Edge-enabled Distributed Network Measurement

V. K. Cody Bumgardner
Department of Computer Science
University of Kentucky
Lexington, Kentucky, USA
Email: cody@uky.edu

Caylin Hickey
Department of Computer Science
University of Kentucky
Lexington, Kentucky, USA
Email: caylin.hickey@uky.edu

Victor W. Marek
Department of Computer Science
University of Kentucky
Lexington, Kentucky, USA
Email: marek@cs.uky.edu

***Abstract*—In the area of network monitoring and measurement a number of good tools are already available. However, most mature tools do not account for changes in network management brought about through Software Defined Networking (SDN). New tools developed to address the SDN paradigm often lack both observation scope and performance scale to support distributed management of accelerated measurement devices, high-throughput network processing, and distributed network function monitoring.**

In this paper we present an approach to distributed network monitoring and management using an agent-based edge computing framework. In addition, we provide a number of real-world examples where this system has been put into practice.

I. Introduction

Telecommunications providers and research-focused organizations have made use of Software Defined Networking (SDN) [1] techniques to address the rapid growth of data transmission. In traditional communications networks, traffic management is a distributed task relying on device-level decision making. SDN separates control and data planes, allowing traffic management decisions to be made centrally, providing methods for the use of so-called commodity network devices in the construction of high-speed networks. Through SDN-enabled networks, so-called "Big Data" flows [2] can be identified and managed independently. In addition, Data Transfer Nodes (DTN) [3] can be used to rapidly move large volumes of data over large distances.

Data transmitted transcontinentally and often intercontinentally will pass through network exchanges, where two or more networks join. One such example is StarLight [4], which functions as a global Software Defined Network Exchange (SDX) [5], peering hundreds of national and international networks. Data generated from telescopes in São Paulo, Brazil destined for Singapore might first pass through AMLight [6], an international network exchange serving South America, and then through the StarLight exchange on its way to Asia. While SDN provides the ability to monitor (replicate to a switch port) traffic with flow-level granularity, one must first know the parameters of the data they wish to isolate. While there are cases where we have an *a priori* knowledge of the flows we would like to monitor, often we must inspect aggregate network traffic. Traditionally, network hardware generated high-level statistics, such as Netflows [7], that could be used to detect and describe traffic, including anomalies. However, in SDN networks simple low-level rules might be used to establish high-speed communications that traverse exchanges, such as with our South America to Asia example, which does not produce actionable high-level network statistics. A number of monitoring frameworks have been proposed that make use of SDN functions. Some SDN monitoring systems continuously poll [8] SDN controllers for data and others make use of periodic updates pushed [9] from controllers. However, there are inherent performance limitations [10], [11] in using central SDN controllers to simultaneously make high-rate traffic decisions and perform network measurements. To lessen the demand on central SDN controllers, it was proposed [12], [13] moving a subset of data collection and analysis tasks to SDN-enabled devices. While metric gathering on the device-level is possible, deep packet inspection [14] is much more computationally demanding. A single 100G network link can transmit 148 million packets per second [15], leaving nanoseconds to process a packet, thus requiring computational demands for single port analysis at the limit of current general purpose processors. A number hardware accelerated [16], [17] network analysis architectures and standalone implementations have been proposed for 100G and beyond network analysis.

Performance concerns aside, not all relevant measurement information is available from SDN controllers and devices. With the rise of Network Function Virtualization (NFV) [18], a trend that moves network functions from monolithic network devices to virtual instances, high-level information related to network functions might not be observable through low-level SDN network monitoring systems. In addition, there might be more than one independent SDN or NFV controller participating in a specific network flow. For example, an OpenStack [19] cluster might provide Network Address Translation (NAT) [20], IPv6-to-IPv4 protocol translation [21], and internal SDN networks that would not be observable through an external central SDN controller.

To address the limitations of a central SDN controller to observe the state of distributed environments a number of Edge- and Fog-Enabled SDN tools for control plane optimization [22], [23] and service orchestration [24], [25], [26], [27] have been proposed. Building on previous experience in distributed high-rate network monitoring [28], we propose an edge-based distributed network monitoring system developed using the Cresco Framework [29].

978-1-5386-3227-7/18/$31.00 ©2018 IEEE

II. System Model

We approach network monitoring and measurement as a streaming data problem, where information is generated, processed, and communicated between heterogeneous systems. We find it convenient to think of streaming data problems in the context of directed graphs, where *nodes* function as computational units and *edges* represent data streams.

Our edge-based network monitoring and measurement system was designed with the following characteristics in mind:

- *Heterogeneous devices and data*: The monitoring system must support a wide range of platforms ranging from low-power devices found within Wireless Sensor Networks (WSN) [30] to high-performance DTN nodes with hardware accelerators. Monitoring and measurement data might also come from unmanaged devices and systems. Systems must support sources of data coming directly from monolithic network devices, SDN controllers, accelerated capture devices, or any number of virtualized network functions.
- *Large number of data sources*: The number of high-powered captured devices will be constrained to a manageable figure through the limited number of associated physical links. However, the number of virtual data-generating devices could easily number in the thousands for even small computational clusters. Monitoring and measurement systems must be able to accommodate large numbers of managed nodes and data sources.
- *Data operations in proximity to data sources*: Where resources allow, data operations such as annotation, analysis, and stream merging should take place as close to the sources of data as possible. Specifically, operations such as the generation of new data streams through the correlation of localized events should take place under a universal clock (single device).
- *Abstract user-defined tasks*: Users should be able to abstractly define a hierarchy of complex measurement tasks, spanning geographically distributed devices and sources of data.
- *Function reuse*: There is a one-to-many relationship between data sources and potential user-defined data operations. Where resources allow, systems should allow simultaneous user-defined measurement tasks on data sources. In addition, where two replicated tasks are defined only one task should be provisioned, with resulting data replicated to two destinations. Likewise, if two identical data streams are defined, through an intermediate node, only one stream should be sent to the node, and the node should replicate the stream. Figure 1 provides a visual example of node and edge provisioning avoidance through stream replication.
- *Security and Privacy*: In many cases, raw streams of data are observed from managed devices. There are serious security and privacy issues related to not only the inspection of network traffic, but also to the derived results. Monitoring and measurement systems must themselves operate securely while providing methods to preserve the privacy of resulting data.

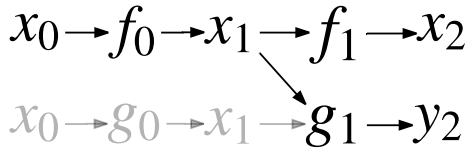

Fig. 1. Node and Edge Provisioning Avoidance

III. Solution

A. System Architecture

Our solution is implemented using the Cresco Framework [29], which provides a number of benefits used in the implementation of edge-enabled applications. Specifically, the framework provides location awareness, geographic distribution, support for large number of nodes, heterogeneous operations, function mobility, secure resource discovery, secure messaging, and pipeline (application) management functions.

While the complete description of the Cresco Framework is outside of the scope of is paper, it is important to recognize that Cresco operates as an agent-based [31] system implementing actor-model [32] concurrency. Agents manage resources, workloads, and establish both control (between agents) and data (between workloads) for communication networks. Figure 2 shows a local agent controlling network device, which streams resulting control and data plane data to remote locations. Resources might contain computational, network, and storage capacities ranging in scale from embedded system to large computational clusters. In this context, workload could refer to the management of an external resource, like a standalone measurement device, or the implementation of functions within the framework. For the remainder of this paper we will present workloads in the context of graphs, referring to them as nodes and workload-to-workload communication as edges. Monitoring and measurement tasks refer to one or more associated nodes and edges configured to provide one or more defined outputs.

B. Resource management

There exist a wide range of capacities between what might be found on edge devices and what is available in a public cloud. Likewise, there are great variabilities in the resource requirements between measurement tasks. Distributed measurement carries additional complication of determining if there is adequate communication capacity between resources to accomplish a desired task. We assess the capacity of a specific resource through agents. Agents report simulated benchmark performance, total capacity, and available capacity to a central service. Through agent-to-agent performance measurements, resource-to-resource edge communication performance is estimated. Likewise, nodes, and (by relation) measurement tasks, have either known (preconfigured) requirements or agents report observed performance utilization per

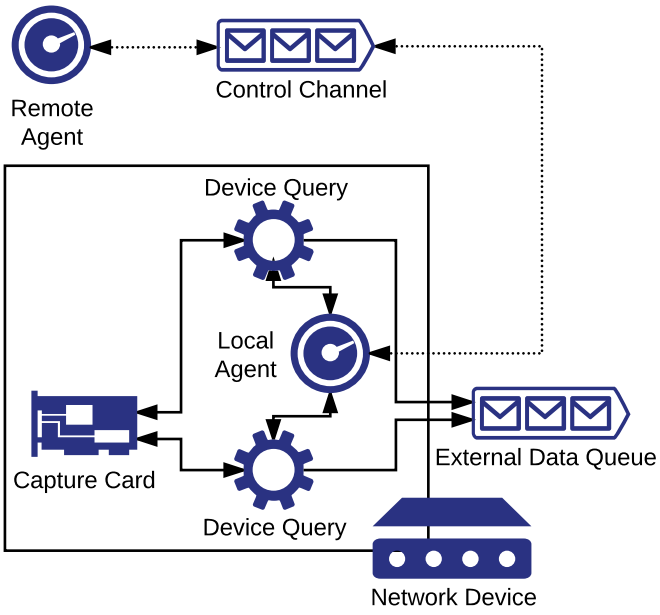

Fig. 2. Agent management of edge resources

node configuration to a central service. Benchmarked resource (node host) performance is compared to observed resource utilization (node instance) to provide synthetic computational node resource requirements. Key Performance Indicator (KPI) metrics, specific to node implementations, provide a unit-level indication of load, thus allowing the estimation of per-unit synthetic metrics.

Through the process described above we are provided a global view of resource availability, requirements, and utilization. In addition, we estimate specific node type requirements to the per-unit (transaction) level.

C. User-Defined Functions

We define a node function, or *plugin* in Cresco parlance, here to mean a piece of code that executes some task and can be run on a given Cresco agent. While a full overview of functions may be beyond the scope of this paper, we find it necessary to review some of their inherent benefits as they pertain to this work:

- The overhead required to define a function is minimal, implying that adapting an existing workload to the Cresco architecture does not require a large amount of framework-specific knowledge.
- As a consequence of their pre-defined nature, functions can be loaded on any Cresco agent with the appropriate resources and permissions required, meaning a call to a specific function can be issued and re-issued to one or many Cresco agents as required for the workload.

A number of functions, implemented as dynamically-loadable plugins that execute some piece of code, have been developed for the Cresco framework to aid in task execution on remote agents. A brief description of some existing functions related to network measurement follows:

1) Executor plugin: The executor plugin represents the most basic operation in a workload; namely, a plugin calls an existing function to be executed on the machine which hosts the initiating Cresco agent. In this manner, existing measurement tasks which have been pre-defined on a remote system can be quickly called as part of a described workload.

2) Container plugin: With the recent uptick in the adoption of containerized application delivery [33], functionality was included to remotely launch Docker [34] containers as means to distribute pre-defined, single-node workloads to remote agents without the need to pre-install all versions of software that may be required for network measurement.

3) CEP plugin: Complex Event Processing (CEP) involves the application of selection and aggregation rules to flows of data. This provides a natural approach for deriving high-level insights into the massive amounts of sensor data that can be generated as part of a network measurement workflow. One implementation of CEP, used in our CEP plugin, is Esper [35]. As Esper requires no external dependencies and only a small amount of resources, the plugin can launch its own instance of Esper and interact with data streams with one or more Event Processing Language (EPL)-defined queries as required by the overall workflow definition.

4) HBase plugin: One of the aims of edge computing, and Cresco as an application of the edge computing paradigm, is to reduce the complexity of data storage and processing in a given application deployment. To that end, we find it prudent to define methods to access some functions, which in the past have been primarily accomplished at a central datacenter. One such functional definition is a plugin to connect to HBase [36], an implementation of Google's BigTable paradigm that allows for large scale storage and processing on a cluster of nodes, to allow nodes closer to the edge to handle some of the heavy lifting related to either storing results from processing of data flows or referencing global data for use in processing at the edge as part of a workload.

5) Syslog plugin: Syslog [37] is a commonly used standard message format for server and device logging. We have implemented a syslog collector that takes in messages and converts them to format that is both parsable and accessible by the monitoring system. Messages are converted to a JSON format representing a syslog class that is directly consumable by other plugins such as the CEP plugin. Syslog messages can be directed to one or more downstream plugins allowing for stream enrichment or CEP.

6) SNMP plugin: Simple Network Management Protocol (SNMP) [38], is a protocol used primary by network devices for management. In addition, a number of software packages exist that implement SNMP on server and other device platforms. SNMP can be used to gather (pull) statistics on a wide-range of network interfaces, sensors, and associated devices. SNMPtraps are used to push events, such as changes in status notifications and alerts to monitoring systems. The SNMP plugin implements both polling metrics gathering and SNMPtrap collection. Both polling and trap metrics are converted

to JSON classes, which are pushed to message exchanges for consumption by one or more downstream plugins.

7) IPFIX plugin: Internet Protocol Flow Information Export (IPFIX) [7] is an IETF protocol used to describe network traffic flows. The IPFIX plugin implements a IPFIX (and Netflow v5) collector, converting IPFIX message to JSON classes, and pushing these classes to exchange for consumption by one or more downstream plugins.

D. Tasks Scheduling

The previous subsection describes a number of functional components managed by the Cresco framework. While it is possible to assign static component configurations to individual agents, Cresco provides a number of automated data collection and dynamic resource scheduling features. In order to take advantage of the framework, we must generate an abstract description of a measurement task in the Cresco Application Description Language (CADL) format. We provide an intermediate programatic monitoring and measurement interface used to interact with the Cresco framework, as described below:

- *Create Task*: Measurement tasks in JSON format are submitted to a central controller through a REST interface. An example of a task to perform high-rate flow measurement on two nodes, aggregate and label the reported measurements, and provide RESTful and queue-based output is shown in Listing 1. As previously mentioned, we convert the simple user input into a format used by Cresco to manage task-related plugins. Once the Cresco application definition is reported active on a Cresco controller, we report that the task is ready to be started.
- *Start Task*: On task start, the monitoring and measurement system confirms availability of distributed components and starts its own processes to listen for output. Depending on the task defined, output might be directed to a data exchange, to a log available through a RESTful API, or both. A task will continue to run for its specified time or when terminated.
- *End Task*: If a task is active, it can be terminated. On termination, active components are stopped, but configuration remains in place. In addition, any output generated when the task was active remains available.
- *Restart Task*: Task that have completed or were terminated can be restarted as long as the task was not removed.
- *Remove Task*: On task removal, all component data, configuration, and log information is removed from the system.

```
{"name": "my test",
 "duration": "60",
     "nodes": [{
         "type": "flow_measurement",
         "name": "UK Netflow",
         "location": "uk",
         "commands": "10 4"
     },{
         "type": "flow_measurement",
         "name": "FIU Netflow",
         "location": "fiu",
         "commands": "10 4"}]}
```

Listing 1. Measurement Task Example

During the task scheduling process, a number of system-derived configurations are used to connect components and data sources. Some components, such as measurement tasks that directly observe network traffic, require explicit location assignments. However. there are a number of event processing, data exchange, and data conversion functions that do not require specific location assignments. While a complete description is outside the scope of this paper, Cresco natively provides a number of scheduling optimization methods using Constraint Programming techniques. Within the intermediate scheduling system, implemented as part of this project, we restrict component configuration to specific locations, thus limiting the Cresco scheduler to a subset of resources. For example, if there was a single measurement device at a specific location, the Cresco scheduler would provision all related components on a single node. Alternatively, if a pool of resources were available in public cloud location, the Cresco framework would select the provisioning instance from the pool.

A number of steps are taken to reduce the computational and network costs associated with measurement tasks. We want to limit redundant computational executions and data flow. However, it is common for the same data and potentially data operations to be requested in full or in part simultaneously. For example, a subgraph related to the underlying measurement of traffic at X locations might exist as part of a number of users measurement tasks. Luckily, Cresco provides native de-duplication of matching component configurations and related data streams. We can take additional steps to structurally reduce redundant network traffic by assigning components with no location restrictions in proximity to related restricted components. For components with a single local input and output with location restrictions, the location assessment is trivial, assuming resources exist in the location/device for the assignment. For configurations with location-diverse inputs or outputs we assign known central-resource locations. Based on metrics maintained by Cresco the framework schedules resources from central-resource locations.

IV. Operations

There exist monitoring and measurement cases where it is important to include transient data from disparate devices in a data flow as it happens. For example, to monitor devices in a wireless network one might want to know the device address (mac), associated access point (AP), and IP address at the time of network flow generation. Since the AP may change through mobile association and IP address change through the address leasing process, this information must be captured

in the recorded flow. We have demonstrated the ability to capture such enriched data streams using or monitoring and measurement system. In our specific case, as deployed at the University of Kentucky, we gathered AP associations using the plugin described in Subsection III-C6, *SNMP plugin*, by capturing SNMPtraps from a central wireless controller. We obtained IP address lease notification through Syslog messages sent from our enterprise DHCP service. Syslog messages were collected using the plugin described in Subsection III-C5, *Syslog plugin*. Netflow information generated by distributed devices was collected by plugins described in Subsection III-C7, *IPFIX plugin*. Information from the related streams was merged through the CEP plugin described in Section III-C3, *CEP plugin*. Finally, the enriched message stream was recorded for further offline analysis using the HBase plugin described in Subsection III-C4.

In addition to data stream enrichment through the inclusion of transient data, we also have the ability, through the combination of Netflow plugin and CEP plugin (as described in III-C3), to watch for known patterns which may be indicative of behavior which can be useful for overall trend analysis such as:

- *Filter Flow Size*

```
select * from netFlow where bytes
       > [some value]
```

- *Filter Src/Dst*

```
select * from netFlow where ip_src =
  '[some ip]' and ip_dst = '[some ip]'
```

- *Top Talkers*

```
select ip_src, ip_dst, bytes from
 netFlow.win:time(5 min).
 ext:sort(10, bytes desc)
```

- *Flows Per Second*

```
select count(*) as fps from
 netFlow.win:time_batch(1sec)
```

- *Missing Flow Patterns*

```
select a.ip_src from pattern [ every
a=netFlow -> (timer:interval(10 sec))
and not netFlow(ip_src=a.ip_src) ]
group by a
```

As a consequence of the Cresco's use of queues to manage data-coordination and conveyance from various stages, as well as its inherent de-duplication of required functions provided during task scheduling, (as described in III-D), we can insert, join, and perform CEP operations on nearly any stage of data flows from one or many different monitored network locations performing any number of measurement tasks, as required by defined workflows. This provides flexible, scalable monitoring to large, distributed networking environments where oversight can be both difficult to deploy as well as reconfigure when new information is required.

As previously described, the management of accelerated devices is required for high-speed network analysis. In addition, monitoring making use of a number of distributed accelerated devices is needed. Not only must we perform local measurements, we must also synchronize measurement execution, coordinate data streams, and process the resulting data. This work was incorporated into related efforts in network capture presented at SuperComputing 2017, during which the solution described here was used to distribute measurement tasks and collect the resulting flow data for further analysis from geographically dispersed locations, annotated in Figure 3, in real-time. Capture devices were placed at Starlight, AMLight (FIU), and the University of Kentucky (UKY), as shown in Figure 3. In addition, UK provided computational resources used in device management, stream coordination, and stream processing.

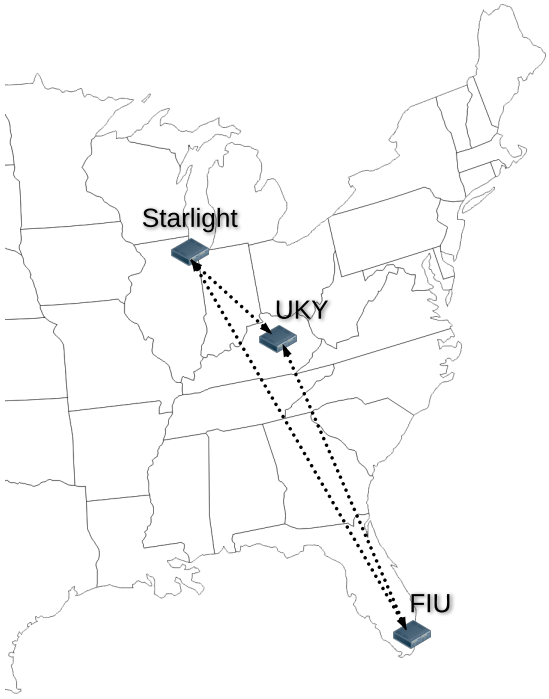

Fig. 3. Measurement Locations for SC 2017 Demonstration

Accelerated measurement tasks made use of functions described in Subsection III-C1, *Executor plugin*, stream coordination made use of functions described in Subsection III-C2, *Container plugin*, and stream processing was accomplished with the *CEP plugin* described in Subsection III-C3.

V. Conclusion

In this paper a motivation has been given for the necessity of edge-enabled monitoring and measurement of distributed network systems. We have presented techniques used to describe, model, and solve a number of problems related to distributed system monitoring. Additionally, we have shown how the presented data model can be used in edge-enabled resource, execution, and data management. Finally we have provided

several real-world operation cases where the described systems have been used.

We will continue to improve this work as the overall framework advances.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. ACI-1450937

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] N. McKeown, "Software-defined networking," *INFOCOM keynote talk*, vol. 17, no. 2, pp. 30–32, 2009.

[2] S. Rivera, M. Hayashida, J. Griffioen, and Z. Fei, "Dynamically creating custom sdn high-speed network paths for big data science flows," in *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*. ACM, 2017, p. 59.

[3] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: Sdn for big data and big data for sdn," *IEEE network*, vol. 30, no. 1, pp. 58–65, 2016.

[4] J. Mambretti, T. DeFanti, and M. D. Brown, "Starlight: Next-generation communication services, exchanges, and global facilities," *Advances in Computers*, vol. 80, pp. 191–207, 2010.

[5] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "Sdx: A software defined internet exchange," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 551–562, 2015.

[6] J. Ibarra, J. Bezerra, H. Morgan, L. F. Lopez, D. A. Cox, M. Stanton, I. Machado, and E. Grizendi, "Benefits brought by the use of openflow/sdn on the amlight intercontinental research and education network," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 942–947.

[7] B. Claise, "Specification of the ip flow information export (ipfix) protocol for the exchange of ip traffic flow information," 2008.

[8] A. Tootoonchian, M. Ghobadi, and Y. Ganjali, "Opentm: traffic matrix estimator for openflow networks," in *International Conference on Passive and Active Network Measurement*. Springer, 2010, pp. 201–210.

[9] S. R. Chowdhury, M. F. Bari, R. Ahmed, and R. Boutaba, "Payless: A low cost network monitoring framework for software defined networks," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*. IEEE, 2014, pp. 1–9.

[10] I. F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, "Research challenges for traffic engineering in software defined networks," *IEEE Network*, vol. 30, no. 3, pp. 52–58, 2016.

[11] Y. Zhao, L. Iannone, and M. Riguidel, "On the performance of sdn controllers: A reality check," in *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*. IEEE, 2015, pp. 79–85.

[12] Y. Yu, C. Qian, and X. Li, "Distributed and collaborative traffic monitoring in software defined networks," in *Proceedings of the third workshop on Hot topics in software defined networking*. ACM, 2014, pp. 85–90.

[13] F. Rebecchi, J. Boite, P.-A. Nardin, M. Bouet, and V. Conan, "Traffic monitoring and ddos detection using stateful sdn," in *Network Softwarization (NetSoft), 2017 IEEE Conference on*. IEEE, 2017, pp. 1–2.

[14] S. Dharmapurikar, P. Krishnamurthy, T. Sproull, and J. Lockwood, "Deep packet inspection using parallel bloom filters," in *High performance interconnects, 2003. proceedings. 11th symposium on*. IEEE, 2003, pp. 44–51.

[15] A. Rasmussen and M. S. Berger, "Towards terabit carrier ethernet and energy efficient optical transport networks," Ph.D. dissertation, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Electromagnetic SystemsInstitut for Elektromagnetiske Systemer, 2013.

[16] J. F. Zazo, S. Lopez-Buedo, G. Sutter, and J. Aracil, "Automated synthesis of fpga-based packet filters for 100 gbps network monitoring applications," in *ReConFigurable Computing and FPGAs (ReConFig), 2016 International Conference on*. IEEE, 2016, pp. 1–6.

[17] X. Wu, P. Li, Y. Ran, and Y. Luo, "Network measurement for 100 gbe network links using multicore processors," *Future Generation Computer Systems*, 2017.

[18] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, pp. 90–97, 2015.

[19] V. C. Bumgardner, *OpenStack in Action*. Manning Publications Company, 2016.

[20] G. Tsirtsis and P. Srisuresh, "Network address translation-protocol translation (nat-pt)," Tech. Rep., 2000.

[21] M. Bagnulo, P. Matthews, and I. v. Beijnum, "Stateful nat64: Network address and protocol translation from ipv6 clients to ipv4 servers," 2011.

[22] E. Borcoci, T. Ambarus, and M. Vochin, "Distributed control plane optimization in sdn-fog vanet," *ICN 2017*, p. 135, 2017.

[23] F. van Lingen, M. Yannuzzi, A. Jain, R. Irons-Mclean, O. Lluch, D. Carrera, J. L. Perez, A. Gutierrez, D. Montero, J. Marti *et al.*, "The unavoidable convergence of nfv, 5g, and fog: A model-driven approach to bridge cloud and edge," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 28–35, 2017.

[24] R. Vilalta, A. Mayoral, D. Pubill, R. Casellas, R. Martínez, J. Serra, C. Verikoukis, and R. Muñoz, "End-to-end sdn orchestration of iot services using an sdn/nfv-enabled edge node," in *Optical Fiber Communications Conference and Exhibition (OFC), 2016*. IEEE, 2016, pp. 1–3.

[25] A. Gupta, R. Birkner, M. Canini, N. Feamster, C. Mac-Stocker, and W. Wallinger, "Network Monitoring as a Stream Analytics Problem," *Proceedings of Hotnets 2016*, pp. 106–112.

[26] Y. Xu, V. Mahendran, and S. Radhakrishnan, "Towards sdn-based fog computing: Mqtt broker virtualization for effective and reliable delivery," in *Communication Systems and Networks (COMSNETS), 2016 8th International Conference on*. IEEE, 2016, pp. 1–6.

[27] R. Vilalta, I. Popescu, A. Mayoral, X. Cao, R. Casellas, N. Yoshikane, R. Martínez, T. Tsuritani, I. Morita, and R. Muñoz, "End-to-end sdn/nfv orchestration of video analytics using edge and cloud computing over programmable optical networks," in *Optical Fiber Communications Conference and Exhibition (OFC), 2017*. IEEE, 2017, pp. 1–3.

[28] V. K. Bumgardner and V. W. Marek, "Scalable hybrid stream and hadoop network analysis system," in *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*. ACM, 2014, pp. 219–224.

[29] V. C. Bumgardner, V. W. Marek, and C. D. Hickey, "Cresco: A distributed agent-based edge computing framework," in *Network and Service Management (CNSM), 2016 12th International Conference on*. IEEE, 2016, pp. 400–405.

[30] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless sensor networks for habitat monitoring," in *Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*. Acm, 2002, pp. 88–97.

[31] S. De Marchi and S. E. Page, "Agent-based models," *Annual Review of Political Science*, vol. 17, pp. 1–20, 2014.

[32] C. Hewitt, P. Bishop, and R. Steiger, "Session 8 formalisms for artificial intelligence a universal modular actor formalism for artificial intelligence," in *Advance Papers of the Conference*, vol. 3. Stanford Research Institute, 1973, p. 235.

[33] "Docker usage stats: Adoption up in the enterprise and production," October 2016. [Online]. Available: https://dzone.com/articles/docker-usage-statistics-increased-adoption-by-ente

[34] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An introduction to docker and analysis of its performance," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 17, no. 3, p. 228, 2017.

[35] S. Kuboi, K. Baba, S. Takano, and K. Murakami, "An evaluation of a complex event processing engine," in *Advanced Applied Informatics (IIAIAAI), 2014 IIAI 3rd International Conference on*. IEEE, 2014, pp. 190–193.

[36] A. Khetrapal and V. Ganesh, "Hbase and hypertable for large scale distributed storage systems," *Dept. of Computer Science, Purdue University*, pp. 22–28, 2006.

[37] C. Lonvick, "The bsd syslog protocol," 2001.

[38] W. Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*. Addison-Wesley Longman Publishing Co., Inc., 1998.