. Click routers are built from individual components, called elements, each implementing different packet processing tasks. Click can be easily extended to support new functionality by creating new elements, which are in practice C++ classes. There is also a rather large library of ready-made elements available making it possible to quickly deploy even complex systems.

Routers are built using Click configuration files, where the elements are connected together in a directed graph, which the packets then traverse. The elements are connected to each other using either push or pull connection method. In push connection the source element passes packets downstream to the destination elements whereas in pull connection the packet transfer is initiated by the destination element. Direct connections between push and pull elements are illegal and queue elements can be used to make the change while also storing packets. Fig. 1 presents a very simple packet forwarding configuration, which receives packets from one interface and then sends them out from another. A queue element is required, because FromDevice uses push connection and ToDevice pull connection.

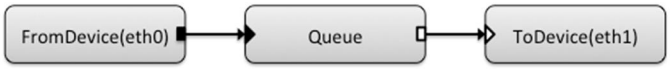

Figure 1. A simple Click packet forwaring configuration.

Click can be run as a normal user space application or as a Linux kernel module. The corresponding Click configurations are presented in Fig. 2 and Fig. 3. Traditionally user space Click has been used as a more nimble development platform whereas kernel-level Click is used when more performance is required. In user space Click, the major bottleneck is in the process of stealing packets from Linux kernel and the data copying between kernel and user space applications.

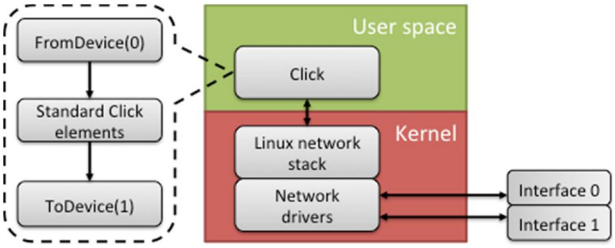

Figure 2. Standard user space Click.

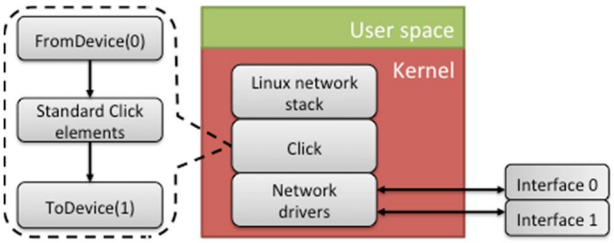

Figure 3. Standard Click kernel module.

Click also supports multithreading and the required synchronization functions, which are critical when concerning the multicore architecture of OCTEON NPUs. The load between cores can be adjusted dynamically during runtime or statically in the Click configuration file. [3]

Cavium OCTEON NPU

OCTEON products from Cavium Networks are processors designed to be used in routers, gateways and other network equipment that require high performance packet processing with specialized features. However, they also offer interesting opportunities for applications outside their traditional usage.

OCTEON network processors consist of processor cores, a memory controller and additional hardware accelerator units that are integrated into one chip. The purpose of the hardware units is to offload the cores from specialized, computing intensive tasks and to reduce the software complexity and overhead. Although the clock frequency of the processor is relatively low and thus the performance of a single core is limited, it is able to achieve high performance by its multicore design. The processor chips also include different types of I/O interfaces for network and peripheral connections. The cores are based on MIPS64 instruction set with some additional Cavium-specific instructions.

While the processor itself may seem complex, the software development process for utilizing all the features is made relatively easy. A standard GNU C/C++ compiler, supporting development tools and Cavium's C-language APIs, called Simple Executive (SE), to the hardware units are made available; hence with a moderate effort one can make special software taking advantage of the available hardware acceleration.

What makes this processor interesting for certain applications, like our work, is its ability to run Linux kernel and applications. Basically, one can take e.g. a stock Debian GNU/Linux with customized kernel and use it as any other Linux running on some common PC hardware.

The accelerator units can be categorized based on the type of acceleration they provide for the overall packet processing: packet-management, security and application accelerator units. Different processor models include different sets of accelerator units, but every model has the packet-management accelerator units. In general, these units take part in processing every packet received or transmitted by the processor. The units execute their functions without assistance from software running on the cores, though software can customize the operation by accessing certain configuration registers.

A generalized view of the OCTEON processors is given in Fig. 4 showing the packet-management accelerators: Packet Input (PKI), Packet Output (PKO), Free Pool Allocator (FPA) and Schedule/Synchronization/Order (SSO) units and also the cores, memory and their interconnections.

PKI handles every incoming packet and is responsible for verifying checksums, classifying the flow, obtaining buffers for the packet data and copying the data into the memory. It also determines the Quality of Service (QoS) class for internal processing of the packet. In a similar fashion, PKO is responsible for retrieving the packet data from the memory and calculating the checksums for transmitted packets. It also handles QoS and output queuing of the packets and freeing of packet buffers.

FPA is responsible for managing different buffer pools including the packet data buffers. It keeps records of free buffers and assigns one on request.

From the cores' perspective, every packet is seen as a work for them. When a core becomes ready, it can request more work from the SSO, which schedules them the highest priority work from the work queue. SSO is responsible for synchronizing the packet processing especially when packets are processed in parallel giving each core an exclusive access to certain data if required. It also maintains the ingress order of packets for ordered transmissions in co-operation with the PKI and PKO.

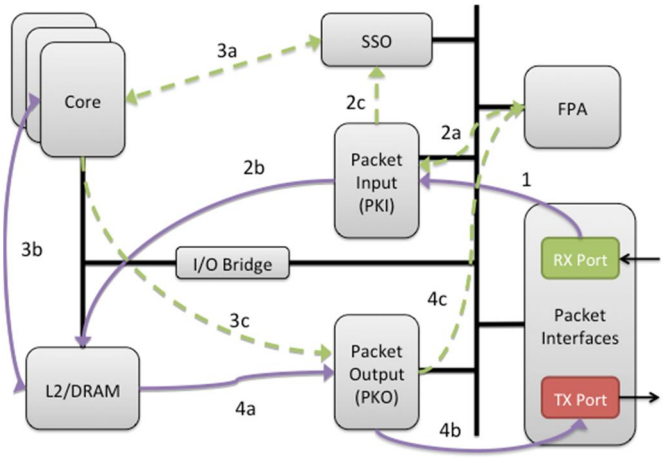

Figure 4. Simplified OCTEON packet flow [4].

Fig. 4 also shows a typical operation of different hardware units for an incoming Ethernet packet. Detailed description of the units' operation is omitted. The solid (purple) lines depict packet data being copied or accessed and the dashed (green) lines where only instructions and addresses are shifted.

First the packet is copied into PKI's internal buffer (1). PKI then requests a buffer for the packet data and for the work entry from FPA (2a), copies the data into the assigned buffer in the memory (2b) and sends a work entry to the SSO including relevant information of the received packet (2c).

After a core becomes idle, it can ask SSO for more work (3a) and if one is available it gets the detailed packet information with memory locations etc. Software running in the core can now start processing the packet as programmed (3b). When finished the core can e.g. drop the packet freeing the resources or send an instruction to PKO for scheduling the packet for output (3c). After this, the core software does not attend to the output process.

PKO receives the instruction that has information about the packet: the address where the packet data resides, the length of the packet and from which port it should be sent out. In the order of the output queue, PKO retrieves the packet from the memory to its internal buffer (4a), calculates necessary checksums and sends the packet out from the proper port (4b). PKO can then free the buffer by giving it back to FPA (4c).

3 System Implementation

Our system is based on the latest user space Click version (2.0.1). We are using Linux kernel version 2.6.32.27 that is included in the proprietary Cavium OCTEON SDK version 2.2. There is also an open-source version of the SDK available for free use [5]. The modified Click was tested on two different category OCTEON devices: a low-end OCTEON model CN5020 that includes three 1 Gbps Ethernet interfaces and two processor cores running at 500 MHz, and a higher-end OCTEON model CN5750 with two 10 Gbps Ethernet interfaces and 12 processor cores running at 750 MHz. We created a series of patches, which add our modifications to the standard Click code path. After this, Click is cross-compiled to run on OCTEON processors using the tools from the SDK.

The basic architecture of our system is presented in Fig. 5. The user space approach was selected, because it offered an easier way to fit OCTEON's buffer management to the simple user space Click Packet structure. Kernel-level Click uses a Packet structure based on sk_buff, which is used everywhere within the Linux kernel and has very strict usage rules. The target of the development was also to minimize all interactions with the kernel.

Simple Executive calls enable to use the hardware directly from user space bypassing the kernel and the network stack, so all packet processing happens in user space. This removes the need for transferring data between kernel and user space, thus allowing Click to perform comparably to a kernel-level implementation. Operating in user space is also more secure and has fewer opportunities to compromise the system stability. However, running applications that use SE calls is not recommended in a multi-user system, because through SE calls the application has more or less unlimited access to the OCTEON hardware and memory.

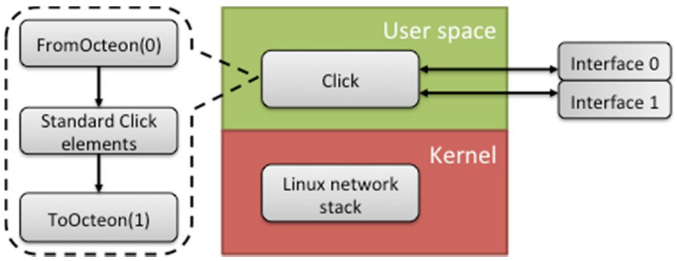

Figure 5. User space Click running on OCTEON hardware.

Our implementation presents two new Click elements FromOcteon and ToOcteon that replace standard Click FromDevice and ToDevice elements. These two elements communicate directly with the OCTEON hardware using SE calls. Our elements should be fully compatible with all current user space Click elements, which makes porting of current Click configurations to OCTEON hardware quite simple.

FromOcteon element uses polling to fetch new packets from the OCTEON hardware. In practice, it requests new work from the SSO. After it has received new work, it creates a new Click packet, which includes a pointer to the actual packet data stored in the OCTEON buffer. This Click packet is then passed downstream to the next element in the configuration. ToOcteon element sends the packet to the PKO unit, which takes care of the output queuing. Due to this, ToOcteon can use push connection method and requires no separate queue element.

Additionally, minor modifications to Click were needed to better support OCTEON hardware. Code for initializing and releasing OCTEON hardware resources was added to Click, and small changes were also required to Click's packet class to enable it to use OCTEON packet pools when creating new packets. Standard Click provides atomic operations only for x86 architecture, so we added similar operations for MIPS architecture from the OCTEON SDK.

OCTEON hardware can take care of the packet order, which simplifies the creation of multithreaded Click configurations. However, this has a negative effect on the maximum performance due to the required locking in the output unit. Due to this, we have made locking an optional parameter in the ToOcteon element.

4 Experimental Performance Evaluation

The early performance measurements on the hardware accelerated Click were performed with a minimalistic setup. The test setup consisted of two OCTEON NPUs connected together using

full-duplex 10 GBASE-SR transceivers and fibers as presented in Fig. 6. One OCTEON device is running an example traffic generator application from the OCTEON SDK. The traffic generator is running as a stand-alone application without any Linux operating system. Although the accuracy of the traffic generator has not been well tested, it servers the purpose for early performance tests.

The traffic generator is capable of reaching line speed at 10 Gbps even with minimum sized Ethernet frames. The generator sends frames from one interface and listens for incoming frames from another interface. A second device is running Linux with a user space Click having a configuration that simply forwards frames from one interface to another. No queue element is required, because the input of ToOcteon element operates in push mode. Using this configuration, we can detect from the traffic generator when the system running Click starts to drop frames. This setup provides sufficient accuracy for the initial performance tests.

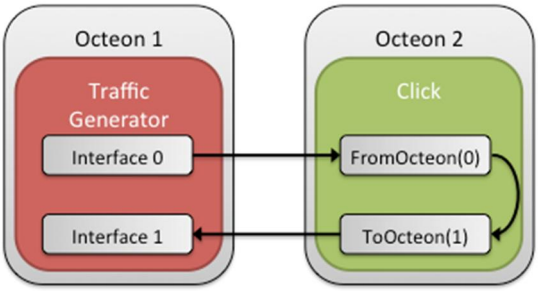

Figure 6. Raw throughput test setup.

Using the above mentioned test setup we measured the maximum loss-free forwarding rate (MLFFR) using different number of OCTEON cores and different frame sizes. The first measurements were performed with minimum sized Ethernet frames, and the related results are presented in the Fig. 7. The 64-byte frames that were used included Ethernet, IP and UDP headers, minimum payload and Ethernet cyclic redundancy check (CRC) value. This test illustrates the worst-case scenario when the NPU has the least time to perform its operations. The measurements were repeated with varying number of cores dedicated for running Click to see how the performance scales. The theoretical line speed of a 10 Gbps Ethernet link when using 7-byte preamble, 1-byte Start Frame Delimiter, 64-byte Ethernet payload and 12-byte inter frame gap is approximately 14.88 million frames per second $\left(i.e.\frac{10\,000\,000\,000\ bps}{(8\,[pre]+64\,[payload]+12\,[ifg])*8\ bits}\right)$.

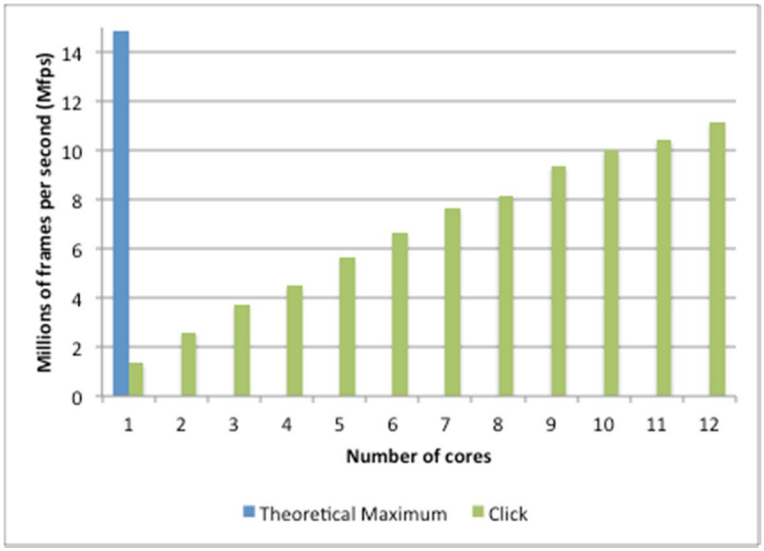

Figure 7. Raw frame forwarding rate with minimum sized (64 Bytes) Ethernet frames.

As can be seen from the figure above, the achieved performance is still far away from the theoretical maximum. However, the results are comparable to the numbers achieved in another study on high-end PC hardware [6]. The overhead from using multiple concurrent threads can be seen in Fig. 7 as the frame rate does not increase linearly when the core count is increased. A similar test was conducted with the low-end device, which was able to reach 1 Gbps line speed with minimum sized frames using two cores.

In the second measurement, the frame size was increased until the system reached line speed processing. The test was again repeated with varying number of cores in use. The test helps to understand the effect of the frame size to the system performance. The results of this test are presented in Fig. 8.

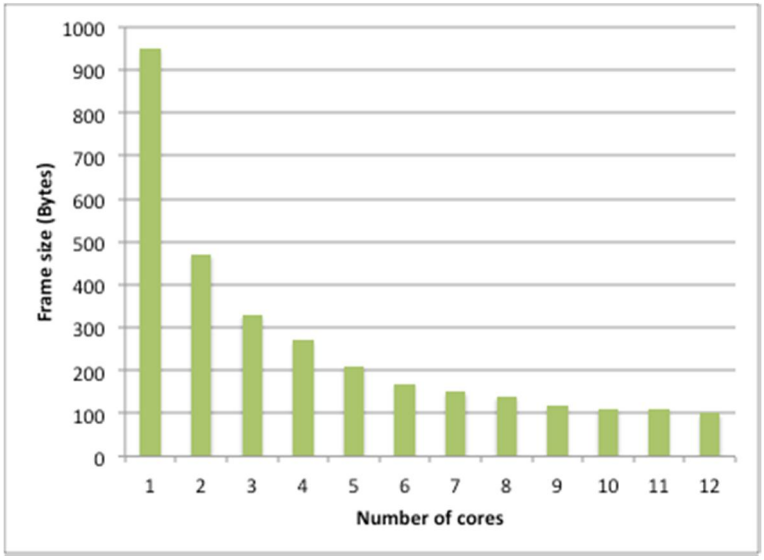

Figure 8. The minimum frame size to reach line speed.

The figure shows, that the effect from adding more cores quickly decreases and already with seven cores the performance increase is minimal. This means that adding more cores will not

improve the forwarding performance any further leaving the other cores free for other tasks. The bottleneck causing this effect requires further studies.

Fig. 9 visualizes the effect of frame size to the maximum forwarding rate. It shows that with our current implementation, Click is able to achieve the 10 Gbps theoretical line speed with a frame size of 128 bytes. With 256-byte frames the device can reach line speed with only 6 cores. This shows potential for the router implementation, as data frames in real-world networks are usually considerably larger than 64-bytes.

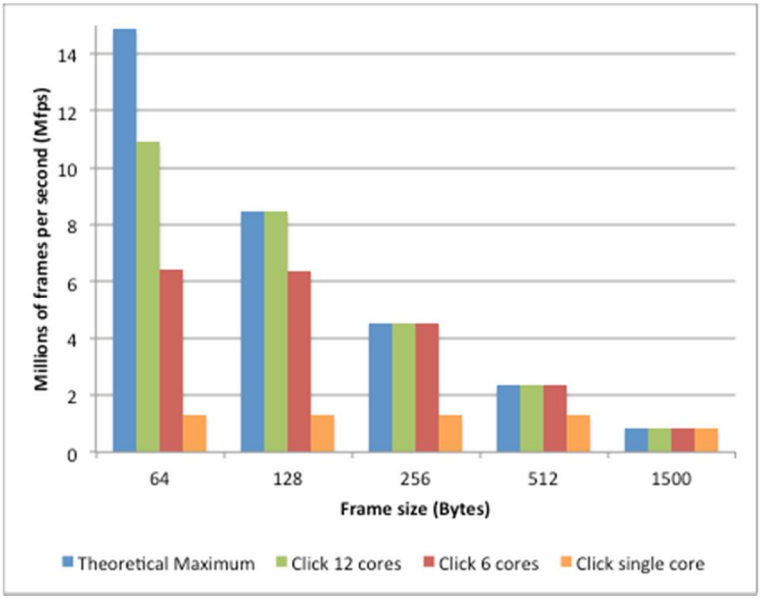

Figure 9. Raw frame throughput with different frame sizes.

To better understand the performance requirements, the available instruction count per incoming packet at 10 Gbps speed can be estimated. Considering the worst-case scenario, that is using minimum sized Ethernet frames, the corresponding packet rate is 14.88 million packets per second. With a core frequency of 750 MHz there are roughly 50 clock cycles available to complete packet processing. By further estimating that one instruction finishes per clock cycle, 50 instructions can be consumed on processing one packet. This means that there are hardly any spare cycles to spend. Even by multiplying this with 12 the instruction count increases to only 600. Taking into account the internal overhead of different hardware units, the effective instructions available for actual packet processing are further reduced.

We also measured a rough power usage of the two devices in idle state and under load. The load power consumption was measured during the throughput test. These measurements are shown in Fig. 10. Both devices managed to keep their power usage at reasonable levels taking into account their packet processing performance. We argue that the power consumption is very competitive although we were not able to conduct comparative tests with similarly performing PC hardware.

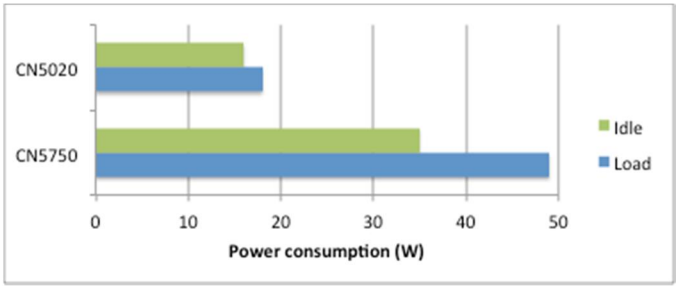

Figure 10. Measured power consumption.

5 Conclusions and Future Work

In this deliverable, we have explored the feasibility of running a software router on a network processor. With minor modifications, we were able to significantly improve the performance of Click software router on an OCTEON network processor. This was achieved by modifying user space Click to take advantage of the available OCTEON hardware accelerator units. Stability and throughput test results proved to be promising though some bottlenecks still affect the overall performance. The implementation is still a work in progress, and the performance can probably be improved with further development. However, with the given stringent timescales the code optimization may require considerable effort.

To better understand the system performance more accurate measurements are required. Deeper analysis is required for identifying the remaining system bottlenecks. More OCTEON features could be included into the Click framework, such as: checksum computation, cryptographic algorithm and hash calculations, and QoS classes. These work items will be tackled in the near future.

6 References

[1] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. "The Click Modular Router," in *ACM Transactions on Computer Systems, vol.* 18, no. 3, pp. 263-297, 2000.

[2] Cavium OCTEON. http://www.cavium.com/OCTEON_MIPS64.html.

[3] B. Chen and R. Morris, “Flexible Control of Parallelism in a Mul- tiprocessor PC Router,” in *Proceedings of the USENIX Annual Technical Conference*, 2001.

[4] Cavium Networks, “OCTEON Programmers Guide – The Fundamentals,” 2010. Available: http://university.cavium.com/ downloads/Mini_version_of_Prog_Guide_EDU_July_2010.pdf.

[5] CNUSERS. http://www.cnusers.org.

[6] N. Varis and J. Manner, “Performance of a Software Switch,” in IEEE HPSR, 2011.

Appendix: Software installation

Prerequisites

- Click Modular Router, http://www.read.cs.ucla.edu/click/click (tested on commit 7104dd4913d9f74d13cd82eb93890718369b1336)
- Click OCTEON patches
- Cavium OCTEON SDK 2.2, http://cnusers.org/ (Open Source version available)

Installation

1. After installing and setting up the OCTEON SDK download the most recent Click source code from the GIT repository

```
git clone git://read.cs.ucla.edu/git/click DIR
```

2. Move to the Click folder and apply our patches

```
cd click
patch -p1 < PATCH
```

3. Configure user space Click for OCTEON devices

```
autoconf
./configure --disable-linuxmodule --enable-userlevel \
--host=mips64-octeon-linux-gnu --build=i684-linux \
--enable-tools=mixed --enable-octeon --enable-user-multithread \
--enable-ipsec
```

4. Build user-level Click

```
cd userlevel
make
```