Key Performance Indicators of TCP Flows

Patryk Brzoza
Advisor: M.Sc. Simon Bauer
Seminar Future Internet SS2018
Chair of Network Architectures and Services
Departments of Informatics, Technical University of Munich
Email: brzoza@in.tum.de

ABSTRACT

As most of today's Internet traffic is provided by TCP, this also means that the overall performance is heavily dependent on the quality of these TCP flows. To make it possible to evaluate the state of connections or networks for further potential problem assessment, it is necessary to introduce metrics that act as key performance indicators. This paper introduces multiple of these performance metrics and classifies them into latency, packet loss, throughput and other indicators. As a next step, various methods to conduct measurements and data processing approaches to extract valuable information are presented and discussed. Finally, this allows to draw conclusions about a flow's or network's quality and state.

Keywords

TCP Flows, Performance Metrics, KPIs, Network Monitoring, Intrusion Detection

1. INTRODUCTION

Due to its broad range of features and its numerous advantages, such as providing reliable and ordered communication, TCP has become one of the most important backbones of today's Internet. As most commonly used protocols (including HTTP, SMTP or FTP) are relying on it, even up to 95% of the total traffic volume is provided by TCP [1, 2]. Corresponding to this development, this also implies that the performance of TCP flows is playing a crucial role for the overall performance of applications that depend on the Internet. This implies that ISPs and network administrators should be interested in analyzing and optimizing it, which can be achieved through making use of various numerous network monitoring tools. By collecting and processing flow information, it is then possible to extract certain metrics that are essential for evaluation of the overall quality and performance of a network and its flows. This paper's goal is thus to find such suitable metrics to assess the performance of TCP connections and to determine how to obtain them effectively.

After discussing related work in Section 2 and revising the backgrounds of the TCP protocol in Section 3, this paper presents and categorizes multiple of these *key performance indicators* (KPIs) in Section 4. To acquire different KPIs for performance evaluation, it is however necessary to conclude certain measurement techniques first, which is further discussed in Section 5. As a next step, the collected raw data must then be - depending on its size - processed to reduce it to a necessary amount of information and to extract various KPIs from it. A range of different approaches from various tools is evaluated in Section 6. Finally, Section 7 shows which assumptions and conclusions about different network states and events can be made by inspecting the priorly collected information.

2. RELATED WORK

While this paper's focus is set on defining and evaluating KPIs of TCP flows, the performance of TCP has long been subject of previous research. A thorough analysis of the role of TCP/IP in high performance networks has been concluded by Hassan and Jain in [3]. Based on the fundamentals of the protocol, it defines how to measure the efficiency and create simulations and suitable models for wired and wireless networks.

Additionally, the passive extraction of valuable information like the round trip time has been thematized by Shakkottai et al. in [4] and forms a base for parts of this paper.

Finally, there is a broad range of tools that rely on monitoring network performance to conclude events like intrusion attacks. Examples for this research topic are the tool FlowScan by Plonka in [5] and FIRE by Dickerson in [6].

3. PROTOCOL BACKGROUND

As this paper evaluates the performance metrics of TCP flows, it is necessary to take a closer look at this protocol first. In 1974 the *Transport Control Protocol (TCP)* was first introduced by Cerf and Kahn in [7] and since then has been gradually improved. Compared to other packet-oriented protocols like UDP, which may deliver packets out of order or even lose them during the transmission, TCP allows for a *reliable*, *connection-oriented* and *ordered* communication between two nodes. The three terms imply that upon establishing a transport layer connection between two exchange partners, the transmitted data segments are guaranteed to arrive at their right destination in the correct order until the connection is terminated [8]. In [4], Shakkottai et al. describe a *flow* as a "transfer of packets between two ports of a source-destination pair using a particular protocol". For this reason, such a bidirectional TCP connection consists of two *flows* in each direction [4].

Apart from addressing the partners by their respective IP addresses, the TCP header - visible in Figure 1 - also includes a port number for each flow respectively to allow

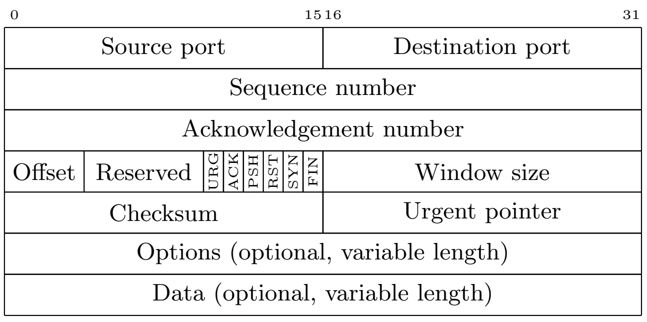

Figure 1: TCP Header [9]

inter-process communication. Additionally there are several other important fields, for instance sequence and acknowledgement numbers to keep the segments in order and retransmit lost packets, or flag bits to exchange control information that is necessary to conclude a three way handshake. This process is necessary to establish a new TCP connection between nodes and will be presented in Section 4. Finally, the congestion window defines how much data a node may send out to its partner before waiting for an acknowledgement. Sophisticated flow and congestion control algorithms are used to avoid congesting the network or the recipient, which may be caused because of sending out too many segments at the same time [9, 8].

4. KPI OVERVIEW

By further inspecting the features which were presented in the last section, it can be noted that many of these properties may constrain the performance of TCP flows. For instance, while retransmissions allow for a reliable communication, they also generate an additional overhead which impacts the ongoing communication by causing delays. This knowledge allows to derive certain metrics which act as key performance indicators and can be subdivided among *latency*, *packet loss*, *throughput* or *other indicators*, which is done in this section [10].

4.1 Latency Indicators

A major subclass of KPIs used to evaluate the connection quality and responsiveness of a TCP flow consists of temporal metrics measuring its latency, as high values can indicate severe delays and thus heavily affect the connection's performance.

Round Trip Time

TCP flows can be modeled in terms of *rounds*, which start with the sender beginning to transmit a window of segments and eventually finish upon receiving back acknowledgements for one or more of these segments from the recipient [11]. Using this assumption, the arguably most common metric in use is the *Round Trip Time* (*RTT*), which is defined as the time interval between a sender transmitting a segment and the reception of its corresponding acknowledgement segment. This interval may be influenced by several factors including propagation times, the processing and queuing times, which are dependent on the TCP stack implementations, or routing impacts [4]. Obviously it is necessary to keep this value low, as higher latencies would impact the flow performance significantly. An example illustration can be seen in Figure 2, further information about how to calculate the RTT from the monitor node and other measurement techniques are discussed in Section 6.

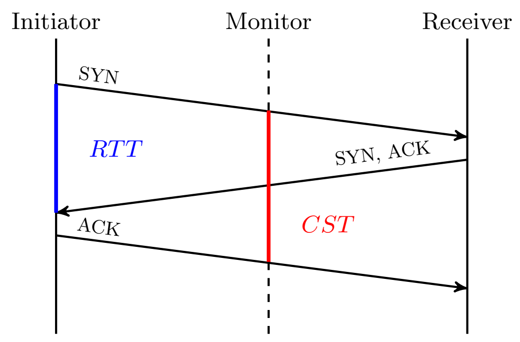

Figure 2: Initiator *RTT* and *CST* during a Three Way Handshake [10]

Connection Setup Time

Apart from the RTT, another metric to measure latency in networks is the *Connection Setup Time (CST)*, which describes the time interval that is necessary to establish a new TCP connection by performing a three way handshake. To gain the CST of a successful connection, it is necessary to inspect the time interval between the segments that are involved in the handshake process, i.e. the first SYN and the last ACK segment sent out by the initiator. The process of a full three way handshake can be seen in Figure 2, together with an illustration of the CST. Obviously this additionally implies, that this metric does include the RTT and can be influenced by different factors, for example due to retransmissions that were caused by lost acknowledgements [11]. A difference to the RTT metric measured in ongoing flows is that some implementations may prioritize handshake segments [10].

4.2 Packet Loss

As mentioned in the protocol background section, an important trait of TCP is its reliability, which is provided by detecting and retransmitting unacknowledged segments that may have been lost during their transmission. A common reason for this may be a congestion in the network. For this reason, packet loss is another key performance indicator, as every retransmission causes an additional delay and thus an increase in latency. [12]

A flow's packet loss rate can be visualized by comparing the amount of retransmitted segments r to the overall number of transmitted segments t during the inspected connection interval. Therefore, the *retransmission rate* (RR) can be calculated with the following formula:

$$RR = \frac{r}{t}$$

While retransmissions are a natural behavior in TCP, this value should nevertheless be kept low and sudden increases further investigated. [10, 13]

4.3 Throughput

A third category for evaluating flow performance is the analysis of its data throughput, which influences the responsiveness of a connection. This can once again be achieved by introducing analytical throughput metrics.

The *throughput rate* describes the current amount of data that is delivered between the connection nodes and may be dependent on several factors. For instance, TCP's flow control algorithms may throttle the throughput of a flow to avoid congesting the partner node [8]. As this rate is predefined by the congestion window, which additionally is inversely proportional to the RTT [4], this value is correlated to other performance metrics. Thus, given the current window size *cnwd* and RTT, the current throughput rate T can be estimated using this formula [9]:

$$T = \frac{cnwd}{RTT}$$

Provided that no new congestions are created, a higher throughput rate allows for better performance, while suspicious spikes should be examined in time.

Alternatively, another metric suitable for measuring throughput based on temporal units is the *data transfer time*, which is defined as the time interval beginning with the first and ending with the last data segment arriving at the client, thus describing the time to answer a request [11, 10].

4.4 Other Indicators

Apart from the aforementioned metrics, there are also several KPIs that do not fit into those categories, but may still impact a flow heavily and for this reason are presented in this subsection.

Response Time

For instance, the response time is an factor that is given by higher-layer server applications which are consuming services from the transport layer. The term describes the processing time between the last packet of a request and the first packet of a response (excluding acknowledgement segments) and gives information about the health of an application [10, 13].

Reset Rate

Finally, a way to measure a server's health is to keep track of the amount of sent RST segments over a long term. While TCP connections are torn down with FIN segments usually, RST packets can be used to immediately abort a connection instead and may indicate an error [8, 13].

Apart from the KPIs that were mentioned in this overview, a list including further metrics can be found in [10] and [13].

5. MEASUREMENT METHODS

To acquire the presented metrics for performance evaluation or problem assessment in networks, it is necessary to conduct measuring methods first. This can be achieved by applying two different approaches: *active* and *passive monitoring*. The idea of the active approach is to generate special probe segments that are then evaluated, while in passive measurement dedicated measurement devices are attached to network links in order to capture and process flow data from them passively. The concept of passive flow monitoring was also already introduced briefly in Section 4 for calculating the CST value.

While conducting active measurements gives us full control on the specific data we are currently interested in, it unfortunately also generates interferences to the original flows, as they are loaded with additional measurement data. For this reason, we choose the passive approach to be the more appropriate method to be applied on ongoing TCP flows, as we can apply it without constraining the performance unnecessarily. Apart from this, it first has to be decided, at which place in the network the monitoring points should be placed. An important task is then to adjust the locations and the amount of those measuring devices, so that the percentage of the monitored traffic is maximized, as unfortunate positioning of monitors could deliver poor results. A description of such optimization algorithms has been presented by Chaudet et al. in [14]. In [15] an environment for conducting passive measurements anywhere between the sending and receiving node is described, which is also similar to the description defined in [4]. Here, two unidirectional optical fibers - marked with 0 and 1 in each direction - connect the networks X and Y. These two networks contain nodes that communicate with each other. We tap both fibers - one fiber for each flow in the bidirectional connection - to siphon a part of the signal to the monitoring device, allowing it to collect segments from the connection that may be processed later on. This setup can is illustrated in Figure 3.

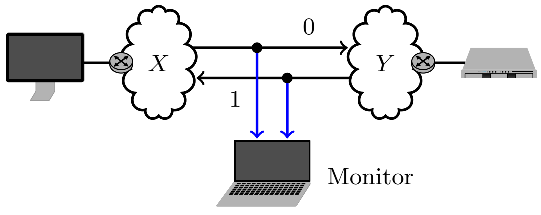

Figure 3: Example of a passive measurement setup used in [4]

Depending on the applicability, it makes sense to choose between acquiring data from one or both flows. Additionally, it is also necessary to detect which connections between which hosts and ports have been established to further concentrate on a concrete flow. Both of these measuring approaches together with possible solutions for the connectivity problem will be discussed in the following two subsections.

5.1 Bidirectional Approach

The *bidirectional approach* assumes, that the monitor has access to segments traversing both direction 0 and 1, meaning that it may access segments going from network X to Y and vice versa. This means that first of all, it has to be determined which hosts are communicating with each other on which ports to focus on a particular connection. One example on how to achieve this is to introduce and make use of a specialized metric that makes it possible to detect connectivity between two hosts [16].

A implementation of such a metric has been presented by Mahdavi and Paxson in [16] and is based on the Framework for IP Performance Metrics (IPPM) specified in [17]: we differentiate between a *instantaneous* connection, which took place at one certain moment in time, and between a *temporal* connectivity, which notes that a connection took place over a specified time interval. As a next step we can then define our metrics, i.e. either an instantaneous or temporal two-way connectivity consisting of two separate one-way connectivities in each flow direction. When we detect an SYN/ACK segment that acknowledges a handshake process in a flow, we may immediately return `true`. Another indicator for a temporal connectivity may be a RST segment with correct port numbers.

The KPIs presented in the overview can then be calculated easily by correlating relevant segments from a connection's flows with each other. For instance, the general approach for estimating the round trip time RTT, which is an essential performance metric, is capturing segments from one node to another together with its corresponding acknowledgement at the measurement point (Figure 2). By subtracting the timestamp values of the two segments, one can calculate an estimation of this value [15]. This value however depends on the location of the monitor and can easily become underestimated, which is why an alternative method is presented in Section 6 about data processing.

5.2 Unidirectional Approach

Unfortunately, it is not always possible to conduct bidirectional measurements. There are many reasons for this, for example the route between the two connection nodes could have different paths due to being asymmetric or we could not have access to such a measuring point, meaning that we can only capture a flow in one direction [18, 4]. For this reason, *unidirectional approaches* have become a new trend in monitoring research.

The first step is to decide, which flow we have access to and to identify which role the transmitting nodes are playing. By looking at the segments that are transmitted, it is possible to subdivide TCP flows into three categories, that are visualized in Figure 4 [4]:

- **Download Flows:** These flows carry large amounts of data. Capturing a SYN/ACK segment at the fiber going in direction 0, which is followed by data packets, may indicate that the receiver node is located in network X.
- **Feedback Flows:** Observing an initiating SYN segment that is followed by an ACK segment as an answer to the receiver may indicate that the initiator node is located in network X and the receiver node in network Y.
- **Unknown Flows:** Flows which have begun outside of the interval cannot be classified as their initiating segments were never captured.

To detect connectivity in unidirectional flows, we can then once again apply the metrics presented in the subsection before [16].

6. DATA PROCESSING

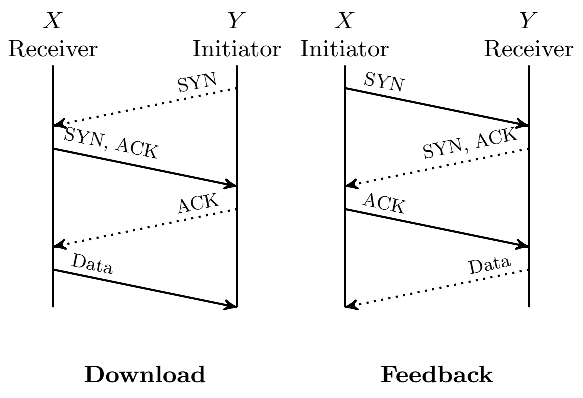

Figure 4: Examples of two possible unidirectional flow types [4]

After having collected enough measurement data, the next step usually is to process it in such a way, so that valuable information can be extracted efficiently and accurately. To achieve this, this section introduces various data processing approaches and evaluate their applicability.

6.1 Data Reduction

Unfortunately, monitoring and capturing packets from heavily loaded flows can easily generate vast amounts of raw data that are hard to process and manage. On the other side, due to collecting too little data one may lose valuable information, which is why it is important to find a good compromise between those two extremes [5]. For this reason, it makes sense to reduce the quantity of input to such a level, so that only necessary information is available. Such reduction techniques can be classified in three different categories [14]:

- **Filtering:** Only packets, that match certain networking criteria (e.g. port numbers or control flags) are captured, otherwise they are excluded.
- **Classification:** Instead of evaluating the whole data set, it is possible to subdivide it into multiple classes and then apply our processing techniques only on necessary classes.
- **Sampling:** Only capture packets randomly. This can be done by either capturing a packet every predefined time interval, every n packets, with a probability of $1/n$ or by combining the aforementioned criteria.

Which technique to use depends on the output we are interested in. For instance, sampling only requires a small amount of calculation and can be set up easily, but does not deliver as specific results as the other two techniques do [14].

Recently newer *data mining* approaches, which are defined as "the process of looking for features and relationships in data" [6], have gained in popularity. One example of such an approach is the implementation of the intrusion detection system FIRE presented in [6], which first extracts multiple fields from the TCP header and then creates an aggregate key defined as *sdp* consisting of the packet's source IP s, destination IP d and destination port number p. Using this key represents a flow's existence and is necessary for the data mining phase: a network data processor (NDP) maintains a table of selected sdp's and can be used to prepare statistics for them.

6.2 KPI Extraction

Eventually, after having processed the measurement data in an appropriate way, it is now possible to extract valuable KPIs from it. As a main example we have chosen to show how to derive and accurately estimate a flow's round trip time RTT, as it is probably one of the most important KPIs listed above, nevertheless other KPIs can be extracted in a similar way. Which approach to choose heavily depends on the measurement method we chose before.

For example, for bidirectional measurement data, this estimation depends on the location of the monitoring node and can so easily become underestimated. This can be seen in Figure 5, where the original RTT estimation $d1$ captured at the monitor turns out to be smaller than the expected value. One example on how to prevent this behavior is introduced with the RTT estimation method presented in [15], where the inferred estimation was split up into two parts, $d1$, which is the initiator RTT value we calculated before, and $d2$, where we calculate another receiver RTT value based on the previous acknowledgement segment coming from the receiver and a data segment coming from the initiator that was triggered by this acknowledgement. This process can also be seen in Figure 5. By inspecting the value $d = d1 + d2$. we see that we now finally derived a value that matches the real RTT as observed from the initiator more accurately.

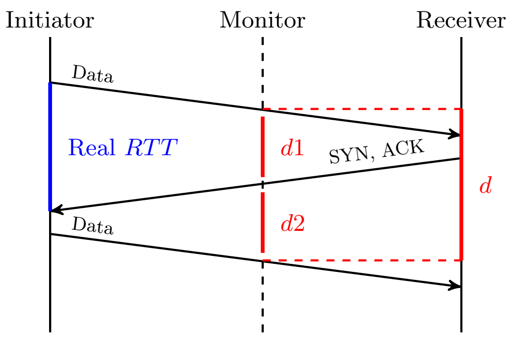

Figure 5: Passive RTT estimation method presented in [15]

This procedure is similar for the other KPIs, for which either the time intervals between segment samples can be calculated using the descriptions from the KPI overview in Section 4 or counted using a counter which increments itself upon e.g. the detection of RST segments to calculate the reset rate or a retransmission to calculate the retransmission rate. A more detailed graphic for this process that includes multiple TCP connection types has been described by Rogier in [10].

On the other side, using unidirectional measurement data for performance metric extraction turns out to be more complicated than the method described before, as we now have to take different flow types into consideration from which information may not instantly be obvious. In [4], three methods that can be used to calculate RTT are introduced:

- **SYN-based Method:** Both download and feedback flows are detected in direction XY. For download flows, this means that we are interested in the RTT for the path XYX, which can be done by calculating the time interval between the SYN/ACK and the data segment. Vice versa, for a feedback flow the path of interest is YXY, meaning that we are interested in the time interval between the initiating SYN and the following ACK segment. Both calculations may differ due to serialization delays.

- **Flight Method:** Due to TCP's congestion control, flows have a specific structure defined as *flight behavior*, whereas flights are defined as "consecutive packets with nearly identical inter-arrival times" [4]. This behavior implies that a flow may consist of groups of flights that are separated by gaps. It is then possible to derive an estimation of RTT by calculating and averaging the intervals between the first packet of each of those flight groups.

- **Rate Change Method:** For this method it is necessary to look back at the flow and congestion control algorithms implemented in TCP and described in the protocol background section. This leads to TCP having two different operation modes: the slow start mode, in which the congestion window increases by one maximum segment size (MSS) for every acknowledgement received, and the congestion avoidance phase, in which the MSS is incremented every RTT, while the congestion window size is halved for every segment that is lost. This special property can be exploited to determine an estimation for the RTT: assuming that x is the number of bits transferred in the time interval $[t_0, t]$, the instantaneous rate is defined as $\frac{dx}{dt}$ and that the interval $[t_0, t]$ is placed in the congestion avoidance phase, does not include packet drops and has a persistent RTT, then the overall RTT can be then calculated using this generic formula:

$$RTT = \sqrt{\frac{MTU}{\frac{d^2x}{dt^2}}}$$

  A detailed proof for this formula with more background information can be found in [4].

Out of the three methods, the SYN-based approach seems to be the most useful one as it delivers estimates for more sets of data than the other two methods do [4]. Its complexity is also significantly lower than for the other methods as it does not require as many calculations. Additionally, this approach can also be readapted to extract other KPIs in a similar way as before.

7. NETWORK EVENT DERIVATION

Now that we have successfully derived several key performance metrics from our monitored TCP flows, we may now use them to gain further information about the state of connections or networks. To keep track of sudden changes or suspicious anomalies in those extracted KPIs, various network visualization tools that create graphical statistics can be used. One example for such a tool is FlowScan, which was specified by Plonka in [5] and provides a powerful option to generate such visualizations even for longer measurement intervals. The following subsections will thematize KPI applications in two different fields, from which assumptions by inspecting certain metrics can be made.

7.1 Intrusion Detection

Due to the persistent danger of becoming a victim of a cyber attack, one of the most important tasks of a network administrator is to keep the network safe from possible attacks from the outside and to detect them in time before it becomes compromised. The process of identifying this kind of activity in a network is defined as *network intrusion detection* [6]. For example, a first step of conducting an attack that exploits certain TCP functionalities may be a port scan, that iteratively sends SYN segments from the attacker node to a list of ports on the target to detect whether they are open. If that is not the case, a closed port would normally send back a RST segment to abort the connection process. Thus, a significantly higher reset rate may already be an indicator of such an attack type. Another type of network abuse attacks that heavily affects the performance of networks are denial-of-service (DoS) floods, which aim to make a target unavailable. A common method to accomplish this is to flood the victim with a mass of ACK segments, which have to be processed and may receive RST segments as a response. As before, this would bring a sudden increase of the reset rate, but other KPIs including the current throughput, latency or response time may also be affected by the mass of requests the server has to process [5]. An example visualization of a DoS flood captured in the tool FlowScan can be seen in Figure 6.

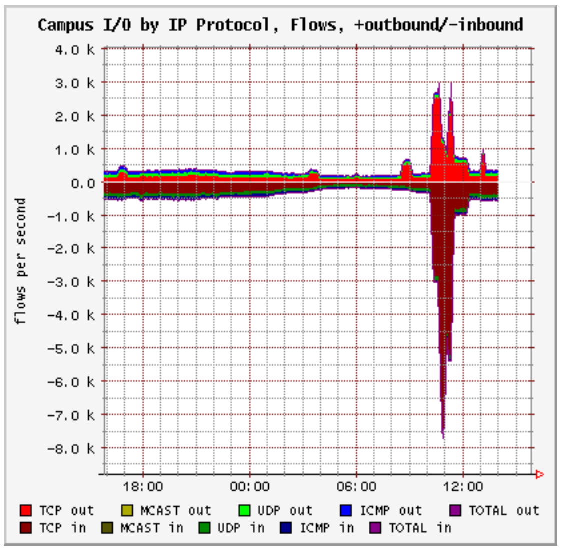

Figure 6: Example of a DoS flood attack visualized using FlowScan [5]

7.2 Network Health

Finally, a significant part of the overall performance is influenced by the network itself. A survey on speed limiting factors of TCP flows was conducted by Timmer et al. in [19] revealed and that approximately a quarter of the flow performance depends on the network itself. This can mean, that e.g. a persistently low throughput can be a sign of a bottleneck located in the network and should be investigated. An additional impact are the applications in use: a slow sender, that is using multiple applications that rely on TCP at once or that is running out of resources, can for instance have a high response time.

8. CONCLUSION

In this paper, we gave a brief summary of the TCP protocol together with its numerous features and the definition of the flow term. Due to the implementation of these features, the performance of TCP flows is constrained by multiple factors. We introduce metrics acting as key performance indicators for connections between end points. These KPIs were subdivided into four categories: latency, packet loss, throughput and other indicators. To be able to determine these values, it is necessary to conduct means of measurement first. We have differentiated between active and passive measurement methods and chose the latter option to be the more appropriate approach for ongoing flows. Depending on the accessibility and other factors, it makes sense to either monitor data from only one or both flows of a connection. Bidirectional approaches can easily detect ongoing connections using specialized metrics, while unidirectional approaches have to determine the flow type before proceeding. To extract specific information from these measurements, it is often necessary to reduce the amount of collected data first. We presented three data reduction methods together with newer data mining approaches to accomplish this task. We evaluated different KPI extraction methods by using round trip time estimation as an example: as for bidirectional data, it can be derived fairly easy by correlating associated segments with each other, while for unidirectional data the SYN-based method delivered the most reliable outputs. Finally, we showed the applicability of KPIs by applying it on intrusion detection and network error assessment.

9. REFERENCES

[1] M. Mellia and H. Zhang. Tcp model for short lived flows. *IEEE Communications Letters*, 6(2):85–87, Feb 2002.

[2] Anja Feldmann, Jennifer Rexford, and Ramón Cáceres. Efficient policies for carrying web traffic over flow-switched networks. *IEEE/ACM transactions on Networking*, 6(6):673–685, 1998.

[3] Mahbub Hassan and Raj Jain. *High Performance TCP/IP Networking*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[4] S. Shakkottai, N. Brownlee, A. Broido, and k. claffy. The RTT distribution of TCP flows on the Internet and its impact on TCP based flow control. Technical report, Cooperative Association for Internet Data Analysis (CAIDA), Mar 2004.

[5] Dave Plonka. Flowscan: A network traffic flow reporting and visualization tool. In *Proceedings of the 14th USENIX Conference on System Administration*, LISA '00, pages 305–318, Berkeley, CA, USA, 2000. USENIX Association.

[6] J. E. Dickerson and J. A. Dickerson. Fuzzy network profiling for intrusion detection. In *PeachFuzz 2000. 19th International Conference of the North American Fuzzy Information Processing Society - NAFIPS (Cat. No.00TH8500)*, pages 301–306, 2000.

[7] V. Cerf and R. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 22(5):637–648, May 1974.

[8] W. Richard Stevens. *TCP/IP Illustrated (Vol. 1): The Protocols*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[9] *Transmission Control Protocol Specification*. RFC 793, September 1981.

[10] B. Rogier. How to measure network performance through passive traffic analysis. `http://blog.performancevision.com/eng/earl/how-to-measure-network-performance-through-passive-traffic-analysis`. Last accessed on 2018/03/23.

[11] N. Cardwell, S. Savage, and T. Anderson. Modeling tcp latency. In *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064)*, volume 3, pages 1742–1751 vol.3, Mar 2000.

[12] P. Benko and A. Veres. A passive method for estimating end-to-end tcp packet loss. In *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, volume 3, pages 2609–2613 vol.3, Nov 2002.

[13] D. Shanahan. 5 Key TCP Metrics for Performance Monitoring. `https://www.linkedin.com/pulse/5-key-tcp-metrics-performance-monitoring-daniel-shanahan`. Last accessed on 2018/03/23.

[14] Claude Chaudet, Eric Fleury, Isabelle Guérin Lassous, Hervé Rivano, and Marie-Emilie Voge. Optimal positioning of active and passive monitoring devices. In *Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology*, CoNEXT '05, pages 71–82, New York, NY, USA, 2005. ACM.

[15] Sharad Jaiswal, G Iannaccone, C Diot, J Kurose, and D Towsley. Inferring tcp connection characteristics through passive measurements, 04 2004.

[16] *IPPM Metrics for Measuring Connectivity*. RFC 2498, January 1999.

[17] *Framework for IP Performance Metrics*. RFC 2330, May 1998.

[18] Hao Jiang and Constantinos Dovrolis. Passive estimation of tcp round-trip times. *SIGCOMM Comput. Commun. Rev.*, 32(3):75–88, July 2002.

[19] M. Timmer, P. T. de Boer, and A. Pras. How to identify the speed limiting factor of a tcp flow. In *2006 4th IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services*, pages 17–24, April 2006.