A Comparison between Text, Parquet, and PCAP Formats for Use in Distributed Network Flow Analysis on Hadoop

Miguel Zenon Nicanor L. Saavedra and William Emmanuel S. Yu

***Abstract*—Hadoop's popularity as a distributed computing platform continues to grow as more and more data is generated each year. As a fault-tolerant and horizontally scalable ecosystem, it becomes a suitable platform for the analysis of big network data. While most network data are currently being analyzed by vertically scaled machines, Hadoop provides an alternative method of analysis, allowing large datasets to be analyzed in one horizontally-scaled cluster. This study attempts to benchmark and profile the current known methods for performing network analysis on Hadoop. After comparing three data storage formats; plain text, Parquet, and raw PCAP files; for use in Hadoop, the study has determined that the Parquet and text formats greatly outperform the use of raw PCAP files using the hadoop-pcap library which fails to complete tests with high volumes of data. This comes at the expense, however, of large data loss due to the need to create a well-defined schema for processing and the conversion time necessary to shift to a different format. However, Parquet still outperforms the text format by an average of approximately 30% in the scan and aggregate queries, and 70% and 40% respectively in the join and aggregate-join queries while showing a 8%-10% increase of performance in aggregate-join queries of over 60 minutes' worth of PCAP data.**

***Index Terms*—Big data, apache hadoop, apache hive, network analysis.**

I. Introduction

Data being generated in the digital world is growing at an exponential rate. In the year 2000, an estimate of 800,000 petabytes of data was being stored by various sources [1] while 130 exabytes of data was estimated to have been stored in 2005, and over 10,000 exabytes in 2015 [2].

This is, in no small part, due to the continuous growth of IP traffic over the years. In the year 2016, there was approximately 1.2 zettabytes of traffic generated [3]; this is expected to triple by the year 2021. Large datasets of this network data also lay publicly available for deeper network research and study.

The volume of data required for network analysis and the increasing need to process these large volumes of data make this an ideal use case for horizontally scalable platforms like Hadoop [4]. Hadoop is a distributed processing framework that attempts to address the problems faced when analyzing this big data [5]. It uses a MapReduce model and its own Hadoop Distributed File System (HDFS), implemented in Java, to create a large-scale and fault-tolerant clustered data processing ecosystem [6]. While this may make it a suitable environment for the analysis of network data, there is still currently no efficient and reliable method to perform such an analysis [4], [7].

This paper seeks to first explore the current methods available for performing analysis on packet capture (PCAP) data on Hadoop as packet captures are currently one of the most popular formats for capturing network traffic [8]. For the purposes of this study, three data formats were identified for evaluation and profiling: text data in CSV format, Parquet, and raw PCAP format using the RIPE-NCC hadoop-pcap library [9] to be evaluated on Hive, Hadoop's SQL-like interface [5].

A. Research Objectives

The long-term goal of this study is to identify the best method of performing scalable network analytics on the Hadoop Distributed Ecosystem. This is done in this paper by first determining the current state of network analytics on the platform through accomplishing the following objectives:

1) To compare the performance of each of the datatypes; text, PCAP, or Parquet; after post-processing:

This study makes use of four types of queries; Scan, Aggregate, Join, and Aggregate-Join; for the performance evaluation of each of the datatypes.

2) To determine and compare the storage components of each datatype:

The storage components evaluated in the study were the space consumed by the datatype, the possible use of the storage format in other applications, and the information retained by each format.

3) To identify which datatype is best for network flow analytics:

Given the performance and storage evaluations, the main objective of this study is to determine which of the given storage types can be best used for the analysis of big network data.

B. Research Questions

To compare the given datatypes, the study seeks to answer the following research questions:

1) What are the advantages and disadvantages of using each datatype?

There may be specific trade-offs when using one datatype over the other. By weighing the advantages and disadvantages of each of the datatypes, the study aims to identify which one is best to use for analyzing network data under certain conditions.

Manuscript received August 4, 2017; revised October 9, 2017.

M. L. Saavedra is with the Department of Information Systems and Computer Science (DISCS) of the Ateneo De Manila University, Philippines (email: miguel.saavedra@obf.ateneo.edu).

W. S. Yu is with Faculty at the DISCS of the Ateneo de Manila University, Philippines (email: wyu@ateneo.edu).

doi: 10.18178/jacn.2017.5.2.241

2) How do the different datatypes perform in comparison with each other?:

The performance measure used in this study is the query response time of the Hive queries on each datatype, measured only after the pre-processing of the data.

3) How do the storage components differ between the datatypes?:

The construction and implementation of each of the datatypes have their own effect on the performance in the tests and information retention of the packets.

In Section II, the paper looks deeper into how the three datatypes have currently been used for processing on Hadoop. Section III then describes the experimental setup used in this study, with the results explained in Section IV. Section V then concludes the paper and describes the next steps that may be taken in this research.

II. Related Work

A. Text

By default, Hive automatically makes use of text-formatted tables [10]. While it is one of the most common formats for use on Hadoop, the data in the table is only partitioned by row, requiring the serialization and deserialization of each row of data whenever a query is executed [11]. This may become inefficient especially if, like in the case of network data, there are several fields in the row need to undergo this process when the query only requires a small subset of the data.

Research has determined that text may be suitable for the analysis of network data on Hadoop, especially when older routers generate network flow data in plain text format. [7]. However, these studies have also determined that a more sophisticated data format such as SequenceFiles or Parquet may be more beneficial for network analytics to yield better compression and easier parsing [7].

B. Parquet

Parquet is a columnar storage format built for the Hadoop ecosystem based off Google's Dremel system [12], [13]. It was optimized both for large-scale query processing and storage through multiple supported compression formats. By default, Parquet implements the Snappy compression format. Other supported forms of compression are: LZO, Gzip, and bizip2 [11].

There is a large amount of research done on network analysis using Parquet on Hadoop as it is deemed to be ideal for traffic analysis [14]. As a columnar-style datatype, it is partitioned both by row and by column, allowing it to skip chunks of data that are unnecessary for a particular query [11], [12].

Parquet has been heavily used for the analysis of DNS Traffic, however, a benchmark and comparison has yet to be conducted, especially in the analysis of network flow [14], [15]. Another issue with Parquet is that little research has been done to study the processing required for converting the raw PCAP data into Parquet formats. Although studies have used this conversion for their analytics, the conversion itself was never a factor in the evaluation of their applications [15].

C. PCAP

Much research has also been done in the processing of raw PCAP data. Although some studies have mentioned that the use of raw PCAP data may be too CPU intensive for processing [15], there has been much work in creating a method to use this file format because of its data retention and general completeness of network data traffic [4], [9], [16], [17]. The PCAP format, however, was never optimized for data processing and was mainly created only for logging and storing packet captures from Wireshark, TCPDump, and other network collection software and devices which pose several challenges when it is attempted to be used for analytics [8].

III. Methodology

A. Description of Data

The study makes use of the CAIDA dataset as its source of network data. The data was first collected from a one-hour bidirectional trace, and was then stripped of its payload and anonymized before being made publicly available.

For the purposes of the tests involving joins, a table of common port numbers and services was also stored in an external table in Hive in CSV format [18].

B. Cluster Configuration

A small 5-node testbed each consisting of 4Gb of memory, 500Gb of storage, and an Intel Core i5-2500 processor @ 3.3ghz running on CentOS 7 Minimal was constructed for the study. The cluster was networked using a 10/100mbps network switch and ran with Hadoop 2.6.0-cdh5.8.0, Hive version 1.1.0 and Java version 1.7.0_67.

C. Preprocessing of PCAP Data and Partitioning

To obtain the text and the Parquet data, the PCAP files were first converted and flattened into a CSV file. For the purposes of this study, only the fields related to network flow data were retained during conversion to ensure a properly defined schema for the text and Parquet datatypes. After obtaining the text data, the text was first inserted into Hive, then later converted into Parquet on another table. The conversion was done on a machine outside of the cluster running with 4Gb of memory, 1Tb of storage, and an Intel Core i5-2500 processor @ 3.3ghz.

TABLE I: Table Schema

Field	Field Type
ts	bigint
protocol	string
ts_usec	double
src	string
src_port	int
dst	string
dst_port	int
len	int
ttl	int

All tables were partitioned by their direction as given by the CAIDA dataset, and in 10-minute intervals. There was a total of two directions with 60 minutes per direction for a total of

12 partitions and 120 minutes. Table I contains the schema used for all tables utilizing the different storage types. However, it should be noted that, while this schema is used for the purposes of this study, the PCAP file format supports several other fields such as DNS, fragment, and payload information.

The RIPE-NCC hadoop-pcap library was used to process the raw PCAP data [9]. However, initial tests showed that not all PCAP files were compatible with the library due to missing information in the collected data (see Section IV for more detailed information). To allow Hive to process the dataset using the PCAP format, the incompatible files were first removed before conducting the formal testing.

D. Testing and Profiling

A total of four distinct tests were run on the dataset stored using the three different datatypes. The tests consisted of the following commonly used Hive Queries:

1) Scan queries :

The scan query was a simple select statement that identified all UDP traffic traveling from port 6699.

```
SELECT * FROM [TABLE] WHERE
src_port = 6699 AND protocol='UDP'
[AND PARTITIONS]
```

2) Aggregate queries:

The aggregate query counted the number of packets that traveled from UDP port 80, then grouped the result by the different partitions.

```
SELECT directory, minutes, COUNT(ts) FROM [TABLE]
WHERE protocol = 'UDP'
AND src_port=80 [AND PARTITIONS]
GROUP BY directory, minutes
```

3) Join queries:

The join query selected all UDP traffic. The query also joins the selected traffic with the list of common port numbers and services.

```
SELECT pcap.src, pcap.dst, pcap.src_port,
src.description, pcap.dst_port,
dst.description FROM [TABLE] pcap, ports src, ports dst
WHERE pcap.protocol = 'UDP' AND src.protocol = 'UDP'
AND dst.protocol = 'UDP' AND pcap.src_port = src.port
AND pcap.dst_port = dst.port [AND PARTITIONS]
```

4) Aggregate-join queries:

The aggregate-join query obtained the sum of all UDP traffic grouped by their source port, joined with the table of common ports, and sorted by the port with the highest count.

```
SELECT pcap.src_port, collect_set(src.description)[0],
pcap.dst_port, collect_set(dst.description)[0],
COUNT(ts) AS COUNT
FROM [TABLE] pcap, ports src, ports dst
WHERE pcap.protocol = 'UDP' AND src.protocol = 'UDP'
AND dst.protocol = 'UDP' AND src.port = pcap.src_port
AND dst.port = pcap.dst_port AND directory='dirA'
[AND PARTITIONS]
GROUP BY pcap.src_port, src.port, pcap.dst_port, dst.port
ORDER BY COUNT DESC LIMIT 10
```

Three trials of each query were done across 10-minute, 30-minute, 60-minute, 90-minute, and 120-minute timeframes of packet data. The average performance of the 20 trials per time frame were recorded as the results to reduce the possibility of other factors affecting the tests.

For consistency, each test was also run independently on the cluster with no other concurrent applications

IV. Results and Analysis

Four factors will be taken into consideration for evaluating each data format: *conversion, storage requirements, processing time, and flexibility*.

A. Conversion

Fig. 1 contains a summary of the time needed to convert the entire dataset into a given format. The PCAP-formatted data needs no initial processing for conversion because the hadoop-pcap library directly handles the data extraction from the PCAP files.

For the text and PCAP file types, there appears to be a minimal difference between the two in terms of conversion only because of the extra step needed to re-insert the data into Hive as Parquet instead of text.

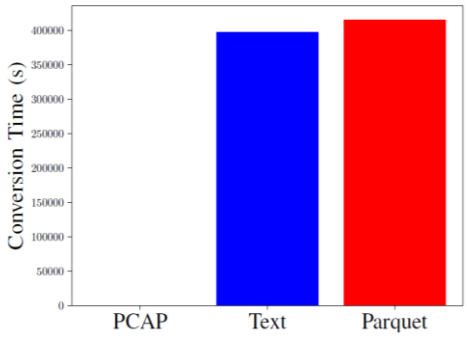

Fig. 1. Conversion time of each storage format.

B. Storage Requirements

Table II contains the storage requirements of each data type. Despite containing the most information, the PCAP files consume the least amount of space. This is because the PCAP files were originally created for the purposes of storing these types of network data, allowing it to store the most complete amount of information using the least amount of storage [8].

The parquet format is only slightly larger than the PCAP files because, by default, Parquet compresses the data for storage [5]. Despite the initial overhead needed to decompress the files for computation, the Snappy format used by Parquet has a very high compression and decompression rate of approximately 250Mb/sec and 500Mb/sec respectively [19]. This balances the need for more storage space and faster computation.

The text format needs the most amount of storage, approximately 1.6 times more than the text and Parquet formats. Text is well-known to have high entropy, becoming an inefficient structure for storing high volumes of information because of the need to encode each symbol separately [20].

TABLE II: Storage Consumed by Each Data Type (GB)

PCAP	Text	Parquet
153.3	249.1	154.8

C. Performance Testing

Table III contains an overview of the test results. For this study, tests were considered failures if they would cause all usable nodes to fail, not allowing the cluster to perform any more processing. The performance of the different storage types is first described generally per storage format, then by query.

TABLE III: SUMMARY OF RESULTS

Minutes	Storage Format	Scan	Aggregate	Join	Aggregate-Join
10	Text	326s	798s	351s	826s
	PCAP	6298s	7481s	7838s	7543s
	Parquet	266s	520s	102s	530s
30	Text	587s	2378s	988s	2309s
	PCAP	FAILED	FAILED	FAILED	FAILED
	Parquet	437s	1371s	266s	1370s
60	Text	1824s	3254s	1983s	4663s
	PCAP	FAILED	FAILED	FAILED	FAILED
	Parquet	1328s	2427s	530s	2873s
90	Text	2853s	5559s	3174s	7011s
	PCAP	FAILED	FAILED	FAILED	FAILED
	Parquet	1991s	3682s	870s	3741s
120	Text	3918s	6598s	4280s	9246s
	PCAP	FAILED	FAILED	FAILED	FAILED
	Parquet	2855s	5106s	1219s	4772s

1) Parquet:

The results are consistent for all tests with Parquet performing significantly faster than all the other data types because of its structured nature, especially during join queries. As the data grows larger and the queries become more complex, the gap between the performance times increases with Parquet performing best because of its structure which is optimized for distributed data processing on Hadoop [12] (see Fig. 2 and 3 for the comparison graphs).

2) Text:

Text performed less efficiently as the queries became more complicated. Because of less structured format as compared to Parquet, mapping text files to obtain the query result was more difficult for text compared to Parquet, making it significantly slower, especially in more complicated maps such as joins (see Fig. 3). This is also due to the fact that it is necessary for Hive to process each query on text data line by line, requiring it to read the entire text file instead of only the data needed for the query [11].

3) PCAP:

The PCAP files were by far the most inefficient for data processing taking far more time to process even small amounts of data. Several other problems also arise when using the hadoop-pcap library for processing the PCAP files directly. The general flexibility of PCAP files and network collection meant some of the packets were incomplete either due to loss during collection or pre-processing right after collection. The library then failed to process a number of PCAP files with packets that were not well-formed. The anonymization of the packets via the stripping of the payload also led the production of unnecessarily large log files that averaged 8Gb per map task, which consumed all the nodes' storage making them unusable for further tasks.

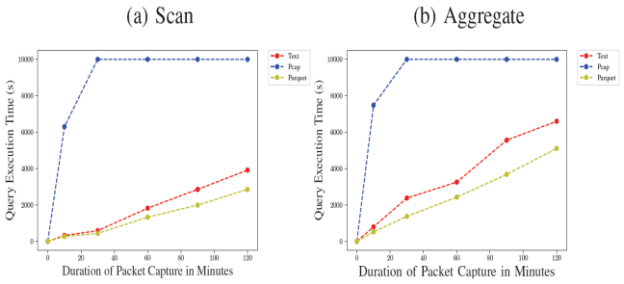

Fig. 2. Performance of scan and aggregate queries.

Fig. 2. contains the graphs of the performance of the select and aggregate queries over the time period of the packet capture. The large performance gap between the PCAP format and the text and Parquet formats show the benefits of the trade-off between conversion and processing, especially if multiple queries of that type are required. This is likely because the PCAP file format was never intended to be optimized for processing and was mainly constructed only for capturing and logging network data [8]. When comparing between the text and Parquet formats, however, there is approximately only an average performance gain of 30% for both queries which is relatively less difference than that of the latter queries (see Fig. 3). This may be due to the fact that the queries here do not use complicated operations, and since most fields are utilized, not much data is skipped from the table. These reduce the advantage of the columnar nature of Parquet.

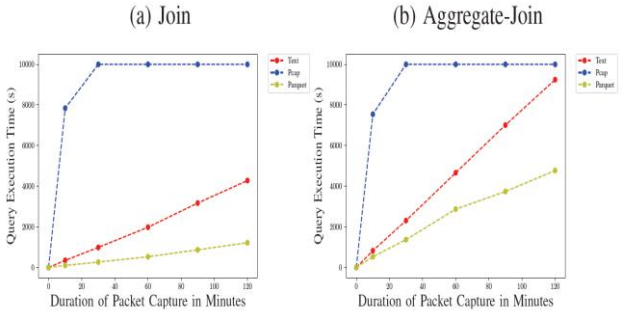

Fig. 3. Performance of join and aggregate-join queries.

Fig. 3 shows a much larger performance gain of approximately 72% for joins and 42% in aggregate-joins on average when running the analyses on Parquet-formatted tables compared to text. In both the join and the aggregate-join queries, the slope of the Parquet performance is significantly less than that of text, even showing a clear decay in processing time with a performance gain of approximately 8%-10% in the aggregate-join results when used for queries on over 60 minutes of packet capture. This is likely due to Parquet's columnar nature and its ability to skip column chunks that it does need for a specific query while processing text requires the serialization and deserialization

of each row [21], [22].

D. Flexibility

In terms of flexibility, while all three are flexible as data types, for Hadoop computation, the hadoop-pcap library is the least flexible. While it is suitable for Hive queries, the hadoop-pcap library was written with only MapReduce v1 (MRV1) support, making it difficult to port to other applications which require MRV2 [9].

Parquet, while generally only limited to the Hadoop ecosystem, can easily be used in any other application in the Hadoop ecosystem [5]. This is even more true for the text format which has compatibility that extends even beyond Hadoop.

However, because the PCAP format stores all the metadata for each packet, the conversion to text and Parquet also entailed a large amount of data loss because of the need for a well-defined structure. For this study, since the focus was on network flow, the conversion to text and Parquet focused on retaining only flow related data present in all packets. Further studies may then require a different implementation depending on their focus

V. Conclusion

This paper presents a comparison of different datatypes for big data analytics of network data in a Hadoop Ecosystem.

The results of the study show that, in terms of performance, using the hadoop-pcap library on raw PCAP data causes not only has the slowest query response time by far, but it also fails in all tests on high volumes of data. This leads to the conclusion that text or Parquet formats may be best for network flow analysis despite the factor of the long conversion time. However, using Parquet over text yields an average performance gain of 30% in scan and aggregate queries and 70% and 40% in joins and aggregate-joins respectively. As the data grows larger, Parquet is shown to perform even better, further reducing query response time by 10% when performing aggregate-join queries on over 60 minutes of PCAP data.

While Parquet clearly performs best after conversion for large datasets, using raw PCAP files may be more advisable when performing a short analysis on small, well-formed packets. Text files, on the other hand, even if they can be used in applications outside of Hadoop, the large performance gain for the short amount of conversion time indicates that it may still be best to use Parquet, especially for use in the analysis of larger network data due to its ability to run the queries on smaller subsets of data.

However, when using Parquet or text, the need for a defined schema leads to a large amount of data loss. PCAP files would then still retain the most complete information among all the datatypes while also utilizing the least amount of storage. The hadoop-pcap library, however, may not be the best way to perform analyses on this data.

New methods for utilizing PCAP data for big data analysis on the Hadoop Distributed Platform are left open for further study. Future work may be done on a better method for PCAP to Parquet conversion by utilizing Parquet's nested data structure for more complete information retention and the Hadoop's distributed environment for faster processing [14]. More work can also be done to create a more optimal PCAP Deserializer for faster processing of raw PCAP files.

Acknowledgment

The authors would like to acknowledge Dr. Ma. Regina E. Estuar and the Ateneo Social Computing Science Laboratory for providing the Hadoop cluster used in this study, and Mr. Andrei M. Jaramillo for his assistance during the cluster setup.

References

[1] D. Deroos, P. Zikopoulos, C. Eaton, T. Deutsch, G. Lapis, and P. Zikopoulos, *Understanding Big Data: Analytics for Enterprise Class Hadoop and Streaming Data*, 1st ed. McGraw-Hill Osborne Media, 2011.

[2] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC IView IDC Anal. Future*, vol. 2007, pp. 1–16, 2012.

[3] CISCO White Paper, "Cisco visual networking index: Forecast and methodology, 2016–2021," *CISCO*, 2016.

[4] Y. Lee and Y. Lee, "Toward scalable internet traffic measurement and analysis with hadoop," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 1, pp. 5–13, 2013.

[5] T. White, *Hadoop: The Definitive Guide*, O'Reilly Media, Inc., 2015.

[6] Y. S. Tan, "Hadoop framework: Impact of data organization on performance," *Softw. Pract. Exp.*, vol. 43, no. 11, pp. 1241–1260, Nov. 2013.

[7] X. Zhou, M. Petrovic, T. Eskridge, M. Carvalho, and X. Tao, "Exploring Netfow data using hadoop," in *Proc. the Second ASE International Conference on Big Data Science and Computing*, 2014.

[8] G. Harris. "Development/libpcapfileformat - The wireshark wiki," [Online]. Available: https://wiki.wireshark.org/Development/LibpcapFileFormat

[9] RIPE-NCC, *Hadoop-pcap: Hadoop Library to Read Packet Capture (PCAP) Files*. RIPE NCC, 2017.

[10] Apache Group, "LanguageManual DDL - Apache hive," *Apache Software Foundatio*n. [Online]. Available: https://cwiki.apache.org/confluence/display/Hive/LanguageManual+DDL#LanguageManualDDL-RowFormat,StorageFormat,andSerDe

[11] M. Grover, T. Malaska, J. Seidman, and G. Shapira, *Hadoop Application Architectures: Designing Real-World Big Data Applications*, 1 ed. Sebastopol, CA: O'Reilly Media, 2015.

[12] Apache Group, "Apache parquet documentation," *Apache Parquet*. [Online]. 2014. Available: http://parquet.apache.org/documentation/latest/

[13] S. Melnik, *Dremel: Interactive Analysis of Web-scale Datasets*, 2010.

[14] M. Wullink, M. Muller, M. Davids, G. C. Moura, and C. Hesselman, "ENTRADA: Enabling DNS big data applications," in *Proc. 2016 APWG Symposium on Electronic Crime Research (eCrime)*, 2016, pp. 1–11.

[15] M. Wullink, G. C. Moura, M. Müller, and C. Hesselman, "ENTRADA: A high-performance network traffic data streaming warehouse," in *Proc. Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, 2016, pp. 913–918.

[16] Y. Lee and Y. Lee, "Detecting ddos attacks with hadoop," in *Proc. of the ACM CoNEXT Student Workshop*, 2011, p. 7.

[17] J. Liu, F. Liu, and N. Ansari, "Monitoring and analyzing big traffic data of a large-scale cellular network with Hadoop," *IEEE Netw.*, vol. 28, no. 4, pp. 32–39, 2014.

[18] J. Touch. "Service name and transport protocol port number registry," *Internet Assigned Numbers Authority*. [Online]. Available: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

[19] Google, *Snappy: A Fast Compressor/Decompressor*, Google, 2017.

[20] H. Balakrishnan, C. J. Terman, and G. C. Verghese, *Bits, Signals, and Packets*.

[21] Apache Software Foundation, "Apache Parquet," [Online]. Available: https://parquet.apache.org/documentation/latest/

[22] Y. Huai, *Major Technical Advancements in Apache Hive*, 2014, pp. 1235–1246.

Miguel L. Saavedra earned his B.S in management information systems from the Ateneo de Manila University. He is currently pursuing the M.S degree in computer science in the same university under the guidance of Dr. William Yu. His research interests center around network security and cryptography, distributed computing, big data computing, and delay tolerant networking.

William S. Yu obtained his M.S and Ph.D degrees in computer science from the Ateneo de Manila University. He is currently a faculty member of the Department of Information Systems and Computer Science in the same university. He is actively into internet engineering, mobile platforms, information security research, software defined networking, and distributed systems.