A Survey on Big Data for Network Traffic Monitoring and Analysis

Alessandro D'Alconzo[‡], Idilio Drago[†], Andrea Morichetta[†], Marco Mellia[†], Pedro Casas[*]
[*] AIT Austrian Institute of Technology, [†] Politecnico di Torino, [‡] Siemens Austria

***Abstract*—Network Traffic Monitoring and Analysis (NTMA) represents a key component for network management, especially to guarantee the correct operation of large-scale networks such as the Internet. As the complexity of Internet services and the volume of traffic continue to increase, it becomes difficult to design scalable NTMA applications. Applications such as traffic classification and policing require real-time and scalable approaches. Anomaly detection and security mechanisms require to quickly identify and react to unpredictable events while processing millions of heterogeneous events. At last, the system has to collect, store, and process massive sets of historical data for post-mortem analysis. Those are precisely the challenges faced by general *big data* approaches: Volume, Velocity, Variety, and Veracity. This survey brings together NTMA and big data. We catalog previous work on NTMA that adopt big data approaches to understand to what extent the potential of big data is being explored in NTMA. This survey mainly focuses on approaches and technologies to manage the big NTMA data, additionally briefly discussing big data analytics (e.g., machine learning) for the sake of NTMA. Finally, we provide guidelines for future work, discussing lessons learned, and research directions.**

***Index Terms*—Big Data; Network Measurements; Big Data Platforms; Traffic Analysis; Machine Learning.**

I. Introduction

Understanding how Internet services are used and how they are operating is critical to people lives. Network Traffic Monitoring and Analysis (NTMA) is central to that task. Applications range from providing a view on network traffic to the detection of anomalies and unknown attacks while feeding systems responsible for usage monitoring and accounting. They collect the historical data needed to support traffic engineering and troubleshooting, helping to plan the network evolution and identify the root cause of problems. It is correct to say that NTMA applications are a cornerstone to guarantee that the services supporting our daily lives are always available and operating as expected.

Traffic monitoring and analysis is a complicated task. The massive traffic volumes, the speed of transmission systems, the natural evolution of services and attacks, and the variety of data sources and methods to acquire measurements are just some of the challenges faced by NTMA applications. As the complexity of the network continues to increase, more observation points become available to researchers, potentially allowing heterogeneous data to be collected and evaluated. This trend makes it hard to design scalable and distributed applications and calls for efficient mechanisms for online analysis of large streams of measurements. More than that, as storage prices decrease, it becomes possible to create massive historical datasets for retrospective analysis.

These challenges are precisely the characteristics associated with what, more recently, have become known as *big data*, i.e., situations in which the data *volume*, *velocity*, *veracity* and *variety* are the key challenges to allow the extraction of *value* from the data. Indeed, traffic monitoring and analysis were one of the first examples of big data sources to emerge, and it poses big data challenges more than ever.

It is thus not a surprise that researchers are resorting to big data technologies to support NTMA applications (e.g., [69], [77], [87], [124]). Distributed file systems – e.g., the Hadoop[1] Distributed File System (HDFS), big data platforms – e.g., Hadoop and Spark, and distributed machine learning and graph processing engines – e.g., MLlib and Apache Giraph, are some examples of technologies that are assisting applications to handle datasets that otherwise would be intractable. However, it is by no means clear whether NTMA applications fully exploit the potential of emerging big data technologies.

We bring together NTMA research and big data. Whereas previous works documented advances on big data research and technologies [34], [116], [126], methods supporting NTMA (e.g., machine learning for NTMA [17], [85]), or specific NTMA applications [12], [52], [105], [120], there is a lack of systematic surveys describing how NTMA and big data are being combined to exploit the potential of network data fully.

More concretely, the goal of this survey is to discuss *to what extent NTMA researchers are exploiting the potential of big data technologies*. We aim at providing network researchers willing to adopt big data approaches for NTMA applications a broad overview of success cases and pitfalls on previous research efforts, thus illustrating the challenges and opportunities on the use of big data technologies for NTMA applications.

By summarizing recent literature on NTMA, we provide researchers principled guidelines on the use of big data according to the requirements and purposes of NTMA applications. Ultimately, by cataloging how challenges on NTMA have been faced with big data approaches, we highlight open issues and promising research directions.

The research leading to these results has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-129 "BigDAMA", with the contribution from the SmartData@PoliTO center for data science and big data. Authors sincerely thank Prof. Tanja Zseby, Dr. Felix Iglesias, and Daniel C. Ferreira from TU Wien for their contributions to the work and early versions of the manuscript.

[1]http://hadoop.apache.org/

A. Survey methodology

We first identify papers summarizing big data research. We have explored the literature for big data surveys, restricting our focus to the last ten years. This literature has served as a basis to our definition for big data as well as to limit our scope in terms of the considered big data technologies.

Since none of these papers addresses big data for NTMA, we have followed a similar methodology and reviewed papers in the NTMA domain. We survey how NTMA researchers are profiting from big data approaches and technologies. We have focused on the last 5 years of (i) the main system and networking conferences, namely SIGCOMM, NSDI, CONEXT, and journals (IEEE TON, IEEE TNSM), (ii) new venues targeting analytics and big data for NTMA, namely the BigDama, AnNet, WAIN and NetAI workshops, and their parent conferences (e.g., IM, NOMS and CNSM); (ii) special issues on Big Data Analytics published in this journal (TNSM). We complement the survey by searching on Google Scholar, IEEE Explorer, and ACM Digital library. To ensure relevance and limit our scope, we select papers concerning publication venues, the number of citations, and scope.

The survey is organized as follows: Sect. II introduces concepts of big data and the steps for big data analytics, giving also some background in big data platforms. Sect. III reviews a taxonomy of NTMA applications, illustrating how NTMA relates to the big data challenges. Subsequent sections detail the process of NTMA and discuss how previous work has faced big data challenges in NTMA. Sect. IV focuses on data capture, ingestion, storage and pre-processing, whereas Sect. V overviews big data analytics for NTMA. Finally, Sect. VI describes lessons learned and open issues.

II. What is big data?

A. Definition

Many definitions for big data have appeared in the literature. We rely on previous surveys that focused on other aspects of big data but have already documented its definitions.

Hu at al. [54] argue that "big data means not only a large volume of data but also other features, such as variety and velocity of the data". They organize definitions in three categories: architectural, comparative, or attributive.

Architectural and comparative definitions are both abstract. Following an architectural definition, one would face a big data problem whenever the dataset is such that "traditional approaches" are incapable of storing and handling it. Similarly, comparative definitions characterize big data by comparing the properties of the data to the capabilities of traditional database tools. Those relying on *attributive* definitions (e.g., [65], [76]) describe big data by the salient features of both the data and the process of analyzing the data itself. They advocate what is nowadays widely known by the big data *"V's"* – e.g., volume, velocity, variety, veracity, value.

Other surveys targeting big data technologies [6], [54], [126] and analytics [34], [116] share this view. We will stick to the "5-Vs" definition because it provides concrete criteria that characterize the big data challenges. We thus consider a problem to belong to the big data class if datasets are large (i.e., volume) and need to be captured and analyzed at high rates or in real-time (i.e., velocity). The data potentially come from different sources (i.e., variety) that combine (i) structured data such as column-oriented databases; (ii) unstructured data such as server logs; and (iii) semi-structured data such as XML or JSON documents. Moreover, data can be of different quality, with the uncertainty that characterizes the data (i.e., veracity). At last, the analysis of the data brings advantages for users and businesses (i.e., value). That is, new insights are extracted from the original data to increase its value, preferably using automatic methodologies.

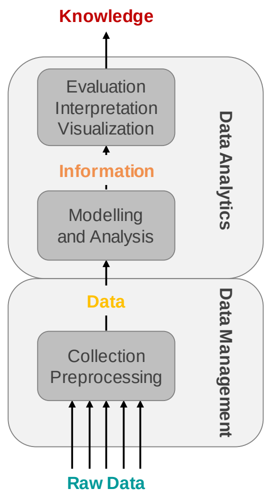

Figure 1: Knowledge Discovery in Data (KDD) [35], [116].

We argue next that NTMA shares these characteristics. Hence, NTMA is an example of big data application which can profit from methodologies developed to face these challenges.

B. Knowledge discovery in big data

The process of KDD is often attributed to Fayyad et al. [35] who summarized it as a sequence of operations over the data: gathering, selection, pre-processing, transformation, mining, evaluation, interpretation, and visualization. From a system perspective, authors of [116] reorganized these operations in three groups: input, analysis, and output.

Data input performs the data management process, from the collection of raw data to the delivery of data in suitable formats for subsequent mining. It includes pre-processing, which are the initial steps to prepare the data, with the integration of heterogeneous sources and cleaning of spurious data.

Data analysis methods receive the prepared data and extract information, i.e., models for classification, hidden patterns, relations, rules, etc. The methods range from statistical modeling and analysis to machine learning and data mining algorithms.

Data output completes the process by converting information into knowledge. It includes steps to measure the information quality, to display information in succinct formats, and to assist analysts with the interpretation of results. This step is not evaluated in this survey and is ignored in the following.

The stages identified by Fayyad et al. can be further grouped into data management and analytics (i.e., data analysis and

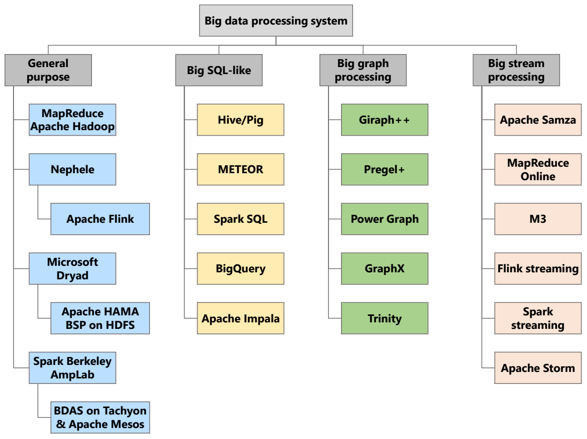

Figure 2: Processing systems (based on [6], [97], [126]).

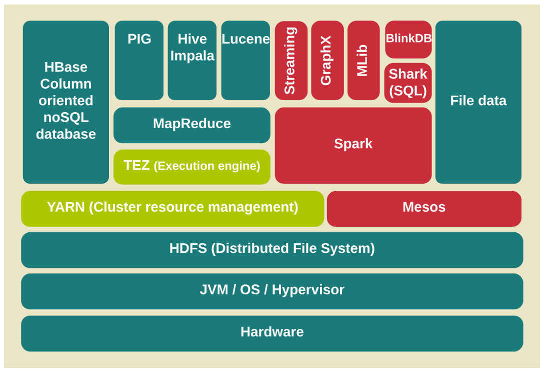

Figure 3: Hadoop 2.0 stack with Spark (based on [14]).

output). Note that differently from [116], we use the term *data analysis* to refer to the algorithms for extracting information from the data, whereas we use *data analytics* to refer to the whole process, from data analysis to knowledge discovery.

We reproduce and adapt this scheme in Fig. 1 and will use it to characterize NTMA papers according to the several stages of the KDD process.

The KDD process described above applies to any data analytics. However, the characteristics of the big data impose fundamental challenges to the methodologies on each step. For example, the data management process will naturally face much more complex tasks with big data, given the volume, velocity, and variety of the data. Data management is, therefore, crucial with big data since it plays a key role in reducing data volume and complexity. Also, it impacts the analysis and the output phases, in particular in terms of speed of the analysis and value of results.

The big data challenges (i.e., the "5-Vs") call for an approach that considers the whole KDD process in a comprehensive analytics framework. Such a framework should include programming models that allow implementing application logic covering the complete KDD cycle. We will show later that a common practice to cope with big datasets is to resort to parallel and distributed computing frameworks that are still on expansion to cover all KDD phases.

C. Programming models and platforms

We provide a short overview of the most relevant programming models and platforms for handling big data. While surveying the literature, we will give particular emphasis on works that make use of such models and platforms for NTMA.

Following the taxonomies found in [6], [126], we distinguish four types of big data processing models: (i) general purpose – platforms to process big data that make little assumptions about the data characteristics and the executed algorithms, (ii) SQL-like – platforms focusing on scalable processing of structured and tabular data, (iii) graph processing – platforms focusing on the processing of large graphs, and (iv) stream processing – platforms dealing large-scale data that continuously arrive to the system in a streaming fashion.

Fig. 2 depicts this taxonomy with examples of systems. MapReduce, Dryad, Flink, and Spark belong to the first type. Hive, HAWQ, Apache Drill, and Tajo belong to the SQL-like type. Pregel, GraphLab follow the graph processing models and, finally, Storm and S4 are examples of the latter.

A comprehensive review of big data programming models and platforms is far beyond the scope of this paper. We refer readers to [6], [126] for a complete survey.

1) The Hadoop ecosystem: Hadoop is the most widespread solution among the general-purpose big data platforms. Given its importance, we provide some details about its components in Fig. 3, considering Hadoop v2.0 and Spark [14].

Hadoop v2.0 consists of the Hadoop kernel, MapReduce and the Hadoop Distributed File System (HDFS). YARN is the default resource manager, providing access to cluster resources to several competing jobs. Other resource managers (e.g., Mesos) and execution engines (e.g., TEZ) can be used too, e.g., for providing resources to Spark.

Spark has been introduced in Hadoop v2.0 onward aiming to solve limitations in the MapReduce paradigm. Spark is based on data representations that can be transformed into multiple steps while efficiently residing in memory. In contrast, the MapReduce paradigm relies on basic operations (i.e., map/reduce) that are applied to data batches read and stored to disk. Spark has gained momentum in non-batch scenarios, e.g, iterative and real-time big data applications, as well as in batch applications that cannot be solved in a few stages of map/reduce operations.

Several high-level language and systems, such as Google's Sawzall [92], Yahoo's Pig Latin [45], Facebook's Hive [110], and Microsoft's SCOPE [21] have been proposed to run on top of Hadoop. Moreover, several libraries such as Mahout [89] over MapReduce and MLlib over Spark have been introduced by the community to solve problems or fill gaps in the original Hadoop ecosystem. Finally, the ecosystem has been complemented with tools targeting specific processing models, such as GraphX and Spark Streaming, which support graph and stream processing, respectively.

Besides the Apache Hadoop distributions, proprietary platforms offer different features for data processing and cluster management. Some of such solutions include Cloudera CDH,[2] Hortonworks HDP,[3] and MapR Converged Data Platform.[4]

[2]http://www.cloudera.com/downloads/cdh/5-8-2.html
[3]http://hortonworks.com/products/data-center/hdp/
[4]https://www.mapr.com/products/mapr-converged-data-platform

Table I: Categories of NTMA applications [15].

Category	Description
Traffic prediction	Maintenance and planning of networks. Historically achieved through time series forecasting.
Traffic classification	Categorize and recognize traffic flows for different objectives, e.g., QoE, security etc.
Fault management	Predict and isolate faults and unwanted behaviors in networks.
Network security	Protect the network, react to (and prevent from) malicious activities and general attacks.

III. Categorizing NTMA applications

A. Taxonomy

We rely on a subset of the taxonomy of network management applications found in [15] to categorize papers handling NTMA with big data approaches. The previous work lists eight categories, which are defined according to the final purpose of the management application, namely: (i) traffic prediction, (ii) traffic classification, (iii) fault management, (iv) network security, (v) congestion control, (vi) traffic routing, (vii) resource management, and (viii) QoS/QoE management.

We only survey works that fit on the first four categories for two reasons. First, whereas the taxonomy in [15] is appropriate for describing network management applications, the level of dependence of such applications on NTMA varies considerably. Traffic routing and resource management seem less dependent on large-scale measurements than traffic prediction and security, for example. Second, the literature on the use of big data approaches for congestion control, traffic routing, resource management, and QoS/QoE management is almost nonexistent by the time of surveying. We conjecture that either large-scale datasets targeting problems in these categories are not available, or researchers have not identified potential on applying big data approaches in those scenarios. We thus ignore the works using big data approaches for those cases.

B. NTMA applications

Tab. I shows the categories used in our survey. Next, we list examples of NTMA applications in these categories.

1) Traffic prediction: Traffic prediction consists of estimating the future status of network links. It serves as a building block for traffic engineering, helping to define, for example, the best moment to deploy more capacity in the network so to keep QoS levels.

Traffic prediction is often faced as a time-series forecasting problem [15]. Here both classic forecasting methods (e.g., ARIMA or SARIMA methods) and machine learning (e.g., deep neural networks) are employed. The problem is usually formulated as the estimation of traffic volumes based on previous measurements in the same links. Changes in network behavior, however, make such estimations very complicated. New services or sudden changes in service configurations (e.g., the deployment of novel bandwidth-hungry applications) poses major challenges to traffic prediction approaches.

2) Traffic classification: Traffic classification aims at identifying services producing traffic [120]. It is a crucial step for managing and monitoring the network. Operators need information about services, e.g., to understand their requirements and their impact on the overall network performance.

Traffic classification used to work well by simply inspecting information in network and transport protocols. For instance, Internet services used to be identified by simply inspecting TCP/UDP port numbers. However, traffic classification is no longer a simple task [85]. First, the number of Internet services is large and continues to increase. Second, services must be identified by observing little information seen in the network. Third, little information remains visible in packets, since a major share of the Internet services run on top of a handful of encryption protocols (e.g., HTTPS over TCP). At last, Internet services are dynamic and constantly updated.

Many approaches have been proposed to perform traffic classification. They can be grouped according to the strategy used to classify the packets: (i) Packet inspection analyzes the content of packets searching for pre-defined messages [16] or protocol *fingerprints*; (ii) Supervised machine learning methods extract features from the traffic and, in a training phase, build models to associate feature values to the services; (iii) Unsupervised machine learning methods cluster traffic without previous knowledge on the services. As such, they are appropriate to explore services for which training data is unavailable [33]. (iv) Behavioral methods identify services based on the behavior of end-nodes [60]. The algorithms observe traffic to build models for nodes running particular services. The models describe, for instance, which servers are reached by the clients, with which data rate, the order in which servers are contacted, etc.

3) Fault management: Fault management is the set of tasks to predict, detect, isolate, and correct faults in networks. The goal is to minimize downtime. Fault management can be proactive, e.g., when analytics predict faults based on measurements to avoid disruptions, or reactive, e.g., when traffic and system logs are evaluated to understand ongoing problems. In either case, a key step in fault management is the localization of the root-cause of problems [15].

In large networks, diverse elements may be impacted by faults: e.g., a failed router may overload other routes, thus producing a chain of faults in the network. Diverse network elements will produce system logs related to the problem, and the behavior of the network may be changed in diverse aspects. Detecting malfunctioning is often achieved by means of anomaly detection methods that identify abnormal behavior in traffic or unusual events in system logs. Anomalies, however, can be caused also by security incidents (described next) or normal changes in usage patterns. Analytics algorithms often operate evaluating traffic, system logs, and active measurements in conjunction, so to increase the visibility of the network and easy the identification of root-causes of problems.

4) Security: Many NTMA applications have been proposed for assisting cyber-security [74]. The most common objective is to detect security flaws, virus, and malware, so to isolate infected machines and take countermeasures to minimize damages. Roughly speaking, there are two main approaches when searching for malicious network activity: (i) based on attack signatures; (ii) based on anomaly detection.

Signature-based methods build upon the idea that it is possible to define fingerprints for attacks. A monitoring solution inspects the source traffic/logs/events searching for (i) known messages exchanged by viruses, malware or other threats; or (ii) the typical communication patterns of the attacks – i.e., similar to behavioral traffic classification methods. Signature-based methods are efficient to block well-known attacks that are immutable or that mutate slowly. These methods however require attacks to be well-documented.

Methods based on anomaly detection [12], [44] build upon the assumption that attacks will change the behavior of the network. They build models to summarize the *normal* network behavior from measurements. Live traffic is then monitored and alerts are triggered when the behavior of the network differs from the baseline. Anomaly detection methods are attractive since they allow the early detection of unknown threats (e.g., zero-day exploits). These methods, however, may not detect stealth attacks (i.e., false negatives), which are not sufficiently large to disturb the network. They sometimes suffer from large numbers of false positives too.

C. Big data challenges in NTMA applications

We argue that NTMA applications belonging to the categories above can profit from big data approaches. Processing such measurements poses the typical big data challenges (i.e., the "5-Vs"). We provide some examples to support the claim.

Considering *volume* and *velocity* and taking traffic classification as an example: it has to be performed on-the-fly (e.g., on packets), and the input data are usually large. We will see later that researchers rely on different strategies to perform traffic classification on high-speed links, with some works applying algorithms to hundreds of Gbps streams.

Consider then *variety*. As more and more traffic goes encrypted, algorithms to perform traffic classification or fault management, for example, have poorer information to operate. Modern algorithms rely on diverse sources – e.g., see [113] that combines DNS and flow measurements. Anomaly detection, as another example, is usually applied to a variety of sources too (e.g., traffic traces, routing information, etc), so to obtain diverse signals of anomalies.

In terms of *veracity*, we again cite cyber-security. Samples of attacks are needed to train classifiers to identify the attacks on real networks. Producing such samples is a challenging task. While simple attacks can be reproduced in laboratory, elaborate attacks are hard to reproduce – or, worst, are simulated in unrealistically ways – thus limiting the analysis.

Finally, *value* is clearly critical in all the above applications – e.g., in cyber-security, a single attack that goes undetected can be unbearable to the operation of the network.

IV. Data management for NTMA

We now survey how the data management steps (cf. Fig. 1) are performed in NTMA applications. Big data analytics for NTMA are instead covered in Sect. V.

A. Data collection

Measuring the Internet is a big data challenge. There are more than half a million networks, 1.6 billion websites, and more than 3 billion users, accessing more than 50 billion web pages, i.e., exchanging some zettabytes per year. At no surprise, the community has spent much work in designing and engineering tools for data acquisition at high-speeds; some of them are discussed in [32]. Here the main challenges addressed by the community seem to be the scalability of data collection systems and how to reduce data volumes at the collection points without impacting the data quality.

Measuring the Internet can be accomplished coarsely in two means: (i) active measurements, and (ii) passive measurements. The former indicates the process of injecting data into the network and observing the responses. It is a not scalable process, typically used for troubleshooting. Passive measurements, on the contrary, build on the continuous observation of traffic, and the extraction of indexes in real-time.

A network *exporter* captures the traffic crossing monitoring points, e.g., routers aggregating several customers. The exporter sends out a copy of the observed network packets and/or traffic statistics to a *collector*. Data is then saved in *storage* formats suitable for NTMA applications. *Analysis* applications then access the data to extract knowledge.

The components of this architecture can be integrated into a single device (e.g., a router or a dedicated network monitor) or deployed in a distributed architecture. We will argue later that big data already emerges since the first stage of the NTMA process and, as such, distributed architectures are common in practical setups. The large data rate in the monitoring points has pushed researchers and practitioners into the integration of many pre-processing functionalities directly into the exporters. The most prominent example is perhaps flow-based monitoring, in which only a summary of each traffic flow is exported to collectors. Next, we provide a summary of packet- and flow-based methods used to collect data for NTMA applications.

1) Packet-based methods: Analyzing packets provides the highest flexibility for the NTMA applications, but also requires a significant amount of resources. Deep Packet Inspection (DPI) means to look into, and sometimes export, full packet contents that include application headers and payload. The data rate poses challenges (i.e., velocity), which can be faced using off-the-shelf hardware [115], provided hardware support is present [58]. Indeed, there exist technologies to perform DPI at multiple Gbit/s, while also saving the packets to disk [31].

The network monitoring community has proposed multiple alternatives to avoid the bottlenecks of packet-based methods. The classical approach is to perform pre-processing near the data collection. Filtering and sampling are usually set on collection points to export only a (small) subset of packets that pass the filtering and sampling rules. Other ad-hoc transformations may be employed too, e.g., the exporting of packet headers only, instead of full packet contents. In a similar direction, authors of [75] propose to collect only the initial packets of each flow, since these packets are usually sufficient for applications such as traffic classification.

As an illustration, Tab. II describes the volume of packet traces captured in some real deployments. The first two lines report the size of packet traces capture at (i) an ISP network where around 20 k ADSL customers are connected; (ii) a campus network connecting around 15 k users. Both captures have been performed in morning hours and last for at least 1

Table II: Size of real packet traces captured at different vantage points with different methodologies.

Description	Duration	pcap (GB)	Headers up to L4 (%)
Full packets of 20 k ADSL users (morning)	60 mins	675	6.8
Full packets of 15 k Campus users (morning)	100 mins	913	6.3
Only 5 kB or 5 packets per flow, 20 k ADSL users	1 day	677	22.5

Table III: Estimated measurement streams for a flow exporter observing 50 Gbit/s on average (source [32], [52]).

Sampling rate	**Protocol**	Packets/s	Bits/s
1:1	NetFlow v5	4.1 k	52 M
1:1	NetFlow v9	10.2 k	62 M
1:10		4.7 k	27 M
1:1	IPFIX	12.5 k	75 M

hour. More than half of TB is saved in both cases. The table also shows that the strategy of saving only packet headers up to the transport layer only partially helps to reduce the data volume – e.g., around 45 GB of headers per hour would still be saved for the ISP trace. Finally, the last line shows the size of a full day of capture in the ISP network with a setup similar to [75] (i.e., saving only 5 packets or 5 kB per flow). Whereas the data volume is reduced significantly, more than 600 GB of pcaps are saved per day.

Nowadays, DPI is challenged by encryption as well as by restrictive privacy policies [40]. Alternatives to performing DPI on encrypted traffic have been presented [103].

2) Flow-based methods: Flow-based methods process packets near the data collection, exporting only summaries of the traffic *per flow* [52]. A network flow is defined as a sequence of packets that share common characteristics, identified by a *key*. NTMA applications that analyze flow records have lower transmission requirements since data is aggregated. Data privacy is better protected and issues related to encryption are partially avoided. Nevertheless, there is an unavoidable loss of information, which makes the analysis more limited. In 2013, Cisco estimated that approximately 90% of network traffic analyses are flow-based, leaving the remaining 10% for packet-based approaches [90].

Diverse flow monitoring technologies have been proposed. Cisco NetFlow, sFlow and IPFIX are the most common ones. NetFlow has been widely used for a variety of NTMA applications, for example for network security [86]. sFlow relies heavily on packet sampling (e.g., sampling 1 every 1 000 packets). This may hurt the performance of many NTMA applications, even if sFlow is still suitable for detecting attacks for example. Finally, IPFIX (IP Flow Information Export protocol) is the IETF standard for exporting flow information [52]. IPFIX is flexible and customizable, allowing one to select the fields to be exported. IPFIX is used for a number of NTMA applications, such as the detection of SSH attacks [53].

Even flow-based monitoring may produce very large datasets [32]. For illustration, Tab. III lists the volume of flow-level information exported when monitoring 50 Gbit/s on average. The table is built by extrapolating volumes reported by [52] for flow exporters in a real deployment. The table includes Cisco's NetFlow v5 and NetFlow v9 as well as IPFIX. Flow-based NTMA applications would still face more than 70 Mbit/s if IPFIX is chosen to export data. Since these numbers refer to a single vantage point, NTMA applications that aggregate multiple vantage points may need to face several Gb/s of input data. Finally, the table reports data speeds when employing sampling (see 1:10 rate). Sampling reduces the data volume considerably but limiting NTMA applications.

At last, NTMA applications often need to handle historical data. The storage needed to archive the compressed flow data from the vantage point used as an example in Tab. II grows linearly over time, and the archival consumes almost 30 TB after four years [32] of archival.

B. Data ingestion

The previous step is the KDD step that depends the most on the problem domain since the way data is acquired varies a lot according to the given scenario. Not a surprise, generic big data frameworks mostly provide tools and methods for *transporting* raw data from its original sources into pre-processing and analytics pipelines. The process of transporting the data into the frameworks is usually called *data ingestion*. Since data at its source is still unprocessed, handling such raw data may be a huge challenge. Indeed, large data streams or a high number of distributed collectors will produce a deluge of data to be parsed and pre-processed in subsequent stages.

Several generic tools for handling big data ingestion can be cited: (i) *Flume*,[5] a distributed system to collect logs from different sources; (ii) *Sqoop*,[6] which allows to transmit data between Hadoop and relational databases; (iii) *Kafka*,[7] a platform that provides a distributed publish-subscribe messaging system for online data streaming; among others. These tools focus on scalability – e.g., they allow multiple streams to be processed in a distributed fashion. Once data is delivered to the framework, pre-processing can take place on the streams.

Considering NTMA, few solutions have been proposed to ingest traffic measurements into big data frameworks. Here the main challenges seem to be the transport of data in different formats and from various sources into the frameworks. The research community has made interesting progresses in this front, and we cite two examples.

Apache Spot[8] is an open platform that integrates modules for different applications in the network security area. It relies on Kafka for performing data ingestion. Users and developers have to provide Spot with python scripts that parse the original measurements into a pre-defined, but flexible, format. Both the original data and converted records are loaded and stored in the big data framework.

[5]https://flume.apache.org
[6]http://sqoop.apache.org
[7]https://kafka.apache.org
[8]http://spot.incubator.apache.org

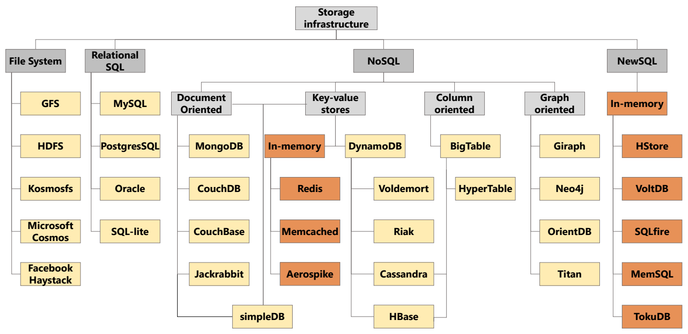

Figure 4: Generic big storage systems (based on [14], [24]).

A promising project is PNDA (Platform for Network Data Analysis).[9] Developed by the Linux Foundation, it is an open-source platform capable of managing and grouping together different data sources. It then performs analysis on such data sources. It does not force data in any specific schema and it allows the integration of other sources, producing custom code for the analysis stage. It makes use of standard tools in big data frameworks, including Kafka for data ingestion. Here, a number of plugins for reading data in virtually all popular NTMA formats is available on the project website.

Spot and PNDA experiences clearly show that ingesting NTMA data into generic frameworks requires some effort to plugin the NTMA formats into the frameworks. Once such plugins exist, standard ingestion tools like Kafka can be employed. The availability of open source code (e.g., PNDA plugins) makes it easier to integrate these functionalities in other NTMA applications.

C. Data storage

In theory, generic store systems of big data frameworks can be used with NTMA too. In practice, a number of characteristics of the big data frameworks complicate the storage of NTMA data. In order to ease the understanding of these challenges, we first provide some background in generic big data storage systems. Then, we evaluate how the NTMA community is employing these systems in NTMA applications.

1) Generic systems: Fig. 4 reproduces a taxonomy of big data storage and management systems [14], [24]. Colors represent the media (i.e., memory or persistent media) primarily exploited by the system.

Most big data storage systems focus on horizontal scalability, i.e., growing the capacity of the system across multiple servers, rather than upgrading a single server to handle increasing data volumes. This approach results in large distributed systems, which carry many risks, such as node crashes and network failures. Consistency, availability, and fault tolerance are therefore of large concern in big data storage [14], [54].

In terms of file systems, Google has pioneered the development by the implementation of the Google File System (GFS) [46]. GFS was built with horizontal scalability in mind, thus running on commodity servers and providing fault tolerance and high performance. Colossus [79], the successor of GFS and Facebook Haystack [8] are other examples. Open source derivatives of GFS appeared later, including Apache HDFS and Kosmosfs.

On a higher abstraction level, relational databases suffer performance penalties with large datasets. NoSQL databases [49] are alternatives that emerged for such scenarios. NoSQL databases scale better than the relational ones by overcoming the intrinsic rigidity of database schemas. NoSQL databases adopt different physical data layouts: (i) *Key-value* databases store data in sets of key-value pairs organized into rows. Examples include Amazon's Dynamo [30] and Memcached;[10] (ii) *column-oriented* databases (inspired on Google's BigTable [23]) store data by column instead of by row. Examples are Cassandra [64] and HBase;[11] (iii) *document oriented* databases store data in documents uniquely identified by a key. An example is MongoDB;[12] (iv) *Graph oriented* databases store graphs – i.e., nodes and edges. They impose a graph schema to the database, but profiting from it to implement efficient operations in graphs. An example is Titan.[13]

Finally, NewSQL systems aim at providing a similar performance of NoSQL databases, while maintaining properties of relational systems [107], e.g., relational data models and SQL queries [19]. An example is Google Spanner.[14]

2) NTMA storage in big data frameworks: Being the Internet a distributed system, a key problem is how to archive measurements in a centralized data store. Here no standard solution exists, despite multiple attempts to provide scalable and flexible approaches [96], [112]. The measurements are usually collected using ad-hoc platforms and exported in formats that are not directly readable by big data frameworks. Therefore, both special storage solutions and/or additional data transformation steps are needed.

For example, `libpcap`, `libtrace`, `libcoral`, `libnetdude` and `Scapy` are some libraries used for capturing packets. These libraries read and save traces using the `pcap` format (or `pcap-ng`). Distributed big data file and database systems, such as HDFS or Cassandra, are generally unable to read `pcap` traces directly, or in parallel, since the `pcap` traces are not separable – i.e., they cannot be easily split into parts. One would still need to reprocess the traces sequentially when reading data from the distributed file systems. A similar problem emerges for formats used to store flow-based measurements. For example, `nfdump` is a popular format used to store NetFlow measurements. While files are split into parts by the collector according to pre-set parameters (e.g., saving one file every five minutes), every single file is a large binary object. Hadoop-based systems cannot split such files into parts automatically.

A number of previous works propose new approaches to overcome these limitations: (i) loading the measurements in original format to the distributed system, while extending the frameworks to handle the classic formats; (ii) proposing new

[9]http://pnda.io/overview

[10]http://memcached.org
[11]https://hbase.apache.org/
[12]https://www.mongodb.com/
[13]http://titan.thinkaurelius.com/
[14]https://cloud.google.com/spanner/

storage formats and tools for handling the big network data; (iii) transforming the network data and importing it into conventional big data storage systems (discussed in Sect. IV-D).

Lee and Lee [69] have developed a Hadoop API called *PcapInputFormat* that can manage IP packets and NetFlow records in their native formats. The API allows NTMA applications based on Hadoop to seamlessly process `pcap` or NetFlow traces. It thus allows traces to be processed without any previous transformation while avoiding performance penalties of reading files sequentially in the framework. A similar direction is followed by Ibrahim et al. [56], who develop traffic analysis algorithms based on MapReduce and a new input API to explore `pcap` files on Hadoop.

In [84], Nagele faces the analysis of `pcap` files in a fast and scalable way by implementing a java-based hadoop-pcap library. The project includes a Serializer/Deserializer that allows Hive to query `pcaps` directly. Authors of [109] use the same Serializer/Deserializer to build a Hadoop-based platform for network security that relies on sFlow, Netflow, DNS measurements and SPAM email captures.

Noting a similar gap for processing BGP measurements, authors of [87] introduce BGPStream. BGPStream provides tools and libraries that connect live BGP data sources to APIs and applications. BGPStream can be used, for instance, for efficiently ingesting on-line data into stream processing solutions, such as Spark Streaming.

All these APIs are however specific, e.g., to HDFS, `pcap` or NetFlow. Thus, similar work has to be performed for each considered measurement format or analytics framework.

Some authors have taken a distinct approach, proposing new storage formats and solutions more friendly to the big network measurements. Authors of [7] propose DBStream to calculate continuous and rolling analytics from data streams. Other authors have proposed to extend `pcap` and flow solutions both to achieve higher storage efficiency (e.g., compressing traces) and to include mechanisms for indexing and retrieving packets efficiently [41], [42]. These solutions are all built upon key ideas of big data frameworks, as well as key-value or column-oriented databases. They are however specialized to solve network monitoring problems. Yet, the systems are generally centralized, thus lacking horizontal scalability.

D. Pre-processing

NTMA algorithms usually operate with *feature vectors* that describe the instances under study – e.g., network flows, contacted services, etc. We, therefore, consider as pre-processing all steps to convert raw data into feature vectors.

Some papers overcome the lack of storage formats by transforming the data when ingesting it into big data frameworks. Similarly, a set of features must be extracted for performing NTMA, both for packet and flow-based analysis. Next, we review works performing such pre-processing tasks for NTMA.

1) Transformations: A popular approach to handle the big network measurements is to perform further transformations after the data leaves the collection point. Either raw `pcap` or packets pre-processed at the collection point (e.g., sampled or filtered) are passed through a series of transformations before being loaded into big data frameworks.

Authors of [124] use an extra pre-processing step to convert measurements from the original format (i.e., `pcap` in HDFS) into a query-optimized format (i.e., Parquet). This conversion is shown to bring massive improvements in performance, but only pre-selected fields are loaded into the query-optimized format. Authors of [98] propose a Hadoop-based framework for network analysis that first imports data originally in perfSONAR format into Apache Avro. Authors of [67] process flow logs in MapReduce, but transforming the original binary files into text files before loading them into HDFS.

Marchal et al. [77] propose “a big data architecture for large-scale security monitoring” that uses DNS measurements, NetFlow records and data collected at honeypots. Authors take a hybrid approach: they load some measurements into Cassandra while also deploying Hadoop APIs to read binary measurement formats directly (e.g., `pcaps`). The performance of the architecture is tested with diverse big data frameworks, such as Hadoop and Spark. Similarly, Spark Streaming is used to monitor large volumes of IPFIX data in [20], with the IPFIX collector passing data directly to Spark in JSON format.

In [70] sFlow records of a large campus network are collected and analyzed on a Hadoop system in order to classify host behaviors based on machine learning algorithms. sFlow data are collected, fields of interest are extracted and then ingested into Cassandra using Apache Flume. Sarlis et al. propose a system for network analytics based on sFlow and NetFlow (over Hadoop or Spark) that achieves 70% speedup compared to basic analytics implementations with Hive or Shark [100]. Measurements are first transformed into optimized partitions that are loaded into HDFS and HBASE together with indexes that help to speed up the queries.

An IPFIX-based lightweight methodology for traffic classification is developed in [83]. It uses unsupervised learning for word embedding on Apache Spark, receiving as input “decorated flow summaries”, which are textual flow summaries augmented with information from DNS and DHCP logs. Finally, specifically for big data scenarios, Cisco has published a comprehensive guide for network security using Netflow and IPFIX [99]. Cisco presents OpenSOC, an integral solution to protect again intrusion, zero-day attacks and known threats in big data frameworks. OpenSOC includes many functionalities to parse measurements and load them into big data frameworks using, for example, Apache Flume and Kafka.

All these works reinforce the challenge posed by the lack of NTMA formats friendly to big data frameworks. At the one hand, transformations boost analysis performance, and performance seems to be the main focus of the community so far. At the other hand, information may be lost in transformations. Eventually, data replicated in many formats increase redundancy and make harder integration with other systems.

2) Feature engineering: Several libraries exist to perform feature extraction and selection in big data frameworks. While many of them are domain-specific, generic examples are found in Spark ML and Spark MLlib.[15] Instead, the research about feature extraction and selection in NTMA is scarce in general. Here the main challenges seem to be the lack of standard or

[15]https://spark.apache.org/mllib/

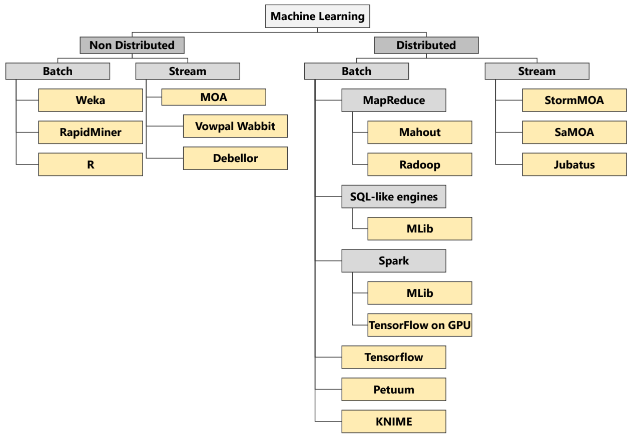

Figure 5: Machine learning for big data (based on [13]).

large-accepted features for each type of NTMA applications. Minimally, the community lacks ways to compare and benchmark systems and features in a systematic way.

Most works either refer to old datasets, e.g., [3], [25], [73]. The work in [57] studies traffic features in classic datasets for attack/virus detection and DM/ML testing (e.g., DARPA [81] dataset). Such datasets have been criticized and their use discouraged [78]. Authors consider whether the features are suitable or not for anomaly detection, showing that features present high correlation, thus mostly being unnecessary.

Abt et al. [1] study the selection of NetFlow features to detect botnet C&C communication, achieving accuracy and recall rates above 92%. In [62] Netflow features undergo feature selection for the case of DDoS detection. In [119] IPFIX records are used in feature selection processes, obtaining a set of key features for the classification of P2P traffic. All these papers handle relatively small datasets. Few authors rely on large datasets [5], but instead propose features that are finely tailored to the specific problem at hand. In a nutshell, each work proposes a custom feature engineering, with no holistic solution yet.

V. Big data analytics for NTMA

The next step of the NTMA path is data analysis. We recall that we do not intend to provide an exhaustive survey on general data analytics for NTMA. In particular, here we focus on how the main analytics methods are applied to *big* NTMA datasets. Detailed surveys of other NTMA scenarios are available, e.g., in [12], [44], [74], [85].

As for data management, generic frameworks exist and could be used for NTMA. We next briefly summarize them. After it, we dig into the NTMA literature.

A. Generic big data frameworks

Several taxonomies have been proposed to describe analysis algorithms. Algorithms are roughly classified as (i) statistical or (ii) machine learning. Machine learning are further categorized as supervised, unsupervised and semi-supervised, depending on the availability of ground truth and how data is used for training. Novel approaches are also often cited, such as deep neural networks and reinforcement learning.

Many challenges emerge when applying such algorithms to big data, due to the large volumes, high dimension etc. Some machine learning algorithms simply do not scale linearly with the input size, requiring lots of resources for processing big data sets. These problems are usually tackle by (i) preprocessing further the input to reduce its complexity; (ii) parallelizing algorithms, sometimes replacing exact solutions by more efficient approximate alternatives.

Several approaches have been proposed to parallelize statistical algorithms [51] or to scale machine learning algorithms [59], [95], [108]. Parallel version of some statistical algorithms are presented in [10]. Pébay et al. [91] provide a survey of parallel statistics. Parallel neural networks are described in [2], [80], [125]. Parallel training for deep learning are covered in [9], [26], [66].

Frameworks do exist to perform such analyses on big data. Fig. 5 reproduces a taxonomy of machine learning tools able to handle large data sets [13]. Both non-distributed (e.g., Weka and R) and distributed (e.g., Tensorflow and MLlib) alternatives are popular. Additional challenges occur with streaming data, since algorithms must cope with strict time constraints. We see in the figure that tools targeting these scenarios are also available (e.g., StormMOA).

In terms of NTMA analytics, generic framework implementing algorithms that can scale to big data could be employed too, naturally. Next we explore the literature to understand whether big data approaches and frameworks are actually employed in NTMA.

B. Literature categorization

In our examination, we focus on understanding the depth of the application of big data techniques in NTMA. In particular Tab. IV evaluates each work under the following perspectives:

- We check whether works face large data *volumes*. We arbitrarily define thresholds to consider a dataset to be big data: All works handling data larger than tens of GBs or works handling backbone network use cases. Similarly, we consider big data volumes when the study covers periods of months or years if dataset size is not specified.
- We verify if popular *frameworks* are used, i.e., *Spark*, *Hadoop*, *MapReduce*; we accept also custom implementations that exploit the same paradigms.
- We check if *machine learning* is used for NTMA.
- We verify if *big data platforms* are used in the ML process (see Fig. 5).
- We check *velocity*, i.e., if works leverage online analysis.
- We address *variety*, i.e., if authors use more than one data source in the analysis.

We leave out two of the 5 "V's", i.e., *veracity* and *value*, since it is cumbersome to evaluate them. For example, while some works discuss aspects of veracity (e.g., highlighting false positives of trained classifiers), rarely the veracity of the big data used as input in the analysis is evaluated.

Tab. IV shows that big data techniques are starting to permeate NTMA applications. Network security attracts more big data research. In general, it is interesting to notice the adoption of machine learning techniques. However, observe the limited adoption of big data platforms for machine learning.

Next, we dig into salient conclusions of this survey.

C. Single source, early reduction, sequential analysis

From Tab. IV, we can see how most of the works are dealing with the big volumes of data. This outcome is predictable since network traffic is one of the leading sources of big data. From the last column, variety, we can notice that the researchers infrequently use different sources together, limiting the analysis on specific use cases.

As we have seen before in Sect. IV, a group of works applies big data techniques in the first phases of the KDD process. This approach allows the analytics phase to be performed using non-distributed frameworks (Fig. 5).

For example, in [94] authors use Hadoop for real-time intrusion detection, but only computing feature values with MapReduce. Classic machine learning algorithms are used afterward on the reduced data. Similarly, Vassio et al. [122] use big data approaches to reduce the data dimension, while the classification is done in a centralized manner, with traditional machine learning frameworks. Shibahara et al. [104] deploy a system to classify malicious URLs through neural networks, analyzing IP address hierarchies. Only the feature extraction is performed using the MapReduce paradigm.

In summary, we observe a majority of works adopting a single (big data) source, performing early data reduction with the approaches described in Sect. IV (i.e., pre-processing data), and then performing machine learning analysis with traditional non-distributed platforms.

D. Big data platforms enabling big NTMA

A small group of papers performs the analytics process with big data approaches. Here different directions are taken. In *Hashdoop* [38], authors split the traffic into buckets according to a hash function applied to traffic features. Anomaly detection methods are then applied to each bucket, directly implemented as Map functions. Authors of [72] present a distributed semi-supervised clustering technique on top of MapReduce, using a local spectral subspace approach to analyze YouTube user comment-based graphs. Authors of [68] perform both pre-processing of network data using MapReduce (e.g., to filter out non-HTTP GET packets from HTTP traffic logs) as well as simple analytics to summarize the network activity per client-server pairs. Lastly, the Tsinghua University Campus has tested in its network an entropy-based anomaly detection system that uses IPFIX flows and runs over Hadoop [111].

In traffic classification, Trevisan et al. [114] developed AWESoME, an SDN application that allows prioritizing traffic of critical Web services, while segregating others, even if they are running on the same cloud or served by the same CDN. To classify flows, the training, performed on large datasets, is implemented in Spark.

Considering other applications, some works consider the analysis of large amounts of non-traffic data with big data approaches. Authors in [106] use MapReduce as a basis for a distributed crawler, which is applied to analyze over 300 million pages from Wikipedia to identify reliable editors, and subsequently detect editors that are likely vandals. Comarela et al. [29] focus on routing and implement a MapReduce algorithm to study multi-hop routing table distances. This function, applied over a TB of data, produces a measure of the variation of paths in different timescales.

In a nutshell, here big data platforms are enablers to scale the analysis on large datasets.

E. The rare cases of online analysis

Only a few works focus on online analysis and even fewer leverage big data techniques. Besides the previously cited [94], [111], Apiletti et al. in [4] developed SeLINA, a network analyzer that offers human-readable models of traffic data, combining unsupervised and supervised approaches for traffic inspection. A specific framework for distributed network analytics that operates using Netflow and IPFIX flows is presented in [27]. Here, SDN controllers are used for the processing to improve scalability and analytics quality by dynamically adjusting traffic record generation.

In sum, online analysis in NTMA is mostly restricted to the techniques to perform high-speed traffic capture and processing, described in Sect. IV. When it comes to big data analysis, NTMA researchers have mostly focused on batch analysis, thus not facing challenges of running algorithms on big data streaming.

F. Takeaway

The usage of ML seems to be widespread, especially for network security and anomaly detection. However, just some works use machine learning coupled with big data platforms. A general challenge when considering machine learning for big data analytics is indeed parallelization, which is not always easy to reach. Not all machine learning algorithms can be directly ported into a distributed framework, basically due to their inherent centralized designs. This hinders a wider adoption of big data platforms for the analytics stage, constraining works to perform data reduction at pre-processing stages.

In sum, in terms of complexity, most ML algorithms scale poorly with large datasets. When applied to the often humongous scale of NTMA data, they clearly cannot scale to typical NTMA scenarios.

VI. Challenges, open issues and ongoing work

We discuss some open issues and future directions we have identified after literature review.

1 — Lack of a standard and context-generic big NTMA platform: The data collection phase poses the major challenges for NTMA. The data transmission rate in computer networks keeps increasing, challenging the way probes collect and manage information. This is critical for probes that have to capture and process information on-the-fly and transmit results to centralized repositories. Flow-based approaches scale better, at the cost of losing details.

Table IV: Analyzed papers divided by application category (rows) and big data characteristics (columns).

Category	Volume?	Big data framework?	ML based?	Big data ML?	Velocity - Online analysis?	Variety?
Traffic Prediction	[36] [102] [47] [123] [111]	[111]	[36] [102] [47] [123] [111]		[47] [111]	[36]
Traffic Classification	[4] [43] [71] [114] [122]	[4] [71] [18] [114] [122]	[4] [37] [43] [127] [71] [101] [18] [114]	[4] [114]	[4] [43] [37]	
Fault Management	[88] [5] [50] [118] [38] [61] [93] [27] [22]	[38] [27] [27]	[5] [50] [118] [38] [61] [93] [63] [122]	[38] [63]	[38] [61] [27]	[61] [27]
Network Security	[11] [94] [106] [55] [82] [68] [117] [48]	[11] [94] [104] [106] [72] [28] [121] [68] [48]	[94] [104] [106] [72] [28] [39] [121] [82] [117]	[28] [72] [111]	[11] [82] [94] [111] [48]	

To solve the lack of flexible storage formats we have seen in Sect. IV-D that researchers have developed APIs or layers that transform the data in pre-defined shapes. Those APIs are not generic and not comprehensive. As an example, in NTMA one would like to associate to a given IP address both its geographical (e.g., country) and logical (e.g., Autonomous System Number) location. There is no standard library that supports even these basic operations in the frameworks.

Considering analytics, few researchers tackled the problem from a big data perspective. There is a lack of generic approaches to access the data features, with the ability to run multiple analytics in a scalable way. Thus, researchers usually rely on single data sources and sequential/centralized algorithms, that are applied to reduced data (see Sect. V-C).

In a nutshell, the community has yet to arrive at a generic platform for the big NTMA problem, and most solutions appear to be customized to solve a specific problems.

2 — Lack of distributed machine learning and data mining algorithms in big data platforms limits NTMA: Several researchers started adopting machine learning solutions to tackle NTMA problems. However, as analyzed in Sect. V, most recent papers focus on "small data", with few of them addressing the scalability problem of typical big data cases. Most papers use big data techniques just in the first steps of the work, for data processing and feature extraction. Most of the machine learning analysis is then executed in a centralized fashion. This design represents a lack of opportunity. For example, applying machine learning with large datasets could produce more accurate models for NTMA applications.

From a scientific point of view, it is interesting to conjecture the causes of this gap: The reasons may be several, from the lack of expertise to platform limitations. We observe that the availability of machine learning algorithms in big data platforms is still at an early stage. Despite the availability of solutions like the Spark MLlib and ML tools that have started to provide some big data-tailored machine learning, not all algorithms are ported. Some of these algorithms are also simply hard to parallelize. Parallelization of traditional algorithms is a general problem that has to be faced for big data in general, and for big NTMA in particular.

3 — Areas where big data NTMA are still missing: From Tab. IV, it is easy to notice the lack of proposals in some important categories. For example, even though fault management is a category in which usually a great amount of data must be handled, few papers faced this problem with big data approaches. The reasons may be linked to what we discussed earlier, i.e., the lack of generic and standard NTMA platforms. Similarly, as examined in Sect. III, some categories of NTMA applications (e.g., QoS/QoE management) are hardly faced with big data approaches.

4 — Lack of relevant and/or public datasets limits reproducibility: To the extent of our survey, only two contributions disclose a public dataset, namely [93] and [114]. Few works use open data, like the well-known MAWI dataset which is used for example in [38], [121], the (outdated) KDD CUP '99 [39], [94], and Kyoto2006 [55]. Apart from these cases, public datasets are scarce and often not updated, posing limitations in reproducibility of researches as well as limiting the benchmark of new, possibly more scalable, solutions.

5 — Ongoing projects on big NTMA: We have seen a solid increase in the adoption of big data approaches in NTMA. Yet, we observe a fragmented picture, with some limitations especially regarding interoperability and standardization. In fact, ad-hoc methodologies are proliferating, with no platform to support the community.

In this direction, Apache Spot was a promising platform (see Sect. IV-B). Unfortunately, its development has stopped, thus questioning its practical adoption by the community and practitioners. PNDA is instead actively developed, and the project starts collecting interest from the community, albeit in its early stage. Beam[16] is a framework offering the unification of batch and streaming models, increasing portability and easing the work of programmers that do not need to write two code bases; yet, no applications for NTMA exists.

In sum, there is a lot of work to be done to arrive at a practical big data solution for NTMA applications. The NTMA community shall start creating synergies and consolidating solutions while relaying on the consolidated platforms offered by the big data community.

References

[1] S. Abt, S. Wener, and H. Baier. Performance Evaluation of Classification and Feature Selection Algorithms for Netflow-Based Protocol Recognition. Proc. of the INFORMATIK, pages 2184–2197, 2013.

[2] J. B. Ahn. Neuron Machine: Parallel and Pipelined Digital Neurocomputing Architecture. Proc. of the CyberneticsCom, pages 143–147, 2012.

[16]https://beam.apache.org

[3] F. Amiri, M. R. Yousefi, C. Lucas, A. Shakery, and N. Yazdani. Mutual Information-Based Feature Selection for Intrusion Detection Systems. *J. Netw. Comput. Appl.*, 34(4):1184–1199, 2011.

[4] D. Apiletti, E. Baralis, T. Cerquitelli, P. Garza, D. Giordano, M. Mellia, and L. Venturini. SeLINA: A Self-Learning Insightful Network Analyzer. *IEEE Trans. Netw. Service Manag.*, 13(3):696–710, 2016.

[5] B. Arzani, S. Ciraci, B. T. Loo, A. Schuster, and G. Outhred. Taking the Blame Game out of Data Centers Operations with NetPoirot. Proc. of the SIGCOMM, pages 440–453, 2016.

[6] F. Bajaber, R. Elshawi, O. Batarfi, A. Altalhi, A. Barnawi, and S. Sakr. Big Data 2.0 Processing Systems: Taxonomy and Open Challenges. *J. Grid Comput.*, 14(3):379–405, 2016.

[7] A. Bar, A. Finamore, P. Casas, L. Golab, and M. Mellia. Large-Scale Network Traffic Monitoring with DBStream, a System for Rolling Big Data Analysis. Proc. of the BigData, pages 165–170, 2014.

[8] D. Beaver, S. Kumar, H. C. Li, J. Sobel, and P. Vajgel. Finding a Needle in Haystack: Facebook's Photo Storage. Proc. of the OSDI, pages 47–60, 2010.

[9] Y. Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[10] J. Bennett, R. Grout, P. Pebay, D. Roe, and D. Thompson. Numerically Stable, Single-pass, Parallel Statistics Algorithms. Proc. of the CLUSTR, pages 1–8, 2009.

[11] K. Benzidane, H. E. Alloussi, O. E. Warrak, L. Fetjah, S. J. Andaloussi, and A. Sekkaki. Toward a Cloud-Based Security Intelligence with Big Data Processing. Proc. of the NOMS, pages 1089–1092, 2016.

[12] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network Anomaly Detection: Methods, Systems and Tools. *Commun. Surveys Tuts.*, 16(1):303–336, 2014.

[13] A. Bifet and G. D. F. Morales. Big Data Stream Learning with SAMOA. Proc. of the ICDMW, pages 1199–1202, 2014.

[14] Big Data Working Group. Big Data Taxonomy. Technical report, Cloud Security Alliance, 2014.

[15] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo. A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities. *J. Internet Serv. Appl.*, 9(16), 2018.

[16] T. Bujlow, V. Carela-Español, and P. Barlet-Ros. Independent Comparison of Popular DPI Tools for Traffic Classification. *Comput. Netw.*, 76(C):75–89, 2015.

[17] A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok. A Survey on Internet Traffic Identification. *Commun. Surveys Tuts.*, 11(3):37–52, 2009.

[18] P. Casas, J. Vanerio, and K. Fukuda. GML Learning, a Generic Machine Learning Model for Network Measurements Analysis. Proc. of the CNSM, pages 1–9, 2017.

[19] R. Cattell. Scalable SQL and NoSQL Data Stores. *SIGMOD Rec.*, 39(4):12–27, 2011.

[20] M. Čermák, T. Jirsík, and M. Laštovička. Real-Time Analysis of Netflow Data for Generating Network Traffic Statistics Using Apache Spark. Proc. of the NOMS, pages 1019–1020, 2016.

[21] R. Chaiken, B. Jenkins, P.-A. Larson, B. Ramsey, D. Shakib, S. Weaver, and J. Zhou. SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets. *Proc. VLDB Endow.*, 1(2):1265–1276, 2008.

[22] M. Chandramouli and A. Clemm. Model-Driven Analytics in SDN Networks. Proc. of the IM, pages 668–673, 2017.

[23] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst.*, 26(2):4:1–4:26, 2008.

[24] M. Chen, S. Mao, and Y. Liu. Big Data: A Survey. *Mobile Netw. Appl.*, 19(2):171–209, 2014.

[25] Y. Chen, Y. Li, X.-Q. Cheng, and L. Guo. Survey and Taxonomy of Feature Selection Algorithms in Intrusion Detection System. Proc. of the Inscrypt, pages 153–167, 2006.

[26] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-Column Deep Neural Networks for Image Classification. Proc. of the CVPR, pages 3642–3649, 2012.

[27] A. Clemm, M. Chandramouli, and S. Krishnamurthy. DNA: An SDN Framework for Distributed Network Analytics. Proc. of the IM, pages 9–17, 2015.

[28] R. Cogranne, G. Doyen, N. Ghadban, and B. Hammi. Detecting Botclouds at Large Scale: A Decentralized and Robust Detection Method for Multi-Tenant Virtualized Environments. *IEEE Trans. Netw. Service Manag.*, 15(1):68–82, 2018.

[29] G. Comarela, G. Gürsun, and M. Crovella. Studying Interdomain Routing over Long Timescales. Proc. of the IMC, pages 227–234, 2013.

[30] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's Highly Available Key-Value Store. *SIGOPS Oper. Syst. Rev.*, 41(6):205–220, 2007.

[31] L. Deri, A. Cardigliano, and F. Fusco. 10 Gbit Line Rate Packet-to-Disk Using n2disk. Proc. of the TMA, pages 441–446, 2013.

[32] I. Drago, M. Mellia, and A. D'Alconzo. Big Data in Computer Network Monitoring. In *Encyclopedia of Big Data Technologies*. Springer International Publishing, 2018.

[33] J. Erman, M. Arlitt, and A. Mahanti. Traffic Classification Using Clustering Algorithms. Proc. of the MineNet, pages 281–286, 2006.

[34] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras. A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. *IEEE Trans. Emerg. Topics Comput.*, 2(3):267–279, 2014.

[35] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17(3):37–54, 1996.

[36] P. Fiadino, V. Ponce-Lopez, J. Antonio, M. Torrent-Moreno, and A. D'Alconzo. Call Detail Records for Human Mobility Studies: Taking Stock of the Situation in the "Always Connected Era". Proc. of the Big-DAMA, pages 43–48, 2017.

[37] S. Fiadino, P. Casas, A. D'Alconzo, M. Schiavone, and A. Baer. Grasping Popular Applications in Cellular Networks with Big Data Analytics Platforms. *IEEE Trans. Netw. Service Manag.*, 13(3):681–695, 2016.

[38] R. Fontugne, J. Mazel, and K. Fukuda. Hashdoop: A Mapreduce Framework for Network Anomaly Detection. Proc. of the INFOCOM WKSHPS, pages 494–499, 2014.

[39] G. Frishman, Y. Ben-Itzhak, and O. Margalit. Cluster-Based Load Balancing for Better Network Security. Proc. of the Big-DAMA, pages 7–12, 2017.

[40] C. Fuchs. Implications of Deep Packet Inspection (DPI) Internet Surveillance for Society. Technical Report 1, Uppsala University, Media and Communication Studies, 2012.

[41] F. Fusco, X. Dimitropoulos, M. Vlachos, and L. Deri. pcapIndex: An Index for Network Packet Traces with Legacy Compatibility. *SIGCOMM Comput. Commun. Rev.*, 42(1):47–53, 2012.

[42] F. Fusco, M. P. Stoecklin, and M. Vlachos. NET-FLi: On-the-Fly Compression, Archiving and Indexing of Streaming Network Traffic. *Proc. VLDB Endow.*, 3(1-2):1382–1393, 2010.

[43] J. Garcia and T. Korhonen. Efficient Distribution-Derived Features for High-Speed Encrypted Flow Classification. Proc. of the NetAI, pages 21–27, 2018.

[44] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez. Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges. *Comput. Secur.*, 28:18–28, 2009.

[45] A. F. Gates, O. Natkovich, S. Chopra, P. Kamath, S. M. Narayanamurthy, C. Olston, B. Reed, S. Srinivasan, and U. Srivastava. Building a High-Level Dataflow System on Top of Map-Reduce: The Pig Experience. *Proc. VLDB Endow.*, 2(2):1414–1425, 2009.

[46] S. Ghemawat, H. Gobioff, and S.-T. Leung. The Google File System. *SIGOPS Oper. Syst. Rev.*, 37(5):29–43, 2003.

[47] R. Gonzalez, F. Manco, A. Garcia-Duran, J. Mendes, F. Huici, S. Niccolini, and M. Niepert. Net2vec: Deep Learning for the Network. Proc. of the Big-DAMA, pages 13–18, 2017.

[48] S. Hameed and U. Ali. Efficacy of Live DDoS Detection with Hadoop. Proc. of the NOMS, pages 488–494, 2016.

[49] J. Han, H. E, G. Le, and J. Du. Survey on NoSQL Database. Proc. of the ICPCA, pages 363–366, 2011.

[50] R. Harper and P. Tee. Cookbook, a Recipe for Fault Localization. Proc. of the NOMS, pages 1–6, 2018.

[51] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag New York, 2 edition, 2009.

[52] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras. Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX. *Commun. Surveys Tuts.*, 16(4):2037–2064, 2014.

[53] R. Hofstede, L. Hendriks, A. Sperotto, and A. Pras. SSH Compromise Detection Using NetFlow/IPFIX. *SIGCOMM Comput. Commun. Rev.*, 44(5):20–26, 2014.

[54] H. Hu, Y. Wen, T.-S. Chua, and X. Li. Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. *IEEE Access*, 2:652–687, 2014.

[55] T. Huang, H. Sethu, and N. Kandasamy. A New Approach to Dimensionality Reduction for Anomaly Detection in Data Traffic. *IEEE Trans. Netw. Service Manag.*, 13(3):651–665, 2016.

[56] L. T. Ibrahim, R. Hassan, K. Ahmad, and A. N. Asat. A Study on Improvement of Internet Traffic Measurement and Analysis using Hadoop System. Proc. of the ICEEI, pages 462–466, 2015.

[57] F. Iglesias and T. Zseby. Analysis of Network Traffic Features for Anomaly Detection. *Mach. Learn.*, 101(1-3):59–84, 2015.

[58] Intel. Data Plane Development Kit (DPDK), 2014.

[59] W. Jiang, E. Zavesky, S.-F. Chang, and A. Loui. Cross-Domain Learning Methods for High-Level Visual Concept Classification. Proc. of the ICIP, pages 161–164, 2008.

[60] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. Proc. of the SIGCOMM, pages 229–240, 2005.

[61] H. Kasai, W. Kellerer, and M. Kleinsteuber. Network Volume Anomaly Detection and Identification in Large-Scale Networks Based on Online Time-Structured Traffic Tensor Tracking. *IEEE Trans. Netw. Service Manag.*, 13(3):636–650, 2016.

[62] M. Kim, H. Na, K. Chae, H. Bang, and J. Na. A Combined Data Mining Approach for DDoS Attack Detection. Proc. of the ICOIN, 2004.

[63] S. Kobayashi, K. Otomo, K. Fukuda, and H. Esaki. Mining Causality of Network Events in Log Data. *IEEE Trans. Netw. Service Manag.*, 15(1):53–67, 2018.

[64] A. Lakshman and P. Malik. Cassandra: A Decentralized Structured Storage System. *SIGOPS Oper. Syst. Rev.*, 44(2):35–40, 2010.

[65] D. Laney. 3d Data Management: Controlling Data Volume, Velocity, and Variety. Technical report, META Group, 2001.

[66] Q. V. Le. Building High-Level Features using Large Scale Unsupervised Learning. Proc. of the ICASSP, pages 8595–8598, 2013.

[67] Y. Lee, W. Kang, and H. Son. An Internet Traffic Analysis Method with MapReduce. Proc. of the NOMS, pages 357–361, 2010.

[68] Y. Lee and Y. Lee. Detecting DDoS Attacks with Hadoop. Proc. of the CoNEXT, pages 7:1–7:2, 2011.

[69] Y. Lee and Y. Lee. Toward Scalable Internet Traffic Measurement and Analysis with Hadoop. *SIGCOMM Comput. Commun. Rev.*, 43(1):5–13, 2013.

[70] B. Li, M. H. Gunes, G. Bebis, and J. Springer. A Supervised Machine Learning Approach to Classify Host Roles on Line Using sFlow. Proc. of the HPPN, pages 53–60, 2013.

[71] M. Li, C. Lumezanu, B. Zong, and H. Chen. Deep Learning IP Network Representations. Proc. of the Big-DAMA, pages 33–39, 2018.

[72] Y. Li, O. Martinez, X. Chen, Y. Li, and J. E. Hopcroft. In a World That Counts: Clustering and Detecting Fake Social Engagement at Scale. Proc. of the WWW, pages 111–120, 2016.

[73] Y. Li, J.-L. Wang, Z.-H. Tian, T.-B. Lu, and C. Young. Building Lightweight Intrusion Detection System Using Wrapper-Based Feature Selection Mechanisms. *Comput. Secur.*, 28(6):466–475, 2009.

[74] H.-J. Liao, C.-H. R. Lin, Y.-C. Lin, and K.-Y. Tung. Intrusion Detection System: A Comprehensive Review. *J. Netw. Comput. Appl.*, 36(1):16–24, 2013.

[75] G. Maier, R. Sommer, H. Dreger, A. Feldmann, V. Paxson, and F. Schneider. Enriching Network Security Analysis with Time Travel. Proc. of the SIGCOMM, pages 183–194, 2008.

[76] J. Manyika and others. Big Data: The Next Frontier for Innovation, Competition, and Productivity, 2011.

[77] S. Marchal, X. Jiang, R. State, and T. Engel. A Big Data Architecture for Large Scale Security Monitoring. Proc. of the BigData.Congress, pages 56–63, 2014.

[78] J. McHugh. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 Darpa Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Trans. Inf. Syst. Secur.*, 3(4):262–294, 2000.

[79] M. K. McKusick and S. Quinlan. GFS: Evolution on Fast-forward. *Queue*, 7(7):10:10–10:20, 2009.

[80] T. Mikolov, A. Deoras, D. Povey, L. Burget, and J. Cernocky. Strategies for Training Large Scale Neural Network Language Models. Proc. of the ASRU, pages 196–201, 2011.

[81] ”MIT Lincoln Laboratories”. DARPA Intrusion Detection Data Sets, 1999.

[82] P. Mulinka and P. Casas. Stream-Based Machine Learning for Network Security and Anomaly Detection. Proc. of the Big-DAMA, pages 1–7, 2018.

[83] A. Murgia, G. Ghidini, S. P. Emmons, and P. Bellavista. Lightweight Internet Traffic Classification: A Subject-Based Solution with Word Embeddings. Proc. of the SMARTCOMP, pages 1–8, 2016.

[84] W. Nagele. Large-Scale PCAP Data Analysis Using Apache Hadoop, 2011.

[85] T. T. Nguyen and G. Armitage. A Survey of Techniques for Internet Traffic Classification Using Machine Learning. *Commun. Surveys Tuts.*, 10(4):56–76, 2008.

[86] B. Nickless. Combining Cisco NetFlow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics. Proc. of the LISA, pages 285–290, 2000.

[87] C. Orsini, A. King, D. Giordano, V. Giotsas, and A. Dainotti. BGPStream: A Software Framework for Live and Historical BGP Data Analysis. Proc. of the IMC, pages 429–444, 2016.

[88] K. Otomo, S. Kobayashi, K. Fukuda, and H. Esaki. Finding Anomalies in Network System Logs with Latent Variables. Proc. of the Big-DAMA, pages 8–14, 2018.

[89] S. Owen, R. Anil, T. Dunning, and E. Friedman. *Mahout in Action*. Manning Publications Co., Greenwich, CT, USA, 2011.

[90] M. Patterson. NetFlow Vs. Packet Analysis, 2013.

[91] P. Pebay, D. Thompson, J. Bennett, and A. Mascarenhas. Design and Performance of a Scalable, Parallel Statistics Toolkit. Proc. of the IPDPS, pages 1475–1484, 2011.

[92] R. Pike, S. Dorward, R. Griesemer, and S. Quinlan. Interpreting the Data: Parallel Analysis with Sawzall. *Sci. Programming*, 13(4):277–298, 2005.

[93] A. Putina, D. Rossi, A. Bifet, S. Barth, D. Pletcher, C. Precup, and P. Nivaggioli. Telemetry-Based Stream-Learning of BGP Anomalies. Proc. of the Big-DAMA, pages 15–20, 2018.

[94] M. M. Rathore, A. Paul, A. Ahmad, S. Rho, M. Imran, and M. Guizani. Hadoop Based Real-Time Intrusion Detection for High-Speed Networks. Proc. of the GLOBECOM, pages 1–6, 2016.

[95] V. C. Raykar, R. Duraiswami, and B. Krishnapuram. A Fast Algorithm for Learning Large Scale Preference Relations. Proc. of the AISTATS, pages 388–395, 2007.

[96] F. D. Sacerdoti, M. J. Katz, M. L. Massie, and D. E. Culler. Wide Area Cluster Monitoring with Ganglia. Proc. of the CLUSTR-03, pages 289–298, 2003.

[97] S. Sakr, A. Liu, and A. G. Fayoumi. The Family of Mapreduce and Large-scale Data Processing Systems. *ACM Comput. Surv.*, 46(1):1–44, 2013.

[98] T. Samak, D. Gunter, and V. Hendrix. Scalable Analysis of Network Measurements with Hadoop and Pig. Proc. of the NOMS, pages 1254–1259, 2012.

[99] O. Santos. *Network Security with NetFlow and IPFIX: Big Data Analytics for Information Security*. Cisco Press, Indianapolis, 1 edition, 2015.

[100] D. Sarlis, N. Papailiou, I. Konstantinou, G. Smaragdakis, and N. Koziris. Datix: A System for Scalable Network Analytics. *SIGCOMM Comput. Commun. Rev.*, 45(5):21–28, 2015.

[101] L. Schiff, O. Ziv, M. Jaeger, and S. Schmid. NetSlicer: Automated and Traffic-Pattern Based Application Clustering in Datacenters. Proc. of the Big-DAMA, pages 21–26, 2018.

[102] K. Shadi, P. Natarajan, and C. Dovrolis. Hierarchical IP flow clustering. Proc. of the Big-DAMA, pages 25–30, 2017.

[103] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy. BlindBox: Deep Packet Inspection over Encrypted Traffic. Proc. of the SIGCOMM, pages 213–226, 2015.

[104] T. Shibahara, K. Yamanishi, Y. Takata, D. Chiba, M. Akiyama, T. Yagi, Y. Ohsita, and M. Murata. Malicious URL Sequence Detection Using Event De-Noising Convolutional Neural Network. Proc. of the ICC, pages 1–7, 2017.

[105] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller. An Overview of IP Flow-Based Intrusion Detection. *Commun. Surveys Tuts.*, 12(3):343–356, 2010.

[106] M. Spina, D. Rossi, M. Sozio, S. Maniu, and B. Cautis. Snooping Wikipedia Vandals with MapReduce. Proc. of the ICC, pages 1146–1151, 2015.

[107] M. Stonebraker. NewSQL: An Alternative to NoSQL and Old SQL for New OLTP Apps, 2011.

[108] P. Sun and X. Yao. Sparse Approximation Through Boosting for Learning Large Scale Kernel Machines. *IEEE Trans. Neural Netw.*, 21(6):883–894, 2010.

[109] H. Tazaki, K. Okada, Y. Sekiya, and Y. Kadobayashi. MATATABI: Multi-Layer Threat Analysis Platform with Hadoop. Proc. of the BADGERS, pages 75–82, 2014.

[110] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: A Warehousing Solution over a Map-Reduce Framework. *Proc. VLDB Endow.*, 2(2):1626–1629, 2009.

[111] G. Tian, Z. Wang, X. Yin, Z. Li, X. Shi, Z. Lu, C. Zhou, Y. Yu, and D. Wu. TADOOP: Mining Network Traffic Anomalies with Hadoop. Proc. of the SecureComm, pages 175–192, 2015.

[112] B. Trammell, P. Casas, D. Rossi, A. Bar, Z. Houidi, I. Leontiadis, T. Szemethy, and M. Mellia. mPlane: An Intelligent Measurement Plane for the Internet. *IEEE Commun. Mag.*, 52(5):148–156, 2014.

[113] M. Trevisan, I. Drago, M. Mellia, and M. M. Munafo. Towards Web Service Classification using Addresses and DNS. Proc. of the TRAC, pages 38–43, 2016.

[114] M. Trevisan, I. Drago, M. Mellia, H. H. Song, and M. Baldi. AWE-SoME: Big Data for Automatic Web Service Management in SDN. *IEEE Trans. Netw. Service Manag.*, 15(1):13–26, 2018.

[115] M. Trevisan, A. Finamore, M. Mellia, M. Munafo, and D. Rossi. Traffic Analysis with Off-the-Shelf Hardware: Challenges and Lessons Learned. *IEEE Commun. Mag.*, 55(3):163–169, 2017.

[116] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos. Big Data Analytics: A Survey. *J. Big Data*, 2(1):21, 2015.

[117] S. O. Uwagbole, W. J. Buchanan, and L. Fan. Applied Machine Learning Predictive Analytics to SQL Injection Attack Detection and Prevention. Proc. of the IM, pages 1087–1090, 2017.

[118] R. Vaarandi, B. Blumbergs, and M. Kont. An Unsupervised Framework for Detecting Anomalous Messages from Syslog Log Files. Proc. of the NOMS, pages 1–6, 2018.

[119] S. Valenti and D. Rossi. Identifying Key Features for P2p Traffic Classification. Proc. of the ICC, pages 1–6, 2011.

[120] S. Valenti, D. Rossi, A. Dainotti, A. Pescapè, A. Finamore, and M. Mellia. Reviewing Traffic Classification. In *Data Traffic Monitoring and Analysis - From Measurement, Classification, and Anomaly Detection to Quality of Experience*. Springer, Heidelberg, 1 edition, 2013.

[121] J. Vanerio and P. Casas. Ensemble-Learning Approaches for Network Security and Anomaly Detection. Proc. of the Big-DAMA, pages 1–6, 2017.

[122] L. Vassio, D. Giordano, M. Trevisan, M. Mellia, and A. P. C. da Silva. Users' Fingerprinting Techniques from TCP Traffic. Proc. of the Big-DAMA, pages 49–54, 2017.

[123] S. Wassermann, P. Casas, T. Cuvelier, and B. Donnet. NETPerfTrace: Predicting Internet Path Dynamics and Performance with Machine Learning. Proc. of the Big-DAMA, pages 31–36, 2017.

[124] M. Wullink, G. C. M. Moura, M. Müller, and C. Hesselman. ENTRADA: A High-Performance Network Traffic Data Stream. Proc. of the NOMS, pages 913–918, 2016.

[125] J. Yuan and S. Yu. Privacy Preserving Back-Propagation Neural Network Learning Made Practical with Cloud Computing. *IEEE Trans. Parallel Distrib. Syst.*, 25(1):212–221, 2014.

[126] Y. Zhang, T. Cao, S. Li, X. Tian, L. Yuan, H. Jia, and A. V. Vasilakos. Parallel Processing Systems for Big Data: A Survey. *Proc. IEEE*, 104(11):2114–2136, 2016.

[127] J. Zhao, T. Tiplea, R. Mortier, J. Crowcroft, and L. Wang. Data Analytics Service Composition and Deployment on Edge Devices. Proc. of the Big-DAMA, pages 27–32, 2018.

Biographies

Alessandro D'Alconzo received the M.Sc. degree in Electronic Engineering with honors in 2003, and the Ph.D. in Information and Telecommunication Engineering in 2007, from Polytechnic of Bari, Italy. Since March 2018, he is head of the Data Science office at the Digital Enterprise Division of Siemens Austria. Between 2016 and 2018, he was Scientist at the Center for Digital Safety & Security of AIT, Austrian Institute of Technology. From 2007 to 2015, he was Senior Researcher in the Communication Networks Area of the Telecommunications Research Center Vienna (FTW). His research interests embrace Big Data processing systems, network measurements and traffic monitoring ranging from design and implementation of statistical based anomaly detection algorithms, to Quality of Experience evaluation, and application of secure multiparty computation techniques to cross-domain network monitoring and troubleshooting.

Idilio Drago is an Assistant Professor (RTDa) at the Politecnico di Torino, Italy, in the Department of Electronics and Telecommunications. His research interests include Internet measurements, Big Data analysis, and network security. Drago has a PhD in computer science from the University of Twente. He was awarded an Applied Networking Research Prize in 2013 by the IETF/IRTF for his work on cloud storage traffic analysis.

Andrea Morichetta (S'17) received the M.Sc. degree in Computer Engineering in 2015, from Politecnico di Torino. He joined the Telecommunication Networks Group in 2016 as a PhD student under the supervision of Prof. Marco Mellia, funded by the BIG-DAMA project. In summer 2017 he had a summer internship at Cisco in San Jose, CA. In 2019 he spent six months at the Digital Insight Lab of the AIT Austrian Institute of Technology as visiting researcher. His research interests are in the fields of traffic analysis, security and data analysis.

Marco Mellia (M'97-SM'08) graduated from the Politecnico di Torino with Ph.D. in Electronics and Telecommunications Engineering in 2001, where he held a position as Full Professor. In 2002 he visited the Sprint Advanced Technology Laboratories, CA. In 2011, 2012, 2013 he collaborated with Narus Inc, CA, working on traffic monitoring and cyber-security system design. His research interests are in traffic monitoring and analysis, and in applications of Big Data and machine learning techniques for traffic analysis, with applications to Cybersecurity and network monitoring. He has co-authored over 250 papers and holds 9 patents. He was awarded the IRTF Applied Networking Research Prize in 2013, and several best paper awards. He is Area Editor of ACM CCR, and part of the Editorial Board of IEEE/ACM Transactions on Networking.

Pedro Casas is Scientist in AI/ML for Networking at the Digital Insight Lab of the Austrian Institute of Technology in Vienna. He received an Electrical Engineering degree from Universidad de la República, Uruguay in 2005, and a Ph.D. degree in Computer Science from Institut Mines-Télécom, Télécom Bretagne in 2010. He was Postdoctoral Research at the LAAS-CNRS in Toulouse from 2010 to 2011, and Senior Researcher at the Telecommunications Research Center Vienna (FTW) from 2011 to 2015. His work focuses on machine-learning and data mining based approaches for Networking, big data analytics and platforms, Internet network measurements, network security and anomaly detection, as well as QoE modeling, assessment and monitoring. He has published more than 150 Networking research papers in major international conferences and journals, received 13 awards for his work - including 7 best paper awards. He is general chair for different conferences, including the IEEE ComSoc ITC Special Interest Group on Network Measurements and Analytics.