Scalable Cyber-Security for Terabit Cloud Computing

Jordi Ros-Giralt, Peter Szilagyi and Richard Lethin
Reservoir Labs, Inc.
632 Broadway Suite 803
New York, NY 10012, USA

Abstract—**This paper addresses the problem of scalable cyber-security using a cloud computing architecture. Scalability is treated in two contexts: (1) performance and power efficiency and (2) degree of cyber security-relevant information detected by the cyber-security cloud (CSC). We provide a framework to construct CSCs, which derives from a set of fundamental building blocks (forwarders, analyzers and grounds) and the identification of the smallest functional units (atomic CSC cells or simply aCSC cells) capable of embedding the full functionality of the cyber-security cloud. aCSC cells are then studied and several high-performance algorithms are presented to optimize the system's performance and power efficiency. Among these, a new queuing policy—called tail early detection (TED)—is introduced to proactively drop packets in a way that the degree of detected information is maximized while saving power by avoiding spending cycles on less relevant traffic components. We also show that it is possible to use aCSC cells as core building blocks to construct arbitrarily large cyber-security clouds by structuring the cells using a hierarchical architecture. To demonstrate the utility of our framework, we implement one cyber-security "mini-cloud" on a single chip prototype based on the Tilera's TILEPro64 processor demonstrating performance of up to 10Gbps.**[1]

Keywords—cyber-security; cloud computing; energy efficient; high speed networks; many-core processors

I. INTRODUCTION

The Department of Energy's is currently developing ESnet, the nation's fastest computer network designed specifically to support science. This initiative, managed by the Lawrence Berkeley National Laboratory, will be bringing ESnet swiftly into the 100 Gbps regime as "a key step to the DOE's vision of an eventual 1 terabit wavelength network to connect DOE facilities" [1]. ESnet will enable unprecedented levels of collaboration, but paramount to its success will be its capability to secure the scientific community against cyber attacks.

In designing secure communication systems to address large network infrastructures such as the ESnet, one needs to address at least two problems: first, data needs to be moved from one node to another as efficiently as possible (forwarding resources); second, data needs to be inspected and potential threats must be detected and resolved (analysis resources) before any damage is inflicted. Given a fixed set of resources, this inevitably leads to contention: allocating more resources to forwarding data necessarily leaves fewer resources available for analysis, and vice versa. A key to the design of safer networks is therefore the identification of an optimal balance (a sweet-spot) between forwarding and analysis resources.

[1] This work was funded by the US Department of Energy, contract number DE-SC0004400.

We propose to exploit the elastic properties of cloud computing to construct a scalable energy-efficient cyber-threat detection system that can flexibly and efficiently allocate forwarding and analysis resources based on the consumer's demand. Our design incorporates several high-performance "knobs" designed to make sweet-spot packet forwarding and analysis decisions and provides a framework to scale the performance of the cloud.

The problem of scaling up the performance of a cyber-threat detection system has traditionally being approached with a cluster-like architecture (e.g., [2]). While clusters provide a mechanism to increase the performance of the system, they lack the degree of elasticity provided by a cloud architecture. As this paper will show, this elasticity is key to the design of cyber-security systems that can sustain very high speed (up to terabit) packet processing rates.

II. ARCHITECTURE

A. Cyber-Security Cloud (CSC)

According to the National Institute of Standards and Technology (NIST), cloud computing is defined as [3]: "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics (on-demand self-service, broad network access, resource pooling, rapid elasticity, measured service), three service models (software as a service, platform as a service, infrastructure as a service), and four deployment models (private cloud, community cloud, public cloud, hybrid cloud)." In this paper we focus on the problem of building a specific type of cloud, one that is dedicated to the detection of cyber-security attacks. Our cloud will follow a platform as a service (PAAS) model—whereby the consumer will have control over the deployed cyber-security applications and their configuration—and it will provide support for any of the four deployment models. We will use the term *cyber-security cloud* or *CSC* to refer to this type of cloud. To introduce our CSC framework, we start with the following definitions.

Definition 1. Cyber-security relevancy. We will say a network packet is cyber-security relevant if the information it contains alters our understanding of the cyber-threats that can potentially affect our network assets.

Definition 2. Cyber data. We define cyber data as that part of the information carried by packets which are cyber-security relevant. For instance, an example of cyber data can be the event "connection C is inflicting a denial of service attack on network asset A." We will say that a packet carries a positive amount of cyber data if it is cyber-security relevant. If

a packet is more cyber-security relevant than another one, then we will say that it carries more cyber data. Note that in this paper we do not intend to define the meaning of relevancy, which is an application-dependent concept.

Definition 3. Recognizable cyber data of a system. The degree to which a system is capable of recognizing cyber data from an input stream of packets.

Definition 4. Selection capacity of a system. The amount of cyber data that a system can detect per unit of cost incurred in performing such detection. In our work, cost will refer to the processing time, space, energy, and capital equipment expenses incurred in the process of detecting cyber data.

Definition 5. Cyber-security Cloud. For the purpose of this paper, we will define a cyber-security cloud as a cloud that complies with NIST's definition and at a minimum satisfies the following two properties:

- *CSC-P1: elastic selection capacity.* The selection capacity of the CSC can be elastically provisioned and released to scale rapidly commensurate with demand. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be appropriated in any quantity at any time.[2]
- *CSC-P2: elastic recognizable cyber data.* The degree of recognizable cyber data of the CSC can be elastically increased or decreased according to consumers' needs. A consumer operating in a more or less demanding cyber-security environment will be able to adjust the degree of recognizable cyber data upward or downward in the CSC.[3]

The concepts of elastic selection capacity and elastic recognizable cyber data are illustrated in . The CSC is modeled as a virtual system capable of elastically increasing or decreasing its selection capacity and of arbitrarily recognizing more or less cyber data to a level that meets a specific consumer's demand. Notice that the second property (CSC-P2) relates to the PAAS model, whereby the user can flexibly add more cyber-security applications (denoted as "apps" in) to the cloud to increase the degree of recognizable cyber data. Our work focuses on the construction of a framework to build cyber-security clouds that satisfy the properties of CSC-P1 and CSC-P2. In this paper, we will focus on CSC-P1. For an extended version of this paper including our work on CSC-P2, refer to [4].

B. Initial Assumption: the Heavy-Tailed Nature of Traffic

In order to derive a framework for the construction of cyber-security clouds, we start with an understanding of a network traffic characteristic that is essential and unique to our problem and which can help identify interesting trade-offs in our design. Our initial observation rests in the well documented "heavy tail" nature of most forms of network traffic (e.g., [5]). This observation can be expressed as follows: "In a computer network, very often, to an external observer, most of the relevant cyber data resides in a small portion of the total traffic." For instance, in [6] the authors find that between 55% and 90% of the total traffic is irrelevant to the scope of their analysis—using our notation, we would say that between 55% and 90% of the traffic carries no cyber data. This property, commonly referred as the *heavy-tailed nature of traffic*, provides a basis for the trade-off decisions in our design.

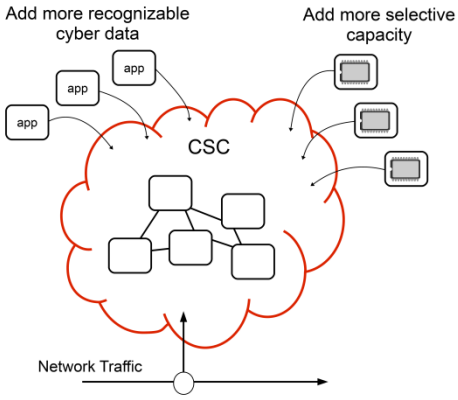

Figure 1. A model CSC with elastic recognizable cyber data and elastic selection capacity

In our work, we will define the heavy-tailed nature of traffic as follows. Let b_n be the n-th byte transmitted in a given connection and let $I(b_n)$ be the amount of cyber data carried by such byte. We will then assume that, often, $I(b_n)$ is a monotonically decreasing function.

As other authors have noted (e.g., [6]), from a security analysis perspective, the above definition of heavy tails is based on the observation that often the very beginning of a connection contains most of the cyber data of interest. For instance, it is at the beginning of a connection that session identification information and credentials are exchanged for most protocols (HTTP, SIP, FTP, etc.). This concept of heavy tails is graphically illustrated in .Building Blocks: Forwarder, Analyzer and Ground

Our framework is composed of the following set of elemental building blocks needed to implement the functions in the CSC ():

- *Ground.* The ground is responsible for dropping traffic. The decision to include a ground in our design stems from the need to drop irrelevant traffic—for instance, the tail of a connection that experiences heavy tailing—as a mechanism to reduce energy costs.
- *Forwarder.* Forwarders are responsible for splitting and mapping traffic onto processing resources in a way that the overall system's selection capacity is maximized.
- *Analyzers.* Analyzers perform the actual traffic analysis, receiving as input a stream of packets and generating as output the detected cyber data.

The basic functional unit within our framework is constructed using these three core building blocks. This basic unit uses the simple network configuration presented in -a, which we refer to as an atomic CSC cell or, abbreviated, an aCSC cell (pronounced "ax cell").

[2] Property CSC-P1 is equivalent to NIST's essential characteristic "rapid elasticity" [3], particularized to the specific case of cyber-security.

[3] Property CSC-P2 can be seen as a specific case of NIST's service model "software as a service" [3], particularized to the case of cyber-security.

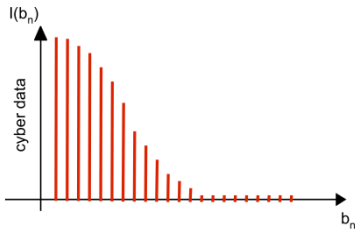

Figure 2. A single connection with a heavy tail

Within an aCSC cell:

- The forwarder receives input traffic and can decide to (1) drop the traffic, (2) forward the traffic out, (3) forward the traffic to an analyzer, or (4) a combination of these actions.
- The analyzer receives traffic and extracts cyber data, which can then feed back to the forwarder. The forwarder can optionally use this cyber data to make forwarding decisions.

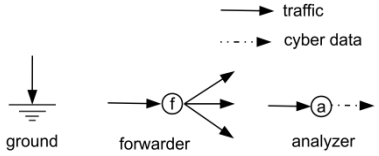

Figure 3. Building blocks

This network configuration allows for two modes of operation: ex ante and ex post. In the ex ante mode (-b), a packet is forwarded after the analyzer has inspected it and conveyed new cyber data back to the forwarder. In the ex post mode (-c), a packet can be forwarded before the analyzer processes the packet. The ex post configuration provides the ability to decouple the forwarding path from the analysis path. In real-time high-performance applications, this can be an important feature to ensure that the throughput of the cell does not deteriorate when the analyzer node becomes overloaded. This approach trades a small feedback delay for performance, since packets are forwarded as fast as they are received without waiting for the analyzers' feedback. In per-connection packet forwarding applications, a delay in executing a forwarding policy simply means that for a short period of time, a connection will operate using a non-intelligent (without feedback) default forwarding policy; then when the forwarder receives the cyber data (ex post), the connection will start to operate under the new (more intelligent) forwarding policy. Due to its high-performance characteristics, in our work we will focus on the ex post configuration.

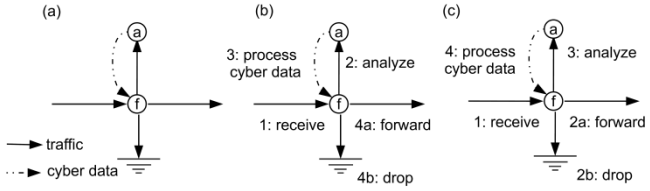

Figure 4. (a) The network topology of one aCSC cell with (b) ex ante and (c) ex post configurations

C. High Performance Data Structures and Knobs

Before we describe how to use aCSC cells to build cyber-security clouds of arbitrary sizes (Section F), we analyze the potential bottlenecks in the cell and the implementation of several high-performance knobs to alleviate them.

-a presents a more detailed, logical view of an aCSC cell for the ex post configuration. A first observation is that this new schematic implements the analyzer element using a set of n sub-analyzers, each one identical to the others. While this is without loss of generality (the drawings presented in -a and -c are logically equivalent), this approach provides an additional mechanism to scale the selection capacity of an aCSC cell up or down (in discrete steps by increasing or decreasing n).

The schematic in -a exposes a set of data-sharing critical regions that can potentially become a performance bottleneck. For each of these, we have designed and implemented a high-performance data structure.

- *Analyzers' input queues (-a/circle 3).* Upon receiving a packet, the forwarder can decide to pass it to one of the analyzers. Since this queue is written by a single writer (the forwarder) and read from a single reader (one of the analyzers), it can be implemented using a classic lockless circular queue [7].
- *Forwarder's input queue (-a/circle 4).* Analyzers need to convey feedback to the forwarder. Since this queue is written by multiple writers (the analyzers), it cannot be implemented using a lockless queue. To resolve this potential contention bottleneck, we propose a new hash table that provides zero-locking contention (lock-free) in exchange of a low probability of false negatives [4]. We refer to this data structure as LF$^-$. (A nested acronym from the words "lock-free with low false negatives".)
- *Forwarder's queuing policy (-a/circle 3).* We also consider the problem of designing an optimal queuing policy implemented by the forwarder when queuing packets into each analyzer. To maximize the degree of selection capacity in the aCSC cell, the optimization criterion for this policy should be the "maximization of cyber data forwarded to the analyzer." To this end, we present a new queuing policy referred as TED (an acronym that comes from the words "tail early detection") which exploits the heavy-tailed properties of the traffic ().

In our extended paper [4], each of these three critical regions is studied in detail. For the sake of brevity, in this paper we will only describe one of them: the forwarding queuing policy.

D. Forwarder's Queuing Policy: Optimal Tail Early Dropping (TED) for Hierarchical Memory Architectures.

Under heavily stressed conditions the analyzers may not be able to handle the full volume of network traffic. Therefore, a "knob" is needed to throttle and select which packets are handed to the analyzer and which are not. When analyzers are able to keep up with the input traffic, this knob should be fully opened. As traffic volume increases and analyzers begin to fall behind, the knob should be turned down to reduce the number of packets forwarded to the analyzers. Our objective is to design a knob that dynamically adjusts the amount of traffic throttled into the analyzers pool while maximizing the selection capacity of the aCSC cell.

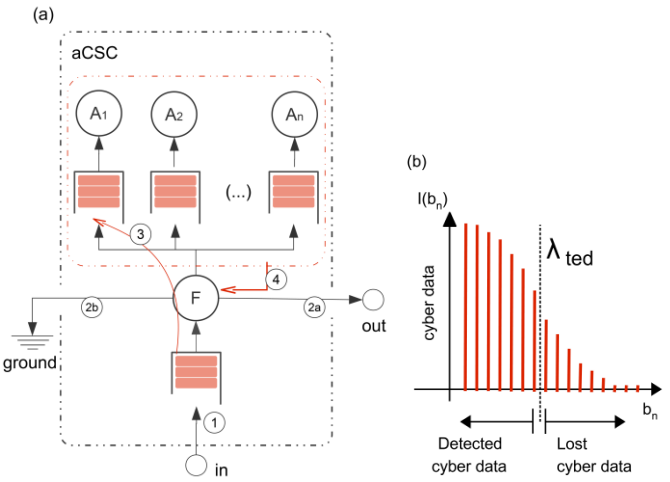

Figure 5. (a) An aCSC cell instance, (b) behavior of the TED static queuing policy for a given connection

To implement this controller, we propose a new queuing policy called *tail early dropping* (TED). TED is similar in concept to the well-known queuing policy RED (random early detection [8]) albeit with a different objective. While RED aims at maximizing fairness amongst connections and reducing effects such as TCP global synchronization, TED's objective is to proactively drop packets so as to maximize the overall selection capacity of the system. Toward this goal, we propose to use a queuing policy designed to exploit the heavy-tail properties of network flows (). We start by introducing first a static version of the TED queuing policy:

TED static queuing policy. Let λ_{ted} be a non-negative integer value and c a connection flowing through an aCSC cell. Upon receiving a byte b_n from connection c, if the total number of bytes received from this connection is larger than λ_{TED}, then drop b_n. Otherwise, queue it.

This static policy (illustrated in -b) has the advantage of being simple to implement—it only requires tracking of the number of bytes received from each connection—and prioritizes bytes which carry higher cyber data according to the heavy tails principle. A main limitation of this static policy is its lack of flexibility: since in general the degree of cyber data carried by traffic changes with time, an optimal λ_{ted} ought to also change with time. The key to the design of a good TED queuing policy is the identification of a control loop that can dynamically adjust the value of λ_{ted} towards tracking a certain optimal value λ_{ted}^{opt}. Before we can identify such optimal value, we need to first define the meaning of optimality in the context of a TED queuing policy.

While we acknowledge that there can be a variety of optimization criteria, given their wide-spread use, we focus our attention on finding a good sweet-spot for architectures that are based on hierarchical memory. These types of architectures are found in most of today's high-performance network appliances such as routers, intelligent switches, and general purpose computers—including the Tilera board on which we have implemented our CSC. (Described later in Section III.)

Our optimal TED policy derives from the following observation: in hierarchical memory architectures, memory accesses are, in general, one order of magnitude slower than cache accesses. For instance, in the Tilera architecture, a memory access (80 cycles) is ten times slower than a L2 cache access (8 cycles) and forty times slower than a L1 cache access (2 cycles) [10].

Lemma 1: memory and cache regimes. Assume an aCSC cell implemented on a hierarchical memory architecture. Let I_{in} be the amount of cyber data per unit of time received by the cell and let π_c be the amount of cyber data per unit of time that it can actually process. Then:

- If $\pi_c < I_{in}$, there exists a value λ_{ted}^h such that for any $\lambda_{ted} > \lambda_{ted}^h$, the analyzers in the aCSC cell will be accessing packets from memory (cache miss).
- For any value of π_c, there exists a value λ_{ted}^l such that for any $\lambda_{ted} < \lambda_{ted}^l$, the analyzers in the aCSC cell will be accessing packets from cache (cache hit).

Proof. For the sake of brevity, refer to [4].

Lemma 1 reveals the existence of a sweet-spot: On one hand, λ_{ted} ought to be large to ensure that the analyzers receive enough cyber data; on the other, if λ_{ted} is too large, the cell will operate in memory regime, penalizing its overall selection capacity. A sweet-spot therefore comes from identifying the largest value of λ_{ted} that will still allow the cell to operate within the boundaries of the cache regime—that is, without entering the memory regime. We refer to this value as λ_{ted}^{cache}.

The general form of our TED queuing algorithm can be expressed as follows:

Constants: Δ_{ted}, t_{ted}, λ_{ted}^i.

Step 1. Start with $\lambda_{ted} = \lambda_{ted}^i$, for an arbitrary λ_{ted}^i

Step 2. If the cell is operating in memory regime, then keep reducing λ_{ted} by a value $\Delta_{ted} > 0$ until the cell starts to operate in cache regime.

Step 3. Wait a period of time t_{ted} and then increase λ_{ted} by Δ_{ted}. After that, return to step 2.

This algorithm, as illustrated in , is designed to track λ_{ted}^{cache} using a conservative iterative approach: with instant (effectively infinite slope) decrease to converge from memory regime to cache regime and with a slow periodic increase (with a slope of Δ_{ted} / t_{ted}). illustrates the two regimes and the behavior of the TED control algorithm.

E. Cloud Scalability

From a functional perspective, an aCSC cell satisfies the properties of the cloud (Definition 5) albeit at a much smaller scale, delivering a bounded amount of selection capacity. Being functionally equivalent, aCSC cells can be used as building blocks for the construction of larger clouds. This can be achieved using a fractal-like architecture as illustrated in Figure 7. The schematic shows that on a macroscopic view, the functional elements of the cloud are equivalent to those of an aCSC cell, with the forwarder function receiving and mapping traffic onto the analyzer function. Zooming into the cloud, we can obtain a new microscopic view, showing that the cloud itself is made of smaller aCSC cells. Each of these elements could be further zoomed in, disclosing a new layer of smaller aCSC cells, resembling the structure of a Russian

nesting doll. (For a more detailed description of the benefits of this self-similar architecture, refer to [4].)

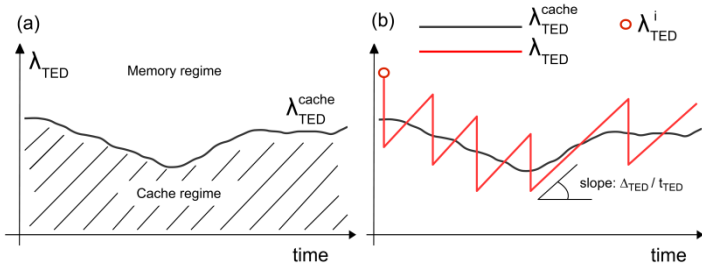

Figure 6. (a) Example of cache versus memory regimes; (b) TED control algorithm used to track λ_{ted}^{cache}.

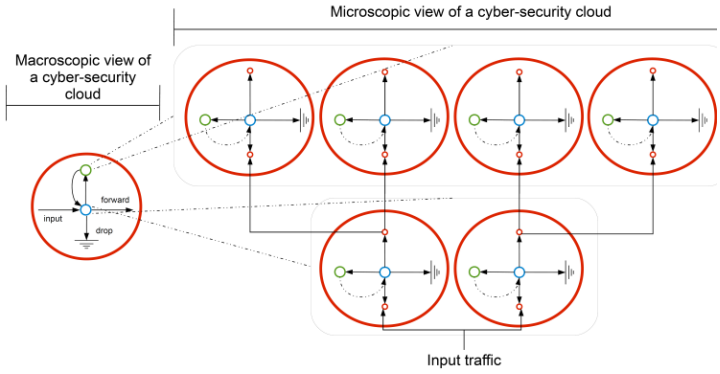

Figure 7. A CSC can be constructed using aCSC cells organized in a fractal-like manner.

This hierarchical architecture for cloud computing has the virtue of providing a new degree of elasticity. By adding more aCSC cells to the cloud and by structuring them using a higher number of fractal levels, one can progressively increase the performance of the cloud. In our design, we propose a cloud structure formed with the four levels shown in the next table:

Table 1. Levels of the CSC

Level	Description and interconnect type
Core	Multiple cores in a single processor forming one aCSC cell. A cell consists of $n+1$ cores, 1 core used as a forwarder node and n cores used as analyzers. Packets are shared/forwarded via main memory or cache.
Processor	For processors with m cores, mapping of $\lceil m/(n+1) \rceil$ aCSC cells onto a single processor to form a larger aCSC cell. Packets are shared/forwarded via main memory or cache.
Box	Multiple processors embedded into a single box. Packets are shared/forwarded via a processor interconnect (e.g. PCI).
Network	Multiple boxes connected over a network fabric. Packets are shared/forwarded over the network.

III. IMPLEMENTATION

We have implemented a cyber-security "mini-cloud" using the framework presented in this paper on a single-chip prototype providing processing speeds of up to 10Gbps. We refer to this prototype as CSC10. On the software side, while the concepts explained in this paper are general and can be implemented using any high-level programmable language, our CSC implementation is based on Bro [9], an open source network analysis framework developed by the International Computer Science Institute (ICSI) in Berkeley, CA, and the National Center for Supercomputing Applications (NCSA) in Urbana-Champaign, IL. On the hardware side, we use a Tilera's TILEncore PCI-Express card, which comes with 64 cores (called tiles) running Linux and several high-performance features including the following:

- NetIO: a Tilera-proprietary high-speed packet capturing module capable of DMA-ing packets directly onto the application (bypassing the Linux kernel).
- Hardware-aided load balancing: IP tuple hashing and load balancing performed in hardware. This feature is used by the analyzers that require per-flow processing.
- Zero overhead Linux: this feature allows the programmer to specify a set of "dataplane" tiles, each of which runs a single user-space process without incurring any Linux system overhead.

(For a more detailed description of these hardware-aided optimizations, refer to our extended paper [4].)

The TILEPro64 network processor comes with two 10Gbps XAUI ports and each of its tiles is clocked at 700MHz and includes L1 and L2 data caches of 16KB and 64KB, respectively. presents a 2-level mapping (cores and processor) of the CSC architecture onto the TILEncore chip. The first level has eight aCSC cells, each with five analyzers and one forwarder, for a total of 48 tiles (six tiles in each aCSC cell). The second level consists of one single-chip aCSC. In addition to allocating 48 tiles for mapping the functions of the mini-cloud, two other sets of tiles are used as follows:

- Ingress Packet Processor (IPP) tiles: Four tiles are used to run the ingress packet processor, a Tilera component part of NetIO which is used to accelerate the reception and transmission of packets [10].
- OS tiles: The rest of the tiles, are used to run the operating system and housekeeping functions.

IV. PERFORMANCE TEST

Figure 9 provides a performance benchmark of our CSC10 mini-cloud. These results were obtained by stressing CSC10 using a cluster of HTTP clients and servers generating traffic at 10Gbps. The figure shows two graphs: the total throughput sustained by the CSC10 (Figure 9-a) and its degree of selection capacity (Figure 9-b), both displayed as a function of the number of forwarders and analyzers being used. In this benchmark, our definition of selection capacity is a measure of the number of events that the Bro application can detect per unit of time. Specifically, we measure the number of events of type "this HTTP connection is a GET request" emitted by a Bro script (this script corresponds to one "app" in the cloud as illustrated in). These graphs illustrate two important concepts. First, the performance of the forwarding path (Figure 9-a) is approximately independent of the analysis path (since throughput is flat regardless of the number of analyzers per forwarder). Second, for the specific application used in this benchmark, a number of analyzers between two and four for each forwarder is sufficient (since selection capacity is flat beyond four analyzers—Figure 9-b). For a more detailed set of performance results, refer to [4].

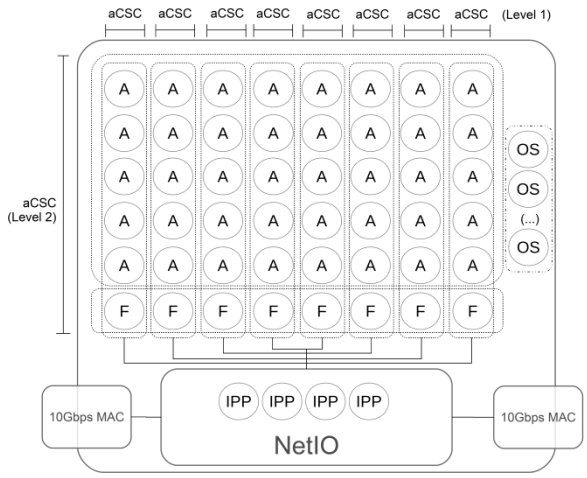

Figure 8. CSC10 on a single TILEncore chip

V. TERABIT CYBER-SECURITY: CSC1000

To illustrate how the proposed architecture can be used to scale up the CSC, we describe a possible design of a terabit-scale CSC (referred as CSC1000). The terabit design is based on a 4-level hierarchical architecture starting from a Tilera Gx processor. The Gx is the newest generation of processors which comes with cores clocked at a frequency up to 1.5GHz and a faster implementation of NetIO called mPIPE [10]. Since the older TILEPro64 chip can sustain 10Gbps (see Section IV), we make the assumption that the Gx will be capable of sustaining 20Gbps. (This could be an average estimate, since the Gx is designed to sustain raw packet forwarding rates up to 40Gbps.) A Gx chip provides the infrastructure for the first and second fractal levels, mapping multiple aCSC into its many-core architecture. Tilera offers a 1U rack-mountable device incorporating four Tilera's TILE-Gx processors, totaling 144 to 288 cores, which would yield a cloud performance of 80Gbps. Each Gx processor can be connected to the PCI-Express switch via an x8 PCI-Express 2.0 link, with a capacity of 32Gbps (per-link) in each direction (full duplex). Each processor is expected to require no more than 15Gbps to forward packets to other processors, although 20Gbps may be required if the input load balance is pessimal. (Each processor takes in one quarter of the total 80Gbps traffic, and, being one of four processors, is expected to forward three quarters of the traffic it receives to other processors.) The 1U device constitutes the third level. Finally, 12 1U devices (or more) can be connected in a network to deliver a total throughput of 960Mbps, for a full four-level architecture.

VI. CONCLUSIONS

In our work, we use a cloud computing architecture to address the problem of scalable cyber-security for very high-speed networks (up to terabit per second rates). The key to sustaining high rates resides in the design of a system that can efficiently triage those parts of the traffic that are most cyber security-relevant and avoid spending precious resources processing traffic components that are less relevant. We argue that the elemental functions needed to implement a cyber-security cloud are three, forwarders, analyzers and grounds, and propose a new design based on aCSC cells—small configurations of these three elements that embed the full functionality of the cloud but at a much lower scale. To control the flow of packets within one aCSC cell, we present tail early dropping, a new closed-loop queuing policy designed to make proactive packet dropping decisions toward maximizing the selection capacity of the cell. We then show how aCSC cells can be used to construct larger clouds using a fractal-like architecture, and provide a configuration example of a cyber-security cloud to sustain terabit rates. To demonstrate the framework, we have implemented a two-fractal level mini-cloud using a Tilera TILEPro64 processor supporting packet processing rates up to 10Gbps.

Our next step is to scale up our implementation to support rates of 100Gbps and beyond by increasing the number of fractal levels using the new generation of Tilera Gx processors and the design described in Section V.

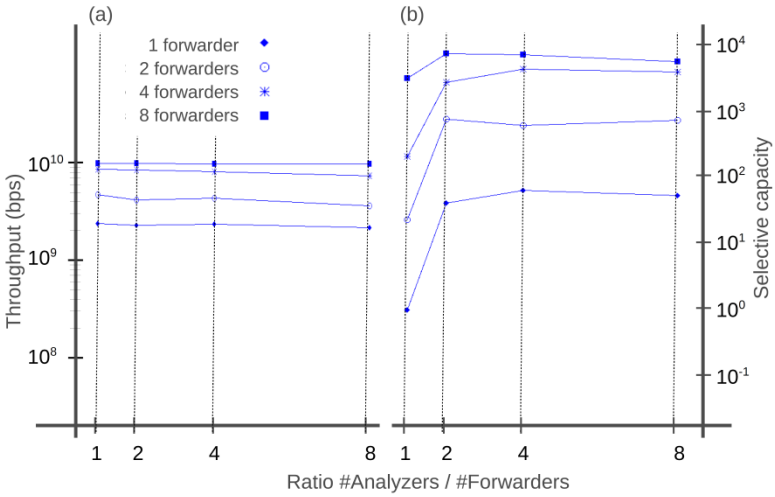

Figure 9 Performance of one CSC10 mini-cloud

REFERENCES

[1] "Department of Energy Builds National 100GigE Research Net," Press Release, July 13, 2011.

[2] M. Vallentin, R. Sommer, J. Lee, C. Leres, V. Paxson, B. Tierney, "The NIDS Cluster: Scalable, Stateful Network Intrusion Detection on Commodity Hardware," Recent Advances in Intrusion Detection, 2007

[3] National Institute of Standards and Technologies, "The NIST Definition of Cloud Computing," Special Publication 800-145, September 2011.

[4] J. Ros-Giralt, P. Szilagyi, R. Lethin, "Scalable Cyber-Security for Terabit Cloud Computing (Extended Version)," Reservoir Labs Technical Report, May 2012.

[5] V. Paxson, "Empirically derived analytic models of wide-area TCP connections," IEEE/ACM Transactions on Networking, 2(4):316–336, August 1994.

[6] Jose Gonzalez, Vern Paxson, Nicholas Weaver, "Shunting: A Hardware /Software Architecture for Flexible, High Performance Network Intrusion Prevention," ACM Conference on Computer and Communications Security, November 2007.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, "Introduction to Algorithms," The MIT Press; 3rd edition, 2009.

[8] Floyd, Sally; Jacobson, Van. "Random Early Detection (RED) gateways for Congestion Avoidance". IEEE/ACM Transactions on Networking 1(4): 397–413, August 1993.

[9] Vern Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," Computer Networks, December 1999.

[10] Carl Ramey, "TILE-Gx100 ManyCore Processor: Acceleration Interfaces and Architecture," Tilera Corporation, August 2011.