The Design and Implementation of a PCIe-based LESS Label Switch

Amy Williams

A Senior Thesis submitted in partial fulfillment
of the requirements for graduation
in the Honors Program
Liberty University
Spring 2017

Acceptance of Senior Honors Thesis

This Senior Honors Thesis is accepted in partial
fulfillment of the requirements for graduation from the
Honors Program of Liberty University.

_____
Feng Wang, Ph.D.
Thesis Chair

_____
Robert F. Melendy, Ph.D.
Committee Member

_____
Evangelos Skoumbourdis, Ed.D.
Committee Member

_____
David E. Schweitzer, Ph.D.
Assistant Honors Director

_____
Date

Abstract

With the explosion of the Internet of Things, the number of smart, embedded devices has grown exponentially in the last decade, with growth projected at a commiserate rate. These devices create strain on the existing infrastructure of the Internet, creating challenges with scalability of routing tables and reliability of packet delivery. Increasing use for real time data collection, computation, and control generate a high demand for a reliable, high performance, scalable network and architecture. In previous work, Location basEd Source Switching is proposed to provide a novel method of labeling and routing packets to ensure rapid, reliable delivery. This thesis seeks to design, implement a PCIe-based LESS label switch to process unrouteable packets under the current LESS forwarding engine.

The concept of the Internet of Things – large heterogeneous networks of devices, sensors, controllers, and more – has emerged as a prevalent new sector of technological development. The vision to develop a wirelessly connected array of a wide variety of sensors, smart devices, and other objects that sense and respond to real world information and situations necessitates new architectures and methods of networking in order to quickly and reliably handle large amounts of real time information. Thus, the concept of structure and design of Future Internet has become a focus of research and discussion in order to accommodate and facilitate these technological advances. To facilitate the simultaneous use, connection, and communication of a large number of devices, the Internet of Things requires high reliability and low end-to-end latency of packet delivery on the network. Various schemes for the architecture have been proposed for the Future Internet; however, they tend to exacerbate one problem to solve another This thesis investigates the theory, design and implementation of an FPGA-based LESS forwarding mechanism for Internet of Things (IoT) in order to meet the scalability, high performance, and low latency requirements.

Motivation and Problem Statement

Since its creation, the Internet has experienced exponential growth, with a wide variety of uses beyond what was initially conceptualized at its creation, going beyond the connecting of computing and sharing of information to connected and controlled devices. Currently, the Internet is mostly seen to facilitate human to human interaction, or human to machine interaction. The Internet of Things, however, places a primary emphasis on machine to machine connections (Khan, Khan, Zaheer, & Khan, 2012). In 2003, there were an estimated 500 million connected devices, compared to the world's 6.3 billion

people, while in 2015 that had grown to 25 billion devices for a world population of 7.2 billion, which means that during the dot com boom, the number of computing devices went from 7.93 devices per one hundred people to 342 devices per hundred. The latter half of this growth results from the beginning precursors to the Internet of Things, and the growth rate is projected to keep developing at an exponential rate, reaching 50 billion devices in 2020 (Evans, 2011). With an estimated population of 7.63 billion people in 2020, that makes for 655 smart, connected devices per one hundred people in the population. Despite advances in network technology and communication protocols, the rate of increase for devices far outpaces the current Internet's ability to provide services for all of them.

While the current Internet exerts its best effort to deliver packets, it does not guarantee bounds on delay or delivery. In the expansion of the current Internet to incorporate IoT, issues of reliability, performance, and scalability must be addressed in order to meet the ever growing and developing technological demands. With the rapid proliferation of these devices, issues of scalability of the network become paramount. Current implementations require expensive look up tables to facilitate routing. Even much of the current research, such as Software Defined Networks (SDN), improves latency at the expense of scalability. These solutions generally feature some sort of centralized controller, which requires forwarding tables that increase linearly with network size. The novel method of addressing and forwarding of data packets in LESS forwarding seeks to provide a readily scalable architecture for Internet of Things by eliminating the need for forwarding tables.

Performance and reliability also pose significant challenges for specific areas of the Internet of Things. Various applications in the "Tactile Internet," such as automated vehicles, require the reaction time of the vehicle to be on the order of milliseconds. Such mission critical speeds necessitate incredibly fast forwarding speeds, in order to allow the packet to be transmitted and the data processed under the time constraints. Furthermore, in the event of a node or link failure, packets must nonetheless be able to reach the intended destination within the specified amount of time.

As the demands on networks grow, both in terms of number of devices connected to the network and the amount of data, and therefore packets, sent across the network, providing a reliable and scalable high speed network becomes imperative to ensuring continued growth and development. To address these challenges, a Location basEd Source Switching (LESS) with rerouting capabilities is designed, implemented, and evaluated to ensure reliable throughput of the system, including forwarding and processing packets.

Objectives

This thesis describes the background, theory, and design process of a new PCIe-enabled LESS system, as well as a strategy for testing and future study when implementation is complete. As the name indicates, LESS forwards packets based on source routes encoded into the packet headers. Source routing is discussed further in the Background section. Since each hop required for the packet to reach the destination is encoded in the header of the packet at the source, a LESS switch does not need routing or lookup tables, nor do they need to be able to translate labels. Since the addressing scheme is location based and hierarchical, it can maintain a low overhead with regards to network

topology. A hardware and software based implementation of a LESS switch was designed for packet processing latency, throughput, and packet re-routing latency in order to demonstrate the reliability and performance of a LESS switch.

Outline

The rest of this thesis is organized as follows: the next section covers background information on the Internet of Things, source routing, and the PCI Express, while the section after that covers the LESS Architecture and Background. The third section details the specific design for LESS Switch with PCIe framework, and the fourth provides a strategy for performance evaluation of the LESS Switch implementation. The last section provides conclusions as well as direction for future research.

Background

The theory and technical background of the Internet of Things, source routing, and the PCI Express technologies provide foundational concepts to the development of the PCIe based LESS switch. As previously mentioned, the Internet of Things provides the context and motivation for this thesis while source routing provides the networking concepts that allow it function. The PCI Express serves as the technology to improve current design.

Internet of Things (IoT)

The Internet of Things, while a focus of much discussion and development, lacks a single cohesive definition, architecture, or standard. A definition of an architecture or protocol for Future Internet has yet to be an agreed upon, though various architectures have been proposed. The following section provides an overview of various architectures,

as well as the challenges that any IoT structure must address in order to meet the needs of the Future Internet.

Future Internet of Things Architectures. As it currently stands, there is no formal consensus or standard for the Internet of Things Architecture, though developing and proposing such models serves as the focus of many national and international organizations. Among these are the International Telecommunications Union (ITU), which is an agency of the United Nations, the European Research Cluster on the Internet of Things (styled either CERP-IoT or IERC), and the National Telecommunications and Information Administration (NTIA), part of the United States Department of Commerce. These organizations have formulated proposals for architectural and protocol standards, which teams in academia and industry research have then developed into other models.

The ITU provides one of the foundational architectures in its Recommendation Y.2060, of which there are now several variations. Their model divides into four layers, each of which have differing capabilities: the device layer, which is the lowest layer and manages the various devices' direct and indirect interaction with the communications network. The network layer communicates the data gathered and computed by the various "things" up to the rest of the network, as well transporting device management information down to the sensors. The next higher layer in the architecture, service support and application support layer, provides services common to IoT applications, such as a database for information and data storage, and device and application specific services. Given that the Internet of Things consists of a heterogeneous blend of devices, sensors, and services, this layer serves an important function in creating a comprehensive and unified structure. The highest level consists of all current and future IoT applications

and their functionality (International Telecommunications Union, 2012). Overall, these four layers form the conceptual basis for the proposed Internet of Things architecture.

Several variations of this recommendation add a fifth layer, albeit in different places in the hierarchy, in order to add additional functionality. Khan et al. (2012) proposes a "Business Layer" above the application layer for the purpose of system management and construction of business models. Tan and Wang (2010) cite a proposal by Anthony Furness to split the bottom two layers into three: an edge technology layer, an access gateway layer, and an internet layer. They then proceed to suggest their own modification, which adds a coordination layer between what they term the backbone network layer (network layer) and the service support and application support layer (which they term middleware). The coordination layer plans to address issues of interoperability between devices, services, and networks.

Future Internet Challenges. In developing an architecture for the Internet of Things, key challenges must be addressed in order to make it a feasible solution for implementation. Among these are object naming and identity management, which encompasses a network's ability to create and assign unique identities for large numbers of devices connected to the Internet and safety and security of those objects since actions taken based on the data provided by objects could be manipulated by physically compromising any of the sensors/devices/objects. Beyond object security, an architecture must provide for data privacy and security and encryption. Sensitive information, such as personal health data collected by sensors for medical purposes, shall be encrypted and protected across a network, but ensuring data integrity over such a vast quantity and range of devices and information provides a large challenge (Khan et al., 2012).

Interoperability constitutes perhaps the greatest challenge, beyond the issues of scalability, reliability, and performance mentioned previously. Since the Internet of Things means to incorporate a wide variety of devices with varying communication protocols, QoS requirements, and manufacturer determined specifications, ensuring interoperability becomes critical to the successful execution of IoT. Khan et al. (2012) proposes standardization for the Internet of Things, while Tan and Wang (2010) added the Coordination Layer to their architecture to ensure interoperability of devices on the network. Solving these challenges becomes paramount to enable the realization of a global IoT network.

Source routing

Source routing is a method of routing where the routing decisions are made in advance of the sending of the packet; that is, that all the routing information is provided by the source. The route the packet takes is comprised of a list of hops it makes from the source to the destination (Peterson, Davie, 2011). In order to utilize this type of routing, the source of the packet must be aware of the entire network topology in order to appropriately list the nodes that provide each hop along the route. Source routing exists in two general categories, based on the degree of route specificity included in the packet header: a strict source route, which details every single node the packet must traverse in order to reach its destination, and loose source routing, which gives only a partial set of nodes to cross. The benefit of the loose source routing is that it minimizes the amount of information that must be contained in the packet header. Furthermore, depending on the size of the network, obtaining the amount of information about the network topology in

order to be able to specify every single node can be arduous at best and impractical at worst, so loose packet routing is more practical for large networks.

Source routing was first proposed in 1977, when the Internet and number of connections was much smaller. It was proposed because it allowed for "the elimination of complex routing responsibilities from intermediate nodes" which greatly simplifies the requirements of routers. It further simplifies the challenge of network names, as a destination is composed not of a name or address, but merely result of the path to get there (Sunshine, 1977). Indeed, these concepts upon which source routing bases itself address some of the challenges that the Internet of Things must grapple with in order to be feasible.

Source routing as proposed above contains a few disadvantages and drawbacks; namely that it is static and unresponsive to changes in the network and performance considerations. When the route is specified by the source and intermediate nodes merely forward the packet according to the header, link or node failure presents a problem, as the packet is unable to take an alternate route, because the destination is not known to the intermediate nodes. Moreover, source routing can produce less than optimal routes, because the dynamic elements of the network, such as bandwidth, traffic, congestion, or delay, among others, cannot be fully predicted or analyzed at the source (Sunshine, 1977).

If the drawbacks associated with source routing were to be addressed in a sufficient manner, then the benefits to the Internet of Things that it provides could be fully utilized. The principle of a LESS system seeks to fix the challenges inherent to a

static network to make it more responsive to real time application and situations, in order to benefit from the simplicity of router requirements and path names.

PCIe Express

The PCIe Express, or Peripheral Component Interconnect Express (abbreviated as PCIe), is a high speed serial interface between two devices, specified for the bottom three layers of the OSI 7-layer model: the transaction layer, data link layer, and the physical layer. It is a full duplex, point-to-point physical connection that offers a variety of lanes, depending on the needs of the devices. The PCIe specification allows for x1, x2, x4, x8, x16, or x32 links between the devices. The number after each 'x' indicates the number of lanes, or pairs of signals, the device supports. Devices with a smaller number of lanes can be inserted into a larger PCI slot, and an algorithm at initialization auto-negotiates the greatest common number of lanes, and that number is used. This ensures the compatibility of any number of PCIe devices with a single physical slot (Budruck 2003). A visual representation of the connection is illustrated in Figure 1.

Furthermore, the architecture and protocol associated with the PCIe bus follows the seven layer OSI model that forms the basis of most communication systems and protocols. In the case of the PCI Express, the standard dictates the bottom three layers of the stack: the physical layer, the data link layer, and the transaction layer. Each layer has its own packet and function to ensure data communication. The transaction layer packet (TLP) contains the header, data, and an end-to-end CRC (ECRC) if desired. When it is passed to the data link layer, a sequence ID is prepended to the packet and an LCRC is appended to the packet in order to create the data link packet (DLP). The DLP is then passed to the physical layer, which is responsible for actually transmitting the entire

Image can be found at

https://www.mindshare.com/files/ebooks/PCI%20Express%20System%20Architecture.pdf

Figure 1: Representation of the interconnect between two PCIe devices. Reprinted from *PCI Express System Architecture* (p. 25), by R. Budruck, 2003, Boston, MA: Addison-Wesley Professional. Copyright 2004 by MindShare, Inc.

packet via a stream of 1s and 0s. The physical layer adds the required start byte to the head of the packet and the required end byte to terminate the packet, which creates the physical layer packet. This packet can then be transmitted via the PCI Express. A visual representation of the flow of the data through the layers, as well as the changes made to the packet, is shown in Figure 2.

PCIe design principles ensure that that it meets technical specification to suit it for current and future communication needs. It seeks to provide an interface for a variety of platform interconnections, be backward compatible with PCI, meet low-overhead requirements to maximize the bandwidth and link efficiency and support a multitude of more advanced specifications relating to power management, debugging implementations, and error handling, among others (Mishra, Singh, & Rousseau, 2015).

Throughput. High efficiency, packet based data transfer forms one of the defining features of the PCI Express. The throughput of devices' connections linked by PCIe depend on the protocol overhead, the size of the packet payload, the characteristics of the devices themselves and the latencies of the flow control mechanism and completion mechanism (Altera Corporation, 2015a). PCI Express specification for Gen2

Image can be found at

https://www.mindshare.com/files/ebooks/PCI%20Express%20System%20Architecture.pdf

Figure 2: Detailed block diagram of the layers specified by the PCI Express. Reprinted from *PCI Express System Architecture* (p. 25), by R. Budruck, 2003, Boston, MA: Addison-Wesley Professional. Copyright 2004 by MindShare, Inc.

utilizes a 8b/10b encoding scheme, meaning that for each byte of data, ten bits are encoded and transmitted, resulting in a 25% protocol overhead (Badruck, 2003). Since the transmission/reception rate specified for PCIe is 2.5 Gbits/sec per lane per direction, for a x4 link, the maximum bandwidth is (2.5 Gbits/sec per lane per direction) x (4 lanes) x (2 directions) / (10 bits / 8 bits) = 16 Gbits/sec or 2 Gbytes/sec (Badruck, 2003). Other factors may affect various actions taken by the PCI Express. In the IP core for the PCI Express for its FPGA chips, Altera noted that the theoretical maximum throughput could be dramatically affected by the payload size. As seen in Figure 3, a packet with a payload of only 16 bytes has an approximate maximum of 50% throughput no matter the header size, while the maximum throughput of a packet with a 4096 byte payload approached 100%, regardless of header size. Thus, throughput improves the greater the packet payload, which is ideal for situations where large amounts of data are sent via the PCIe bus.

Image can be found at

https://www.intel.com/content/www/us/en/programmable/documentation/nik1412473924913.html

*Figure 3.*Throughput maximum for PCIe based on packet payload size. Reprinted from "PCI express high performance reference design," Altera Corporation, 2015, San Jose, CA. Copyright 2015 by Altera Corporation.

LESS Background and Architecture

The LESS system architecture, proposed by Wang, Goa, Shao, Harai, and Fujikawa (2015), is comprised of four main components: the hierarchical addressing scheme, an LMAC directory system, LESS agents, and LESS-enabled switches. Figure 4 provides an overview of the architecture of a LESS enabled network. The scope of this paper focuses on the LESS-enabled switch design and implementation, but the other items are mentioned briefly to explain the necessary components of the LESS switch. The hierarchical addressing scheme provides Location-based pseudo MAC (LMAC) addresses to both switches and hosts according to a novel algorithm provided by Wang et. al (2015) with the LMAC directory system keeping track of all of the mappings between the device's IP address and the assigned LMAC address. Further details on the addressing scheme is provided in Section III of Wang, et al. (2015). The LESS agent runs on the hosts and accesses the LMAC directory system to swap the source and destination MAC addresses with LMAC addresses in order to derive the route to the destination. This maintains the simplicity, and thus high performance, of the LESS switch by enabling it to forward the received packets to the next hop according to the included, pre-derived

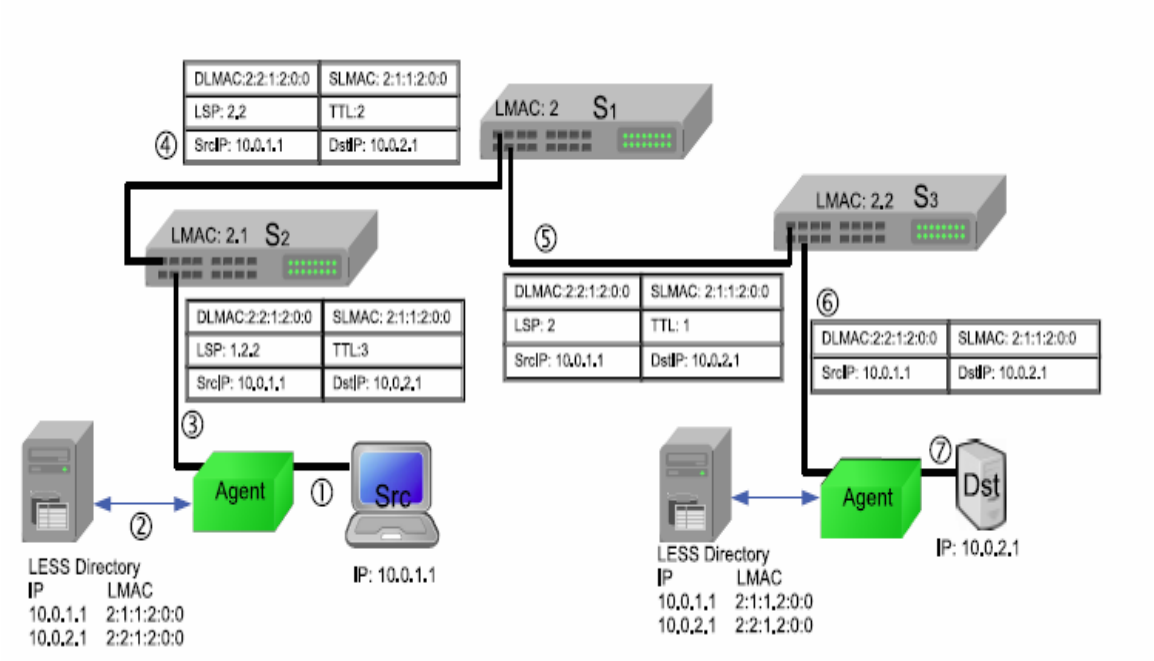

Figure 4. Example of LESS architecture with the four main components included and labeled. Reprinted from "Towards reliable and lightweight source switching for datacenter network" by F. Wang, L. Goa, X. Shao, H. Harai, & K. Fujikawa, 2017, Manuscript submitted for publication.

source route, while providing a method of packet re-routing in case of link failure in order to maintain reliability (Wang et al., 2015). Thus, the LESS enabled router forward packets without regard to the IP addresses and expensive IP lookup tables. In order to do so, however, the forwarding information must be included in the packet header.

LESS Switching

In order to support this novel method of source routing, a new type of source switching label, called a Label Switched Path (hereinafter LSP), is implemented to encode the source route of the packet. The LSP label is added to the packet in the shim header,

which resides between the data link layer and the IPv4 layer. The source switching label includes the sequence of nodes the packet must traverse in order to reach the destination. A Multi-Protocol Label Switching (MPLS) encapsulation format is leveraged to represent the LSP labels. As seen in Figure 5 below, an LSP label is comprised of 32 bits, which are allocated as follows: the first 24 bits encode 3 port numbers, with one byte for each port number. If more than 3 ports are needed, multiple LSP labels are embedded in the header. When a port number is read from these bits, the first 24 bits are then shifted to the left by one byte in order to remove that number and make the next hop the first byte in the LSP label. The next 1 bit (S bit) is the label stack bit, used to denote the last LSP label in the stack. If the S bit is set to 1, it indicates that the LSP label is the last and an IP packet header is stored after the LSP label. Otherwise, additional LSP labels follow. The last 7 bits of the LSP label are the Time to Live (TTL) field, which is decremented by 1 each hop, and when the field reaches 0, the LSP label is discarded. The LESS switch contains a switching fabric and forwarding engine that parse these labels in order to determine the next hop. Because the source route is an explicit series of steps, when a packet reaches a switch, the forwarding engine pops the first label off the packet, parses it to determine the output port through which to forward the packet, deletes that number from the label, then pushes the label pack on the packet. The process repeats until the packet reaches the destination, at which point the entire LSP has been deleted, leaving only a regular IP packet (Wang et al., 2017).

Image can be found at https://ieeexplore.ieee.org/document/8057152

Figure 5. LSP Switching label format of a LESS packet. Reprinted from "Towards reliable and lightweight source switching for datacenter network" by F. Wang, L. Goa, X. Shao, H. Harai, & K. Fujikawa, 2017, Manuscript submitted for publication.

LSP Switching Label Design and Construction. The LSP switching label for the packet to be forwarded by LESS switches is composed by an algorithm designed for this purpose based on the LMAC addresses of the source and destination hosts. In order to construct the LSP label, the LESS agent looks up the source and destination LMAC addresses in the LESS directory system and concatenates them in a specific way in order to specify all of the output ports along the source route. Since forwarding a packet from a source to a destination requires that the two hosts share a root node, the algorithm first compares the LMAC addresses of source and destination hosts in order to determine the common prefix of the address, which it then discards. Next, the chopped source LMAC address must be reversed, since the LMAC address follows a top down hierarchy, but the packet needs to follow a bottom up hierarchy in order to travel from the host to the root node. These two fragments are then concatenated together to produce the full path the packet must traverse in order to reach the destination. However, this string represents both the input and the output ports of each LESS switch through which the packet travels. Only the output ports are necessary for source routing, so the even numbers in the string are extracted in order to produce the final LSP label to be prepended to the IP packet (Wang et al., 2015).

Components of LESS Switch

The LESS switch is designed in such a manner that it can receive packets from a number of input ports, process them in an efficient manner with no packet loss, and then forward them to the next hop via the designated output ports. In order to manage the flow, it requires a queue to receive the packets and schedule them for processing, a forwarding engine in order to actually parse the headers and determine the output ports and finally a switching fabric to route the packets according to the LSP labels in the headers. Though the concept of the LESS system has been proposed as a whole in several publications, the details of the components involved in a LESS switch were proposed by Dr. Feng Wang at Liberty University and worked on by his research team, of which the author was part (Wang et al., 2015, Wang et al., 2017, Shao, Wang, Gao, Fujikawa, & Harai, 2016). No details have yet been published, though a manuscript has been submitted for publication.

Queue. The LESS switch contains two queues, one for incoming packets and one for outgoing packets. The first queue component of the switch receives the packets from the input port and utilizes a first in, first out scheduling algorithm in order to allow the packets to be processed by the switching fabric. After the packet has been processed by the LESS switch, the output queue passes it to the appropriate output port.

Forwarding Engine. The forwarding engine designed for the LESS switch encapsulates the main functionality of the switch. It contains the LESS header parser block, which bears the responsibility for popping the LSP switching label off, determining the output port number associated with the next hop of the packet,

decrementing the TTL bit, pushing the new switching label back onto the packet, and passing the modified packet to the switching fabric to be sent on the appropriate next hop.

Switching Fabric. The switching fabric receives the modified packet from the forwarding engine and puts it in the output queue in order to be sent on the next hop toward its destination with a rewritten packet header.

Rerouting Mechanism. The advantage of LESS is the ability of the system to dynamically reroute the packages in the case of a link or node failure. If the switching fabric determines that the output port is not available, the packet is sent to the rerouting mechanism in order to create a new LSP header that sends the packet to its destination.

Related Work

As mentioned in the background work, several other research teams are working on architectures and protocols to solve the challenges facing Future Internet. The Sourcey switching architecture provides a similar approach to the one taken in this thesis and the works on which it builds by using source routing to simplify the switch and remove the need for routing tables (Jin, 2016). Path Switching provides a SDN-type packet forwarding mechanism that mimics source routing by encoding the packet's route through the network in existing headers. The data center fabric proposed by Jyothi, Dong, and Godfrey (2015) encodes the source route in a single field, based on OpenFlow 1.3.The CONGA data center scheme uses source routing as a load balancing mechanism at the first hop of the flowlet in the uplink (Alizadeh, Esdell, Dharmapurikar, Vaidyanathan, 2014). The LESS Switch implementation differs from the aforementioned designs and works in that it does not require the source node to know the full topology of the network. Requiring each source node or host to know the full network technology

may impede the scalability of the architecture for large networks. Instead, in our implementation, the source route is generated simply from the source and destination LMAC addresses, which does not require a full network topology.

LESS Switch Design Framework

Overview

The LESS switch is designed in Quartus II software using Qsy, and functional block diagram is shown in Figure 6. Qsys is utilized because it provides a variety of predefined IP Cores for use in designing a hardware system. These IP cores correspond to, among other components, the peripherals and devices found on the DE4 development board. A simple graphic interface allows the programmer to quickly and easily connect the various components of the hardware and generate an image file as needed.

Each IP core is used to define a portion of the hardware to be used in the implementation. The four Triple Speed Ethernet modules correspond to the four Gigabit Ethernet ports on the DE4 development, and these serve as both the input and output ports for the implementation of the LESS switch. The four FIFO modules as well as the multiplexer create the queue that funnels the received packet into the LESS switching fabric. The forwarding engine is a custom created IP Core designed by Dr. Feng Wang. Finally, the packets are forwarded via the switching fabric to the appropriate output port via the output queue. Each output port channel is specified by the number in the LESS LSP switch label.

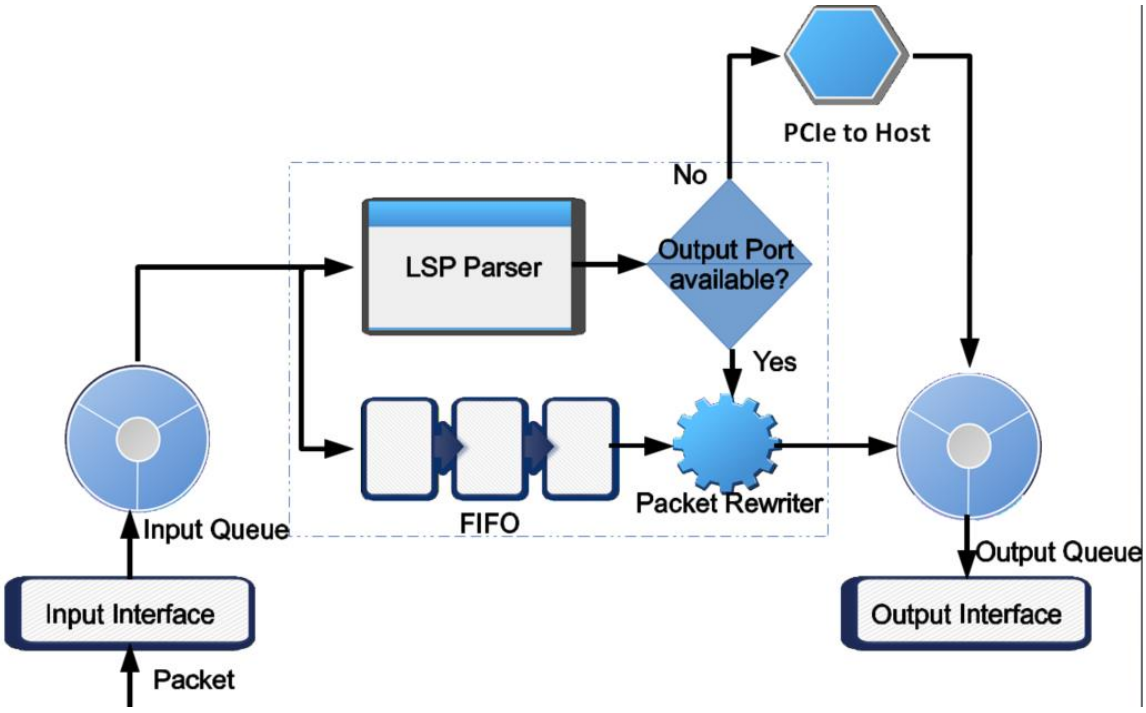

*Figure 6***.** LESS switch functional block diagram

FPGA-based Platform

The LESS switch is designed and implemented using an Altera DE4 FPGA board, pictured in Figure 7, which contains a Stratix IV, 4 Gigabit Ethernet ports, a PCIe card, SRAM and DDR2 DRAM. An FPGA based platform was chosen for its inherent flexibility, associated peripherals and memory, and low cost. The DE4 development board and Stratix IV also comes with a variety of reference design code and additional documentation to allow for ease of development. Altera provides further support through its online videos and tutorials, many of which are free with a registered account. The hardware portion of the LESS Switch is designed in Quartus II, with the major components of the queue, switching fabric, and forwarding engine being defined via the IP Cores provided by Altera.

Image can be found at

ftp://ftp.intel.com/Pub/fpgaup/pub/Intel_Material/Boards/DE4/DE4_User_Manual.pdf

Figure 7. The top view of the DE4 development board with peripherals labeled. Reprinted from "DE4 User Manual," Altera Corporation, 2012, San Jose, CA. Copyright 2012 by Altera Corporation

LESS header parser block

Using the example shown in Figure 4, a packet sent from the source Src, IP address 10.0.1.1 to the destination Dst, IP address 10.0.2.1would be given an LSP of 1.2.2 by the labeling algorithm. The packet is transmitted from the source to switch S_2. The LESS header parser block pops the label "1.2.2" from packet's LSP label stack and removes the first number in order to determine the outgoing port number (port 1) to forward the packet to the next hop. The new LSP, with the port number removed, is then pushed back onto the LSP label stack. Thus, in this diagram, the packet is forwarded to switch S_1 via output port 1 with a new LESS header that includes the new LSP of 2.2. Switch S_1, the root node of the network, repeats this process. The output port through which the packet must be forwarded is the new first number of the LSP, "2". This number is stripped from the LSP label, which is now pushed back onto the packet as simply "2". The packet is then routed to the next switch from the output port corresponding to the stripped number. In this case, output port 2 of switch S_1 is physically connected to switch S_3 so the packet is sent there. When switch S_3 receives the packet, it pops the LSP label off the stack for the final time to determine the output port. Since the TTL value of the packet now decrements to 0, the switching label is deleted and the packet is forwarded to the destination, which is linked to switch S_3 via its output port 2. This means that the

destination host receives a regular IP packet, free of the LSP label stack (Wang et al., 2017).

Depending on the needs of the packet, the LSP label stack can have a number of labels in it to ensure the packet travels from source to destination. The contents of the LSP label determines what action will be taken with the packet after it is processed. There are four general combinations of LSP labels, based on the values it can take. These LSP types and the actions that are taken on their headers are summarized in Table 1. Those are defined in Wang et al. (2017) as (1) Last Hop: the first LSP label is the last one with a TTL value of 1, (2) Last LSP: the first LSP label is the last one with a TTL value larger than 1, (3) Last Port: there are multiple LSP labels, and the first has a TTL value of 1, and (4) Top LSP: there are multiple LSP labels, and the first has a TTL value larger than 1. The category to which an incoming LESS packet belongs determines how the switch handles the pack. The following actions can be carried out on the LSP label: 1) decrement the TTL, 2) shift the first 24 bits of the top LSP to the left by one byte, 3) rewrite a new TTL value, 4) drop the last or top LSP label, and 5) modify Ethernet type. Each switch carries out one or more of these operations on the LSP label in the LESS packet header, until finally, when a packet reaches its destination, all the LSP labels have been removed from the packet so that the destination host only obtains an ordinary IP packet.

Table 1: Actions Taken to Modify the LSP Label based on type. Reprinted from "Towards reliable and lightweight source switching for datacenter network" by F. Wang, L. Goa, X. Shao, H. Harai, & K. Fujikawa, 2017, Manuscript submitted for publication. Image can be found at https://ieeexplore.ieee.org/document/8057152

Forwarding Module

The purpose of the LESS forwarding module is to use the number extracted from the LSP label in order to select the correct output port. The LESS forwarding IP core created originally by the research team at Liberty University for use in Qsys consists of the LSP parser block mentioned above and a packet rewriter. The LSP parser derives the output port from an LSP label as a packet arrives at the input interface. The port number is then checked to see if the port is available. If it is, the packet rewriter manipulates the LSP in accordance with the type of label, as mentioned above. If the output port is not available, the packet is transmitted to the host computer via the PCI Express to dynamically derive the LSP label for rerouting around the failed link or node.

PCI Interface

The PCIe component serves an important purpose in the LESS switch: although the basic functionality of the LESS switch is implemented in the FPGA design, it does not handle the dynamic packet rerouting that creates robust, reliable failure handling. Instead, this functionality is implemented in a host OS (Linux), which processes and forwards the rerouting packets. The PCIe component provides the bus to quickly and reliably transmit the packets to and from the host OS and LESS switch.

The IP Compiler for PCI Express serves as the IP core used for the LESS switch design, given that it is the one IP Core of the PCI category built to interface with the Stratix IV FPGA. Design of the PCIe interface in based on the example provided by Altera Corporation (2015b) in the document “Using PCIe on DE4”. The reference material for the IP Compiler for PCI Express provides both hard and soft implementation options, but only the hard implementation is available as a native endpoint for the Qsys design flow. The hard implementation uses embedded dedicated logic to implement the PCI Express protocol stack, including the lowest three layers. Qsys allows the user to select the parameters for their implementation according to what fits the specification from Altera. In order to implement the packet rerouting successfully, though, additional components are necessary to support the PCIe interface. Whereas packet flow through the LESS forwarding IP core occurs as a stream, the packet needs to be stored in memory on the host computer used to derive the rerouting LSP label. Therefore, a Scatter Gather DMA (SGDMA) is used in order to convert the packet stream to memory when sending it via the PCI Express to the host application and a second SGDMA is used to convert the packet format from memory back to stream before sending it to the output queue to be forwarded through the correct output port.

Host Application

The host application, which is intended to run on a Linux based system, exists in order to process and re-route packets that otherwise cannot be handled by the switch. The packets are forwarded from the LESS switch through an SGDMA via the PCI Express and placed into system memory. The application then will derive a rerouting LSP label for the failed packet, which is then placed on the packet. Once the rerouting and rewriting

process is complete, the packet is passed back to the LESS switch via the PCIe for parsing and forwarding. At the time of publication, the host application for the system is still undergoing development, in order address challenges in finding the packet in system memory and determining packet length to ensure the entire packet remains intact and will be forwarded completely.

Performance Evaluation

Performance Metrics

Performance metrics are used to quantitatively assess the performance of a network, whether it be speed, reliability, or other factors, and to provide a standard set of parameters by which various networks, topologies, and protocols can be evaluated and compared. The first portion of this section defines various metrics by which the design of the LESS switch will be evaluated, and the second portion of the section describes the testbed used to test the switch implementation.

Latency. Latency is defined in computer networking as the time that it takes a bit to propagate across a link or channel. In this context, it refers to the amount of time that it takes a packet to enter the DE4 development board, be processed and forwarded, and then exit the DE4 board on its next hop. Fundamentally, latency measures the speed performance and efficiency of the LESS architecture.

Throughput. In computer networking, throughput's formal definition is the observed rate at which data is sent through the channel. In the context of the evaluation and measurement performed in this paper, the throughput also indicates the percentage of packets successfully forwarded to the correct output port versus those dropped or lost. This measurement will determine if all the LESS packets can be forwarded as expected

and speaks to the reliability of the LESS system architecture to be able to handle the packets appropriately.

Testbed

The testbed for the system will be composed of a network of computers in the Liberty University School of Engineering and Computational Sciences. One computer running a Linux operating system with a fully licensed version of Quartus II will be used to generate the file to program the FPGA. The Altera DE4 board is inserted into the PCIe slot in a second Linux system that served as the host computer, to which the packets that could not be processed are forwarded for handling. A third computer is used to generate the packets via Click with the LSP headers that are sent to the LESS switch for forwarding. Prior to transmitting packets, a driver designed for the PCIe enabled device is manually loaded into the kernel of the host operating system.

Three types of tests are designed to be run on the PCIe-based LESS system and results analyzed, in order to obtain a variety of performance metrics in order to evaluate the feasibility and reliability of the proposed architecture.

The purpose of the first test is to measure the latency of the switch and the packet processing delay time. The time from when the packet enters the DE4 development board for processing to when it exits the board on its way to the next hop is recorded and analyzed in order to determine the delay. Packets of various sizes with a variety of LSP headers are sent to the switch for forwarding in order to fully test the switch's latency under a variety of circumstances.

The second test intends to verify the throughput capability of the switch. Packets are generated via the Click software located on one of the test computers and are

forwarded to the LESS switch via the Gigabit Ethernet ports on the DE4 development board. The percentage of packets that are successfully processed will be calculated and recorded, in order to determine that the LESS switch is able to forward all the LESS packets. The dropped packets will also be noted for analysis additional analysis.

The final test to be run on the switch implementation measures the latency of packet re-routing; this is known as slowpass analysis. This test looks specifically at the packets that have to be forwarded to the host application for processing and records the delay between when they enter the switch and when they leave for the next hop on the way to the destination.

Current Status

Thus far, the PCIe design has been incorporated with the forwarding engine, switching fabric, and queues. Preliminary testing done by sending packets directly from a computer generating LESS packets through the LESS switch to the host computer via the PCI Express has shown that the PCIe bus functions. This has been verified by examining the status LEDs that indicate transmission of packets through the input port to the PCIe card. Additionally, checking the status of the kernel driver for the PCIe enabled LESS switch on the host computer indicates that there is traffic flow across the PCIe bus. Current challenges with the host application that is being developed have precluded testing of the entire system. The tests described in the testbed section above have been prepared and will be run on the system when the application has been successfully completed, and results will be published.

Conclusions and Future Work

As demonstrated in the introduction and background sections, significant effort and research is being dedicated to allow the continued growth and development of the Internet of Things. The National Science Foundation is funding multiple grants for collaborative teams to address the outstanding issues, and various national and international organizations are also seeking to promote solution research and development. Location basEd Source Switching focuses on addressing the scalability, reliability, and performance studies challenges facing the development of the Internet of Things. Current preliminary case studies done (Wang et al., 2017; Shao et al., 2016) suggest that LESS provides a viable solution to the issue. None of those studies, however, implemented the proposed dynamic rerouting for packet failure. This thesis served to research, propose, and design a mechanism through which packets to be handled to allow failures to be handled while maintaining switch simplicity. The PCI Express has maximum throughput capacity to reliably and quickly allow a packet to be passed from the forwarding engine to the host application and back again.

The National Academy of Engineering has identified fourteen Grand Challenges for Engineering in the 21st Century in order to further technological innovation and improve quality of life around the globe. While none of these challenges directly involve the Internet of Things, the success of achieving many of these goals, including advancing personalized learning, enhancing virtual reality, engineering better medicines, and advancing health informatics, are predicated on the assumption of a reliable, rapidly growing Internet. Therefore, addressing the challenges of scalability and reliability in a robust and sustainable manner remains imperative to ensuring the continued advancement of technology. LESS serves to contribute to this development.

References

Alizadeh, M., Edsall, T., Dharmapurikar, S., Vaidyanathan, R., Chu, K., Fingerhut, A., ... & Varghese, G. (2014). CONGA: Distributed congestion-aware load balancing for datacenters. *ACM SIGCOMM Computer Communication Review*, *44*(4), pp. 503-514. New York: ACM. doi: 10.1145/2740070.2626316

Altera Corporation (2012). DE4 user manual. San Jose, CA. Retrieved from ftp://ftp.altera.com/up/pub/Altera_Material/Boards/DE4/DE4_User_Manual.pdf

Altera Corporation (2014). IP compiler for PCI Express user guide. San Jose, CA. Retrieved from https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_pci_express.pdf

Altera Corporation. (2015). PCI express high performance reference design. San Jose, CA. Retrieved from https://www.altera.com/en_US/pdfs/literature/an/an456.pdf

Altera Corporation. (2015). Using PCIe on DE4. San Jose, CA. Retrieved from ftp://ftp.altera.com/up/pub/Altera_Material/11.1/Tutorials/Using_PCIe_on_DE4.pdf

Budruk, R., Anderson, D., & Shanley, T. (2003). *PCI express system architecture*. Boston: Addison-Wesley Professional.

Evans, Dave. (2011). *The Internet of Things: How the next evolution of the Internet is changing everything.* Retrieved from http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

Hari, A., Lakshman, T. V., & Wilfong, G. (2015). Path switching: Reduced-state flow handling in SDN using path information. *Proceedings of the 11th ACM*

Conference on Emerging Networking Experiments and Technologies. New York: ACM. doi: 10.1145/2716281.2836121

International Telecommunications Union. (2012). Recommendation Y.2060: Overview of the Internet of Things. Retrieved from https://www.itu.int/rec/T-REC-Y.2060-201206-I/en

Jin, X., Farrington, N., & Rexford, J. (2016). Your data center switch is trying too hard. *Proceedings of the Symposium on SDN Research*, 12. New York: ACM. doi: 10.1145/2890955.2890967

Jyothi, S. A., Dong, M., & Godfrey, P. (2015). Towards a flexible data center fabric with source routing. *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*. New York: ACM. doi: 10.1145/2774993.2775005

Khan, R., Khan, S. U., Zaheer, R., & Khan, S. (2012). Future Internet: The Internet of Things architecture, possible applications and key challenges. *2012 10th International Conference on Frontiers of Information Technology (FIT).* pp. 257-260. IEEE. doi: 10.1109/FIT.2012.53

Kohler, E., Morris, R., Chen, B., Jannott, J.i, and Kaashoek, M.F. (2000). The click modular router. *ACM Transactions on Computer Systems (TOCS), 18*(3), pp. 263–297. New York: ACM. doi: 10.1145/354871.354874

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., ... & Turner, J. (2008). OpenFlow: Enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, *38*(2), pp. 69-74. doi: 10.1145/1355734.1355746

Mishra, S., Singh, N. K., & Rousseau, V. (2015). *System on chip interfaces for low power design*. US: Morgan Kaufmann Publishers Inc.

Peterson, L. L., & Davie, B. S. (2011). *Computer networks: A systems approach* (5th ed.). San Francisco, CA: Morgan Kaufmann.

Shao, X., Wang, F., Gao, L., Fujikawa, K., & Harai, H. (2016). Distributed encoding for multiple-inherited locators to accommodate billions of objects in the Internet. *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS),* pp. 751-752. IEEE. doi: 10.1109/ICDCS.2016.76

Sunshine, C. A. (1977). Interconnection of computer networks. *Computer Networks (1976), 1*(3), pp. 175-195.doi: 10.1016/0376-5075(77)90003-4

Tan, L., & Wang, N. (2010). Future internet: The Internet of Things. *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE),* 5, pp. 376-380. IEEE. doi: 10.1109/ICACTE.2010.5579543

Wang, F., Gao, L., Shao, X., Harai, H., & Fujikawa, K. (2015). Compact location encodings for scalable Internet routing. *2015 IEEE Conference on Computer Communications (INFOCOM,)* pp. 666-674. IEEE. doi: 10.1109/INFOCOM.2015.7218435

Wang, F., Gao, L., Shao, X., Harai, H., & Fujikawa, K. (2017). Towards reliable and lightweight source switching for datacenter networks. Manuscript submitted for publication.