Employing the CEP Paradigm for Network Analysis and Surveillance

Ruediger Gad, Martin Kappes
University of Applied Sciences Frankfurt am Main
Frankfurt am Main, Germany
e-mail: {rgad, kappes}@fb2.fh-frankfurt.de

Juan Boubeta-Puig, Inmaculada Medina-Bulo
Universidad de Cádiz
Cádiz, Spain
e-mail: {juan.boubeta, inmaculada.medina}@uca.es

***Abstract*—In this paper, we present a network analysis and surveillance system based on the Complex Event Processing (CEP) paradigm. We demonstrate how complex event hierarchies based on single packets can be leveraged for detecting attacks such as, e.g., SYN Flooding, and present experimental performance results indicating that current CEP implementations running on consumer class computers are well capable of analyzing network traffic volumes with such patterns in the Gigabit range, rendering our approach applicable for enterprise network monitoring.**

Keywords*—*CEP; network analysis; network surveillance.

I. INTRODUCTION

Today, Information Technology (IT) and its underlying computer networks are the foundation of virtually all business infrastructures. Since mission-critical processes depend on the reliable operation of the IT, the continuous operation of the network has to be ensured. Thus, the quality, reliability, performance and serviceability of the underlying computer network are paramount. Consequently, systems for assuring the proper operation of computer networks are needed.

Computer networks resemble, in many ways, a complex nervous system interconnecting individual nodes. Their operation is affected by a broad range of factors such as, but not limited to, e.g., component failures, traffic overload, misconfigurations, or attacks. In order to assure the proper operation of a computer network, situations which negatively affect the network operation must be rectified in a timely manner. Ideally, critical situations are avoided altogether by detecting and reacting to conditions that may eventually become critical beforehand.

Thus, information about relevant situations is needed. In terms of computer networks this means that data about the computer network and especially the occurrences in it is required. Hence, the very first step for assuring the proper operation of a computer network is the collection and analysis of data about the computer network. This data is the very basis for all other subsequent activities like the notification of administrative personnel or the implementation of counter measures.

Gathering and analyzing data in today's complex computer network structures is a challenging task. In order to acquire a comprehensive overview it is insufficient to collect and analyze data at a single point with standalone tools. Instead, a network analysis and surveillance system must be capable of distributed operation, allow for the integration of heterogeneous data sources, and to quickly and flexibly grow or adapt to new situations.

Ideally, the general system structure will be suitable for an extension beyond network analysis and surveillance and support the integration of methods for reactions into the overall system. This way, perspectively, a unified network management and security infrastructure can be implemented that supports the whole network operation, maintenance, and security process.

Consequently, we chose a system approach that is very flexible, not bound to a specific application domain, and allows for a simple extension. Our approach is based on the Complex Event Processing (CEP) [1] paradigm. In this paper, we present the general idea of our approach to leverage CEP in the field of computer networks and analyze its practicability by exemplarily implementing methods for detecting different situations in computer networks and measuring the performance. While CEP had been already used in specialized and isolated areas in the application domain of computer networks, to the best of our knowledge, no overarching, general and unified system had been proposed yet.

The remainder of this paper is organized as follows: at first, we give a summary of different network analysis and surveillance approaches. Then, we give an overview over work related to our research followed by an introduction of the relevant technologies in the fields of network analysis and surveillance and CEP. Afterwards, we introduce our approach and present the prototype that we used to assess the feasibility. Following, we describe the approach we used to assess the feasibility of using CEP in the field of network analysis and surveillance and present and discuss our results. Finally, we conclude this paper with a summary and outlook on future work.

II. RELATED WORK

While other approaches on employing CEP and event-driven architecture (EDA) exist in the field of computer networks, these approaches are typically employed in specialized and isolated application fields only. However, they demonstrate that the CEP paradigm, in general, can be effectively applied in the application domain of computer networks.

Some approaches aim on implementing intrusion detection systems based on CEP [2][3]. These approaches solely focus on detecting intrusions and can be viewed as evidence that CEP is capable of modeling important situations in the area of computer networks. Our approach has a wider scope; we use CEP for general network analysis and surveillance. This includes the detection of intrusions, but also other use cases like network monitoring or detecting congestion situations, misconfigurations, or faults.

In [4] and [5], a joined infrastructure for detecting attack preparations like so called “stealthy port scans” based on CEP is proposed. This research shows the suitability of modeling computer network incidents by means of event patterns and illustrates the potential of CEP and EDA for creating distributed network surveillance systems. However, [4] and [5] are focused on a single, very specific application while we propose to apply the CEP paradigm in the whole field of network analysis, surveillance and eventually reaction and mitigation.

Another line of research considers CEP-based systems for network management [6]. This work primarily focuses administrative aspects and largely ignores technical problems like data acquisition, data analysis, or triggering reactions. In the long term, we plan to create a unified network analysis and surveillance system that enables the seamless integration of other components like management systems or reaction systems.

In [7], an approach on using CEP for anomaly detection in computer networks is presented. This work shows the potential of using CEP-based systems even for advanced analysis methods like anomaly detection. However, this work is limited on anomaly detection. In our approach, we plan to integrated a multitude of possible analysis methods including classical, pattern-based approaches, as well as machine learning and anomaly detection techniques.

Performance aspects, both in the field of network analysis and surveillance as well as in the field of CEP had also been subject to research. In [8], a cooperative approach on capturing and processing packets at wire-speed in computer networks is proposed. In this work, the load of capturing and processing packets in a network is distributed among multiple capturing components. This approach is not implemented with means of CEP and thus is not as versatile and flexible as our proposed approach.

In [9], a hardware-based CEP engine is presented. With this “in-hardware” CEP implementation a throughput of 20 Gbps could be achieved. This shows the performance potential of CEP systems and that CEP systems can be implemented in-hardware in order to increase the performance. Here, we use general-purpose Java implementations. However, depending on the requirements, special hardware-based solutions may become viable solutions as well.

There also exist well-known monitoring tools like Nagios and Zabbix [10][11]. These tools emphasize on the hardware and services and not on the network itself. Devices and parameters that are usually monitored include performance, like CPU and memory, disk space, temperature, databases, or power systems. Furthermore, typically, such tools display the gathered data in large dashboard-like views. As we will discuss in the presentation of our approach, we believe that it is very important to support the administrative personal by inferring and extracting the important information instead of overwhelming them by displaying all available information. Yet, the information gathered by such monitoring systems is also very valuable. Hence, we plan to incorporate such monitoring solutions in our system as well. Our chosen architecture already supports the flexible and effortless extension of our system and integration of different data sources and provides mechanisms for meaningfully processing heterogeneous data from different data sources.

III. Network Analysis and Surveillance

In order to maintain a reliable, robust, and operational computer network, information about the computer network and the occurrences in it must be collected and analyzed in order to identify issues early on. Apart from noticing problems that already affected the operation of a network like a defective hardware component, continuous monitoring of a computer network also enables to discern trends like, e.g., an increase in network traffic or error rates over time, and thus aids in identifying and resolving issues before they become critical. For example, an upcoming congestion situation could be extrapolated by a monitoring system long before the actual effects impair the network functionality as suitable measures can be prepared and implemented early enough. Additionally, continuous monitoring also allows for identifying suspicious activities which may be related to ongoing attacks or attack preparations.

Thus, the very first step in assuring the proper operation of computer networks is to gather information about the network and the occurrences in it.

The act of gathering this data is referred to as network analysis and surveillance or network reconnaissance [12]. While usually the analysis of the data and the deduction of meaningful information is also usually considered part of network analysis, surveillance or network reconnaissance, we primarily consider the data acquisition part here.

Data acquisition can be distinguished into passive and active approaches [13]. Passive data acquisition is based on observing the network only whereas active data acquisition also generates network traffic, e.g., by attempting to contact a device. Here, we only consider passive approaches due to space limitations.

Examples for activities that may be observed in computer networks include packets being transmitted, connections being established or torn down, attackers scanning a network, hosts being added or removed from a network, or services being announced, requested, or used. While there is a multitude of different activities that happen in computer networks, there are only few “elementary” data acquisition techniques, namely:

- Packet capturing (also known as sniffing);
- Connection tracking, and
- Netflow-based methods.

These techniques have different properties and different advantages and drawbacks. Packet capturing provides the most information [14] as, in principle, all transmitted information is accessible. However, the actual outcome depends on factors like, e.g., the placement of the data acquisition device, the network technology, or the network topology. In general, capturing (and analyzing) packets can be a costly operation. Considering a fully loaded Gigabit

Ethernet (1000Base-T) [15] transmitting Ethernet packets with a payload of 1,500 bytes only, we obtain a packet rate of approximately 82,000 packets per second; for minimum size packet (520 bytes) approximately 240,000 packets per second.

These rates are important because the monitoring system must be dimensioned accordingly. If the number of arriving packets exceeds the number of processed packets for a duration, the queue memory will eventually be exhausted and packets will be randomly discarded leading to information loss. Various approaches for achieving high capture rates exist, such as, special setups [8], optimized software [16], special hardware, or combinations thereof.

Connection tracking observes state changes of connections in a network [17]. While, formally, connection tracking is only applicable for stateful connection protocols such as, e.g., TCP [18], the notion of an implicit connection and states has also evolved for connectionless protocols, such as, e.g., UDP [19].

One advantage of connection tracking is that most implementations are highly optimized and known to perform very well. An example is nf_conntrack from the Linux Netfilter Project, which is, e. g., utilized by stateful Linux packet filters, like the well established iptables implementation. However, the amount of information which can be obtained from connection tracking is quite limited compared to packet capturing.

Netflows are also a performance-friendly form of providing information about network traffic in an aggregated form. A netflow is defined as unidirectional data flow between two endpoints and fully characterized by the 5-tuple of source and destination network layer address, source and destination transport layer port, and transport layer protocol type. When netflows are used for monitoring, further information about each flow is collected such as, e.g., packet counts and byte counts. From the perspective taken here, netflows can be considered similar to connection tracking and have the same advantages and disadvantages. It is worth mentioning that many commercial network interconnection devices likes switches and routers allow to export netflow data out of the box, and standards have been defined for exporting and importing netflow information [20][21].

A. *Desirable Properties*

In the following, we will outline desirable properties for a network analysis and surveillance system.

Distributed data collection: Nowadays, IT infrastructures and the underlying networks are spread topologically and topographically. Companies and governmental institutions have several subsidiaries, which are all integrated into the overall IT and network infrastructure. Even in smaller businesses the company network is usually composed of different subnetworks. In order to get a comprehensive overall picture of the situation in such networks, it is insufficient to collect data at a single place. Instead, data about all topographical and topological parts of the network is required. Hence, data must be gathered at a large number of different places in the computer network. The underlying network analysis and surveillance infrastructure must support this.

Heterogeneous data sources: many different ways for collecting data about computer networks exist. The individual approaches have advantages and disadvantages. While, e.g., packet capturing provides a very high detail of information, it is perfomance-intensive. Connection tracking, on the other hand, requires less performance but also offers less detailed information. The required level of detail strongly depends on the actual application scenario. Furthermore, as we will show, it is even possible to derive the same higher-level information from different types of natural events. Additionally, some approaches may deliver information that cannot be retrieved by other means. Thus, in order to acquire a complete overview of a network and the occurrences in it, ideally, multiple, heterogeneous data sources should be used.

Near realtime information: In computer networks, it is often required to swiftly respond to hazardous situations; e.,g., failures must be compensated for quickly or attacks must be stopped early. Therefore, the available information about a network must be as current as possible. Ideally, the information should be available immediately when incidents in a network happen. A network analysis and surveillance infrastructure should support the propagation of information in a timely manner such that the information is available in near realtime, i.e., “soon enough” or “in a timely manner” to react meaningfully [23]. We refer to this property as "near realtime" since strict definitions for “realtime” exist in other areas of computer science, particularly in the scope of embedded devices and realtime hardware.

Focused user interface: It is equally important that the important information can easily be found and that it is easily understandable. In the field of network analysis and surveillance, vast amounts of data are available. Thus, identifying the important information is crucial in order to enable taking correct decisions. Administrative personal can only perform its tasks efficiently and solve problems quickly if the important information is presented in an easily perceivable and understandable fashion. Confronting the administrative personal with too much information significantly slows down the decision making process. Presenting to much unimportant information may result in in-effective actions being taken. In the worst case, unimportant or even misleading information may result in the wrong, counter-productive decisions. Thus, a network analysis and surveillance system should assist the responsible administrative personal by presenting the important data in a way that is easily understandable and quickly perceivable.

While the above list of desirable properties of a network analysis and surveillance system is not meant to be extensive, we believe it covers the most important aspects for such a system. In the following, we will motivate our choices and results presented here by referring back to these desirable properties.

IV. COMPLEX EVENT PROCESSING

In the following, we will present the salient features of Complex Event Processing (CEP) and define some terminology. Please refer to the Event Processing Glossary [24] for more details.

Complex Event Processing (CEP) is an approach for processing data in form of events [1]. A typical usage scenario is to infer increasingly complex information from simpler information by correlating the simpler information with each other. Other frequently used actions are filtering or transformation of information.

In CEP, the basic unit of information is an "event". "Event", thereby, has at least two meanings. Firstly, so called "natural events" are occurrences that naturally happen in the field a CEP system is applied in. The field in which a CEP system is applied in is also referred to as the "application domain" of the system. In the application domain of network analysis and surveillance, a natural event is, e. g., a packet being transmitted or a connection state being changed. Secondly, the other meaning of "event" is the entity that is processed inside a CEP system and which contains information describing this event. This entity can also be seen as an "event object" that carries information inside a CEP system, possibly about a natural event. Please note that event objects inside a CEP system do not necessarily relate to natural events outside the CEP system. The information contained in such an event object is also referred to as "event properties".

Inside a CEP system, events are processed. One of the most powerful operations is the derivation of events from other events. We use the terms "simpler events" and "complexer events" to clarify the relationship between the processed and the derived events [25]. Complex events may be derived from simple events directly, but can be also be derived from other (intermediate) complex events.

The actual correlation of events happens in a component known as the "CEP engine". The CEP engine is one of the key components of every CEP system. For inferring complexer events from simpler events special processing rules are used. These processing rules are usually expressed in an Event Processing Language (EPL). The actual form and syntax of an EPL depends on the respective implementation. One popular way to express EPLs is in a way similar to Structured Query Language (SQL), but with additional functionality for correlating events [26].

A more general term in this context is "derived event". "Derived events" are events which were processed by some mean. Thereby, it doesn't matter if events had only been filtered or transformed or had been inferred from other events with means of a CEP engine.

Closely related to event objects is the idea of an "event type" or "event class". In a nutshell, an "event type" roughly defines the structure of concrete event objects. However, please note that an "event class" is not a class in the object-oriented paradigm. An event type definition is much less strict than a class definition in object-oriented languages. When considering the application domain of network analysis and surveillance one event type is that of a packet being transmitted. A packet can be of many different protocol types like TCP, UDP, ICMP, or ARP [27][28]. Clearly, all these protocols contain very different information and so do the resulting event objects inside the CEP system. So, an event type in a CEP system can be more seen roughly similar to an eXtensible Markup Language (XML) Schema Definition (XSD) than a class in the object-oriented sense.

When events are derived from each other, typically, events of one class or type are derived from events of another type. The relationship between these complexer and simpler events can be represented in a directed graph with the event types being the vertices and the edges resembling the derivation relations. The direction of the edges indicates the derivation relation from simpler events to complexer events. Such a graph representation of events and their relations is also referred to as "event hierarchy".

The event processing itself is usually composed of a larger number of other components. When the events are processed in a CEP system they usually flow through what is also called "Event Processing Network" (EPN). An EPN can consist of different components like event filters, event transformers, CEP engines, or communication infrastructures. In Figure 1, an exemplary EPN is depicted. The feedback loop at the event pattern matching component indicates that increasingly complex events can be derived in an iterative manner.

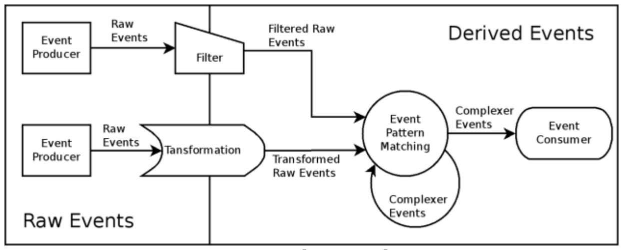

Figure 1: Simple Example EPN.

The communication infrastructure that is used in a CEP system can be manifold like common network sockets, higher-level infrastructures like Message-Oriented Middleware (MOM) [29] or an Enterprise Service Bus (ESB) [30] or traditional mechanisms for interprocess communication, like shared memory when components are locate on the same system.

V. OUR APPROACH

Much information about a computer network inherently has the nature of events. Many different things happen in computer networks all the time and naturally occur as events. The different events range from low-level to high-level. Examples of events happening in a network are packets being sent, connections changing the state, hosts "entering" or "leaving" a network, services being announced, requested, or used, devices failing, or attacks taking place.

Hence, in our opinion, an event-driven architecture [31] is the natural choice for processing data in the application domain of network analysis and surveillance [32]. In an event-driven system, the processing of data is triggered instantaneously upon the arrival of new data. Thus, an event-driven system supports the processing of data in near realtime. This actively helps to fulfill the desired property that a network analysis and surveillance system shall operate in near realtime.

In event-driven architectures, data exchange also follows the event-driven paradigm. One approach for communicating data inside event-driven systems is via messages. Messages simply contain the event data that needs to be exchanged. Thereby, messages can be exchanged locally via “classical” interprocess communication or via different hosts across computer networks. This allows to integrate components from different spatially distributed locations. Thus, we can address the desired property of integrating distributed event sources this way.

However, using the event-driven paradigm alone is not sufficient for addressing the other desired property: a focused user interface that aids the administrative personal in identifying and solving critical situations. It is crucial that the important information is clearly and easily perceivable and understandable. Thus, a mechanism for efficiently filtering information and inferring high-level information is required. Consequently, we chose CEP as paradigm because this not only offers all features of an event-driven architecture but also allows to filter information and most important of it all enables to infer high-level information.

Furthermore, the message-driven event exchange results in a loose coupling between the individual components. The loosely-coupled infrastructure in combination with the CEP paradigm allows for the simple integration of heterogeneous components. In Figure 2, the general architecture of our proposed system is depicted. Our prototype takes advantage of all these properties. It combines the capability of integrating heterogeneous data sources at distributed locations with an EDA and a CEP engine.

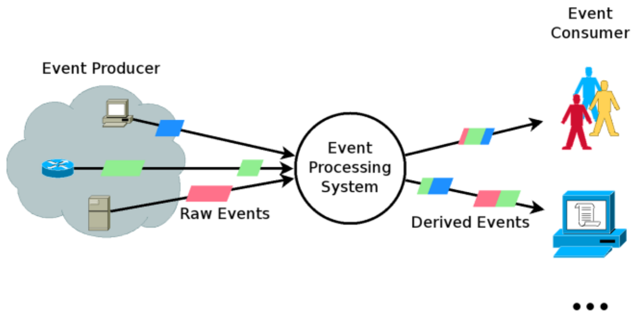

Figure 2: System Overview.

VI. EVALUATION

In order to assess the feasibility of our approach we implemented a CEP-based network analysis and surveillance system as prototype. We tested this prototype in real world network scenarios. Furthermore, we performed synthetic load tests for benchmarks.

In a CEP-based system, event patterns are used for processing information. For meaningfully applying a CEP system in a respective application domain, it is crucial that the intended application domain can be modeled with means of event patterns. To show that CEP can be meaningfully applied in the application domain of network analysis and surveillance we developed different event patterns and tested these in real-world network scenarios with our prototype. The event patterns were written in the Esper EPL language [26] and aim on detecting different situations in a computer network. These EPL patterns serve as proof of concept implementations and show that it is possible to effectively deduce meaningful high-level data in the application domain of network analysis and surveillance with means of CEP. In Figure 3, we show an exemplary event hierarchy for detecting Denial of Service (DoS) attacks.

We actually implemented different event patterns for the DoS attack scenario. It is, e. g., possible to conclude from a TCP SYN packet that is not followed by an established TCP connection that something irregular is happening; only the initial SYN packet was observed but the actual three-way handshake didn’t happen. When there is a large number of such events within a short time it can be concluded that a SYN flood is going on and in turn it can be concluded that a DoS attack is performed. The act of checking for the absence of a certain event as described here is also referred as the “absence pattern” [33]. Similarly, it could be concluded that an DoS attack is going on by just looking at the frequency with which SYN packets are sent without looking for non established connections. In the event hierarchy shown in Figure 3 the optional ways for deriving events are indicated by dashed edges. For the sake of brevity we only depict and explain one exemplary event hierarchy. However, we could successfully implement many other patterns, e.g., for detecting ARP spoofing attack, congestion situations, or brute-force attacks. This example illustrates that the domain of network analysis and surveillance can be modeled with means of event patterns.

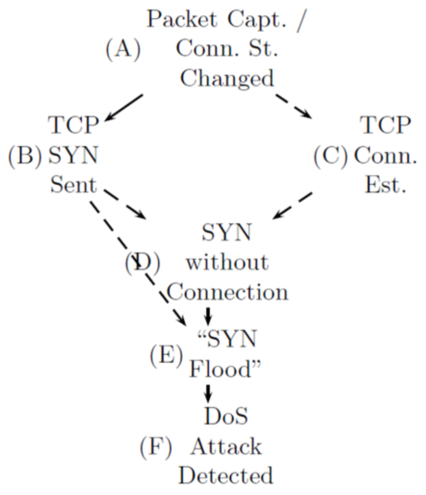

Figure 3: Exemplary Event Hierarchy for Detecting Denial of Service Attacks.

Furthermore, in the example from Figure 3, it is also possible to use events from connection tracking for detecting DoS attacks. This property can, e.g., be leveraged for optimizing the performance of a network analysis and surveillance system. In our prototype, we implemented and tested the DoS detection pattern shown in Figure 3 for natural events from packet capturing and connection tracking. Both variants showed to be capable of detecting the same DoS situations. With this example, we could show that, thanks to the loose coupling enabled by the CEP and EDA paradigm, the integration of heterogeneous data sources with our approach is simple and quick.

In order to assess the performance of a CEP system in the application domain of network analysis and surveillance, we exemplary used the Java-based, general purpose Esper CEP implementation. For our benchmarks, we used three different event patterns from the targeted application domain based on natural events from packet capturing: a simple pattern for simply filtering the input data, a pattern with average complexity for calculating the time between the occurrence of certain packets, and a complex pattern for detecting the TCP three-way handshake. The sample data had been collected with Wireshark. For the benchmarks, the data was synthetically fed to the CEP engine at different rates. For each measurement, the rate with which the data was “replayed” was constant. During each measurement run, we determined the CPU usage of the system and the percentage of events that could be successfully processed. The computer system on which we made the benchmarks was a common consumer class laptop with an Intel Core i5 CPU and 4GB memory. In order to avoid measuring artifacts of multi-CPU features as offered by this CPU, like impacts on multi-threading, etc., we artificially disabled the multi-processor functionality of the CPU.

The results of our benchmarks are shown in Figure 3. Please note that we used the rate of input events for the x-axis. The overall rate of events inside the CEP engine is usually higher because the derived events also attribute to the overall event rate. Similarly, we calculated the ratio of successfully processed input events for the statistics.

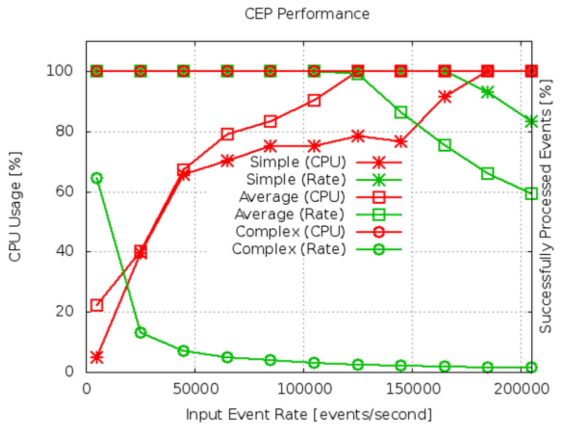

Figure 4: CEP Performance Comparison for Events with Different Complexity.

It can be clearly seen that the actual performance strongly depends on the complexity of the respective event patterns. Thereby, the percentage of successfully processed events is approximately 100% as long as enough CPU resources are available. Once the CPU resource limit is reached, the percentage of successfully processed events continuously decreases with increasing input event rate. Thus, we consider the highest rate of input events for which the percentage is still 100% as the maximum achievable rate for the respective event pattern. For the simplest patterns, we could achieve a throughput of ~170.000 events per second; with enabled multi-processing this was even higher, in the magnitude of 500.000 events per second. However, with multiple CPU cores, the result data showed some irregularities that we attribute to the multi-CPU features. For the sake of clarity we, therefore, disabled the multi-CPU support via the operating system. For the average complexity pattern, the maximum achievable throughput in our test setup is about 125.000 events per second. Meaningfully processing the most complex pattern was not possible at all; even at a rate of 5000 input events per second it was not possible to get a event processing rate near 100%. Still, this shows that even a general purpose CEP engine on consumer class commodity hardware CEP can generally perform with Gigabit Ethernet speed with a packet rate of roughly 82000 events per second. Yet, the actual performance strongly depends on the respective event patterns. With increasing event pattern complexity the performance degrades until it becomes impossible to meaningfully process the input data. However, this strongly depends on the applied event patterns. Furthermore, we purposely used packet capturing as data acquisition approach as it has the highest rate of emitting data. Other approaches like connection tracking or netflows emit data at a much lower speed. Additional ways for compensating performance issues are additional pre-processing and filtering steps.

Also, note that the CEP engine we used for benchmarking is a general-purpose implementation, which is intended to run on a large variety of platforms. In the field of CEP, there is also work and research on highly optimized CEP engines that leverage hardware acceleration.

VII. Conclusion and Future Work

Computer networks are crucial for the operation of nowadays IT infrastructures. Failures in computer networks very often directly impact the functionality of the corresponding IT with possibly severe consequences. Thus, maintaining operational computer networks is highly important.

Modern data processing paradigms, modern IT infrastructures and architectures, and increased performance open up new possibilities for gathering, processing, combining, and using data. We take advantage of this and propose an improved approach for network analysis and surveillance.

Based on an overview of existing approaches for network analysis and surveillance, we defined and explained desirable properties for a modern network analysis and surveillance system. Our approach on addressing these requirements is to leverage the CEP and EDA paradigm of data processing. Based on these technologies, we could successfully implement a distributed network analysis and surveillance system prototype that operates in near realtime and offers powerful functionality for processing, filtering, and enriching information. We tested our system in real world networking scenarios and with benchmarks. The results, so far, show that our approach works and is suited to fulfill the requirements we stated.

In future, we are going to further extend our system. We are currently working on integrating machine learning and anomaly detection techniques into the system. This way, we will further improve the capability of the system for deducing meaningful information and detecting important situations. We will also test the system in more scenarios and will optimize the performance.

REFERENCES

[1] D. C. Luckham, “The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems,” Addison-Wesley Longman Publishing Co., Inc., Amsterdam 2001.

[2] M. Ficco and L. Romano, “A Generic Intrusion Detection and Diagnoser System Based on Complex Event Processing,” First International Conference on Data Compression, Communications and Processing (CCP), IEEE Computer Society, Los Alamitos, 2011, pp. 275-284.

[3] J.J. Martinez-Molina, M.A. Hernandez-Ruiz, M.G. Perez, G.M. Perez, and A.F. Gomez-Skarmeta, “Event-Driven Architecture for Intrusion Detection Systems Based on Patterns,” Second International Conference on Emerging Security Information, Systems and Technologies SECURWARE ’08, Cap Esterel, 2008, pp. 391-396.

[4] L. Aniello, G.A. Di-Luna, G. Lodi, and R. Baldoni, “A Collaborative Event Processing System for Protection of Critical Infrastructures from Cyber Attacks,” Proceedings of the 30th International Conference on Computer Safety, Reliability, and Security SAFECOMP’11, Springer-Verlag Berlin, Heidelberg, 2011, pp. 310-323.

[5] L. Aniello, G. Lodi, and R. Baldoni, “Inter-domain Stealthy Port Scan Detection through Complex Event Processing,” Proceedings of the 13th European Workshop on Dependable Computing EWDC ’11, ACM New York, 2011, pp. 67-72.

[6] V. Krishnamoorthy, N.K. Unni, and V. Niranjan, “Event-driven Service-oriented Architecture for an Agile and Scalable Network Management System,” International Conference on Next Generation Web Services Practices, 2005.

[7] S. Cheng, Z. Cheng, Z. Luan, and D. Qian, “NEPnet: A scalable monitoring system for anomaly detection of network service,” 7th International Conference on Network and Service Management (CNSM), 2011.

[8] C. Morariu and B. Stiller, “DiCAP: Distributed Packet Capturing architecture for high-speed network links,” 33rd IEEE Conference on Local Computer Networks, LCN 2008.

[9] H. Inoue, T. Takenaka, M. Motomura, “20Gbps C-Based Complex Event Processing,” International Conference on Field Programmable Logic and Applications (FPL), 2011.

[10] Nagios Enterprises, “Nagios - The Industry Standard in IT Infrastructure Monitoring,” Online, http://www.nagios.org/, last accessed 2013-02-16.

[11] Zabbix SIA, “Homepage of Zabbix, An Enterprise-Class Open Source Distributed Monitoring Solution,” Online, http://www.zabbix.com/, last accessed 2013-02-16.

[12] S.A. Shaikh, H. Chivers, P. Nobles, J. A. Clark, and H. Chen, “Network Reconnaissance,” Network Security, 2008.

[13] S. Webster, R. Lippmann, and M. Zissman, Experience Using Active and Passive Mapping for Network Situational Awareness, Fifth IEEE International Symposium on Network Computing and Applications, 2006.

[14] C. Sanders, “Practical Packet Analysis : Using Wireshark to Solve Real-World Network Problems,” No Starch Press, Incorporated, San Francisco, 2007.

[15] The Institute of Electrical and Electronics Engineers, Inc., “IEEE Std 802.3-2008 Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications,” 2008.

[16] L. Deri, "nCap: wire-speed packet capture and transmission," *Workshop on End-to-End Monitoring Techniques and Services, 2005,* pp. 47-55.

[17] P. Ayuso, “Netfilter’s Connection Tracking System,” LOGIN: The USENIX magazine, Berkeley, 2006.

[18] J. Postel, “Transmission Control Protocol – RFC 793 (Standard),” Request for Comments, Internet Engineering Task Force, 1981.

[19] J. Postel, “User Datagram Protocol – RFC 768 (Standard),” Request for Comments, Internet Engineering Task Force, 1980.

[20] P. Phaal, S. Panchen, and N. McKee, “InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks – RFC 3176,” Request for Comments, Internet Engineering Task Force, 2001.

[21] B. Claise, “Cisco Systems NetFlow Services Export Version 9 – RFC 3954,” Request for Comments, Internet Engineering Task Force, 2004.

[22] M. Natu and A.S. Sethi, “Active Probing Approach for Fault Localization in Computer Networks,” 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services, 2006.

[23] C. Ballard, M.S. Roopa, O. Mueller, Z.Y. Pen, A. Perkins, and P.J. Suresh, “Preparing for DB2 Near-Realtime Business Intelligence,” IBM Redbooks, 2004.

[24] D.C. Luckham, et al., “Event Processing Glossary – Version 2.0,” Online, http://www.complexevents.com/wp-content/uploads/2011/08/EPTS Event Processing Glossary v2.pdf, last accessed 2013-02-16.

[25] R. Gad, J. Boubeta-Puig, M. Kappes, I. Medina-Bulo, “Hierarchical Events for Efficient Distributed Network Analysis and Surveillance,” Proceedings of the 2nd International Workshop on Adaptive Services for the Future Internet, WAS4FI-Mashups ’12, ACM New York, 2012, pp. 5-11.

[26] EsperTech Inc., “Esper - Event Stream and Complex Event Processing for Java Reference Documentation,” Online, http://esper.codehaus.org/esper/documentation/documentation.html, last accessed 2013-05-02.

[27] J. Postel, “Internet Control Message Protocol – RFC 792 (Standard),” Request for Comments, Internet Engineering Task Force, 1981.

[28] D.C. Plummer, “An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses – RFC 826 (Standard),” Request for Comments, Internet Engineering Task Force, 1982.

[29] B. Snyder, D. Bosanac, and R. Davies, “ActiveMQ in Action,” Manning Publications, 2011.

[30] D. Bo, D. Kun, and Z. Xiaoyi, “A High Performance Enterprise Service Bus Platform for Complex Event Processing,” Seventh International Conference on Grid and Cooperative Computing, IEEE Computer Society Washington, DC, 2008, pp. 577-582.

[31] H. Taylor, A. Yochem, L. Phillips, and F. Martinez, “Event-Driven Architecture: How SOA Enables the Real-Time Enterprise,” Addison-Wesley Professional, Boston, 2009.

[32] R. Gad, J. Boubeta-Puig, M. Kappes, and I. Medina-Bulo, “Leveraging EDA and CEP for Integrating Low-Level Network Analysis Methods into Modern, Distributed IT Architectures,” VII Jornadas de Ciencia e Ingenierıa de Servicios (JCIS - SISTEDES 2012), Almerıa, Spain, 2012.

[33] O. Etzion and P. Niblett, “Event Processing in Action,” Manning Publications Co., 2010.