Information Sciences and Technologies Bulletin of the ACM Slovakia

Association for Computing Machinery

Advancing Computing as a Science & Profession

June 2020
Volume 12, Number 1

Aim and Scope of the Information Sciences and Technologies Bulletin of the ACM Slovakia

ACM Slovakia offers a forum for rapid dissemination of research results in the area of computing/informatics and more broadly of information and communication sciences and technologies. It is primarily a web based bulletin publishing results of dissertations submitted at any university in Slovakia or elsewhere, perhaps also results of outstanding master theses. Besides that, conferences that meet bulletin's expectations with regard to scientific rigor are invited to consider publishing their papers in the bulletin in form of special issues. Besides the web version of the bulletin, a paper version is available, too.

The Bulletin aims:

- To advance and to increase knowledge and interest in the science, design, development, construction, languages, management and applications of modern computing a.k.a. informatics, and more broadly of information and communication sciences and technologies.
- To facilitate a communication between persons having an interest in information and communication sciences and technologies by providing a forum for rapid dissemination of scholarly articles.

Scope of the Bulletin is:

- original research in an area within the broader family of information sciences and technologies, with a particular focus on computer science, computer engineering, software engineering and information systems, and also other similarly well established fields such as artificial intelligence or information science.

Types of contributions:

- **Extended abstracts of doctoral dissertations.** This is the primary type of article in the Bulletin. It presents main contributions of the dissertation in form of a journal paper together with separate section with list of published works of the author. In Slovakia and the Czech Republic, it corresponds to typical length of so called *autoreferat*. In fact, it is envisaged that publishing the extended abstract in the Bulletin makes *autoreferat* obsolete and eventually can replace it completely. It should be noted that by publishing it in the Bulletin, the extended abstract will receive a much wider dissemination. Exceptionally, at the discretion of the Editorial Board, the Bulletin may accept extended abstracts of other than doctoral theses, e.g. Master theses, when research results reported are sufficiently worthy of publishing in this forum. Rules and procedures of publishing are similar.
- **Conference papers.** The Bulletin offers organizers of interesting scientific events in some area within the scope of the Bulletin to consider publishing papers of the Conference in the Bulletin as its special issue. Any such proposal will be subject of discussion with the Editorial Board which will ultimately decide. From the scientific merit point of view, method of peer reviewing, acceptance ratio etc. are issues that will be raised in the discussion.

Besides that the Bulletin may include other types of contributions that will contribute to fulfilling its aims, so that it best serves the professional community in the area of information and communication sciences and technologies. There are four regular issues annually.

Editorial Board

Executive Editor: *Dominik Macko*
Cover Design: *Peter Lacko*
Typeset in LaTeX using style based on ACM SIG Proceedings Template.

Transparent Redirection in Content Delivery Networks using Software Defined Networking

Tomáš Boros*
Institute of Computer Engineering and Applied Informatics
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 2, 842 16 Bratislava, Slovakia
tomas.boros@stuba.sk

Abstract

The Internet is the technology of choice for mass distribution of information today. Content Delivery Networks, as the method used to deliver majority of the Internet's content, are becoming the most important piece of making this kind of communication sustainable. They do however impose some limitations as opposed to older legacy approaches. Content Delivery Networks heavily rely on redirections, which directs user requests to the closest surrogate server to lower the delay for content delivery and distribution on a massive scale. Redirections have not received a significant amount of attention in the recent year. This paper will show a TCP session handoff can be achieved using Software Defined Networking, which can be used a transparent redirection mechanism in Content Delivery Networks. The paper presents a working prototype including the achieved results with comparison with existing legacy approaches.

Categories and Subject Descriptors

C.2.0 [**Networks**]: General; C.2.1 [**Networks**]: Network Architecture and Design; C.2.3 [**Networks**]: Network Operations; C.2.6 [**Networks**]: Internetworking

Keywords

Software Defined Networking, Network Protocols, Transmission Control Protocol, Content Delivery Networks, TCP Session handoff, Session handover

*Recommended by thesis supervisor: Prof. Ivan Kotuliak Defended at Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava on February 14, 2020.

Boros, T. Transparent Redirection in Content Delivery Networks using Software Defined Networking. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 12, No. 1 (2020) 1-6

1. Introduction

Content Delivery Networks (CDN) are systems for efficient delivery of digital objects (e.g. files with multimedia content such as video on demand or other file types) and multimedia streams (e.g. live television streams) over IP networks to many end points and viewers. Typically, a CDN consists of one or more servers that deliver the digital objects and/or streams, and a management/control system. The management/control system takes care of content distribution, request routing, reporting, metadata and other aspects that make the system work. [8]

There are two main types of CDNs with global coverage today. The first type is a single CDN operated by a single entity that is managing all the infrastructure globally. Some examples of such CDNs are Akamai, Google Cache, Limelight, Level 3 and others. The second kind is represented by a group of smaller CDNs with limited coverage that, when combined, create a single service with global coverage. This kind of CDN is called a CDN Federation [4].

One of the main drawbacks is delay introduced in CDN. In this article, we propose a novel approach to session redirection using an SDN-based TCP session handoff. Using this approach, the delay becomes similar to DNS based redirection while having the accuracy of application layer redirection. The redirection occurs fully transparently to all the components. Furthermore, the proposed solution does not require any additional changes to the endpoints.

The rest of the paper is organised as follows: Section II gives insight into the state of the art. Section III is the core of our proposal using a TCP handoff mechanism. Section IV introduces the used testbed which was used to evaluate the prototype. Section V presents the achieved results and the final conclusions are given in Section VI.

2. State of the Art

2.1 CDN Redirection Methods

Redirection is the crucial part of the content delivery. A proper redirection mechanism ensures that the user requests are redirected to the closest optimal surrogate server in the CDN network. Barbir et al. in RFC3568 lists and compares the known CDN request-routing mechanisms [3]. The RFC lists the following types of redirect mechanism:

- DNS based Request-Routing mechanism
- Application-Layer Request-Routing mechanism
- Transport-Layer Request-Routing mechanism

Each mechanism has its advantages and drawbacks. DNS based redirection works by redirecting the client to the surrogate server during the DNS resolve. In such case the Request Router in the CDN works as a DNS server and dynamically returns an A or AAAA record based on the client's IP address or the client's DNS server's IP address if the DNS translation occurs in a recursive way. The DNS response points to a surrogate server, which delivers the content to the client. Such redirection is fast, however does not always returns the optimal surrogate server. [17]

Application layer mechanism occurs on the application layer of the TCP/IP model. In such case the request router works as a surrogate server, however does not return any content but only generates redirection messages to a surrogate server which caches and delivers data to endpoints. Each content distribution protocol which supports redirection may be used in CDN e.g. HTTP, RTSP, RTMP, DASH etc. Such redirection method is more accurate as application layer parameters may be analyzed during the redirection such as User Agent, URI during HTTP or available codecs in SDP during RTSP redirections. As the redirection message must be processed and often an additional domain name resolve must occur if the redirection message contains a domain instead of an IP address, this redirect method generates delays on session initiation. To speed up the redirection often a URL rewriting mechanism is used, which modifies the contents' URL in HTML to point directly to the surrogate server [3].

Transport-Layer Request-Routing is not a commonly used redirection method as it requires a service awareness and support from the network too. Transport-Layer request routers use information in the transport-layer headers to determine which CDN surrogate server should serve the client. The request router examines the IP address and port numbers in the TCP SYN and forwards to an appropriate CDN server. The target CDN server establishes the connection and proceeds to serve the requested content. Forward traffic from the client goes to the request router which is then forwarded to the CDN server but the response from the CDN server travels on the direct path to the client in a triangular routing way. This can be achieved using IPv6 too. IPv6 supports mobility which eliminates the triangular routing, however the majority of the clients are still using IPv4 [2].

2.2 Software Defined Networks

Traditional network technologies lack flexibility in implementing new features. Implementing new features and protocols may take years due to required standardizations, testing and doployment of the new code in a fully proprietary environment [16]. Software Defined Networking (SDN) presents a radically different approach. The key advantage is that the control plane is separated from the data plane. In traditional networks both the control plane and the data plane were confined within a single networking device, making development of complex control plane to data plane communication protocols necessary. In SDNs the data plane stays distributed but the control plane is removed from the physical device and placed into a centralized node responsible for managing all the data planes in the network. The data plane consists of a network of SDN forwarders which are not capable of making decisions. The data plane forwards and processes packets based on match rules and actions installed in their flow tables. If a does not match any rule, the forwarder might send the packet to the control plane via the Southbound API for further processing [20]. The control plane consists of a controller, a separated node which controls the data plane from a centralized view. It communicates with the forwarders in the data plane via the Southbound API and feeds the forwarders with instructions how to handle their incoming communications. One of the common protocols used in this API is OpenFlow. This centralized control plane is called as SDN Controller in SDN terminology. The SDN Controller is a fully software-based element that does not have the burden of having to communicate every single decision to any of its peers. This means that new features can be quickly added to the controller and they will be instantly available throughout the whole network that is under its control. The SDN Controller can control multiple SDN forwarders in the network. The resources available in the data plane under its controls are the Controller‘s only limiting factor. It does not have to follow a specific protocol that dictates exactly how these resources should be used. New features can be easily implemented and added to the network via the Controller's Northbound API. [5]

3. Transparent CDN Redirection Using SDN

Transport-Layer Request-Routing is transparent to the client, however it does not take into consideration the application layer parameters during redirection. To achieve higher accuracy, we must intercept packets on the data plane. In SDN world it is very easy to capture and analyze certain type of packets. Upon successful analysis, we can easily handover the session to the appropriate surrogate server. Such redirection is very easy for UDP based communication as UDP is stateless, however TCP is more often used as HTTP is encapsulated in TCP transport segments [1] [19].

TCP is a stateful transport protocol and the sessions are identified by a 4-tuple of source/destination IP addresses and ports. Data sent during a session must be acknowledged by the receiving side to ensure reliable data transfer. Each segment of data is labeled with a sequence number which is then acknowledged by the acknowledge number on responses. Sequence number is chosen dynamically on session initialization so it is not possible to hand over sessions easily. There were various attempts to hand off an existing live TCP sessions. Wichtlhuber et al. used the `TCP_REPAIR` flag in Linux kernels to migrate the socket form the terminating BRAS to a surrogate server, however such approach works only for Digital Subscriber Lines (DSL) and requires modification on the surrogate server to be able to accept the migrated session [21]. Guerney Hunt, Erich Nahum and John Tracey in IBM T.J. Watson Research Center already came up with an idea of handoff an existing TCP session from a load balancer to a content server in 1997 [15]. Authors in this report introduce the handoff mechanism of TCP sessions from the load balancer, where the initial TCP session from the client is established. After establishment, the clients send the HTTP request, which is examined and inspected by the load balancer, and based on the requested content, the TCP session is handed off to a content server, which has

the requested content cached. The initial TCP session parameters between the client and the load balancer are passed to the cache server (Windows sizes, various TCP options, Sequence Numbers) re-using the same TCP session parameters between the load balancer and the cache server. Once the session is handed off, the communication flows directly between the client and the cache server, while the IP address of the cache server is rewritten, to mimic the load balancer. It is not clear from the report, whether a working prototype was implemented or not. Authors also mention a different approach using T/TCP (Transactional TCP) where the initial HTTP request can be sent already in the 3 way handshake phase [6] [7]. This protocol is faster than TCP and delivery reliability is comparable to that of TCP. T/TCP suffers from several major security problems as described by Charles Hannum [11] [12]. It has not gained widespread popularity and the protocol was moved to historic status in May 2011 by RFC 6247. Authors also mention in the report a commercial product by Resonate Inc, which adopts the same mechanism, which they call âĂŸconnection hop', however, the product does not seem to be existing anymore [9]. Yutaro Hayakawa, Michio Honda, Lars Eggert and Douglas Santry in their paper [13] present Prism proxy for fast load balancing technique, where the TCP sessions established with the load balancer, upon reception of the request, are handed-off to a chosen backend server from the pool. The solution is designed for datacenter scale load balancing; however, it could be used for CDNs as a transparent redirection engine over greater areas. The Prism proxy uses the same TCP_REPAIR [10] flag to pass the sessions between the servers. The solution uses HTTP/1.1 and is capable of reusing existing TCP sessions for subsequent requests. The authors used mSwitch, which is a kernel software switch that can forward packets at a rate over 10 Mpps on a single CPU core [14]. To perform a transparent redirection the CDN must be able to perform a TCP session handoff, otherwise, the redirection becomes visible to the end user. All the mentioned related work above points out the challenges associated with the handoff procedure. In our proposal we do not hand off an existing TCP session from the Request Router to the Surrogate server, but instead we synchronize two different TCP sessions with each other by rewriting TCP and IP header fields including the acknowledgement and sequence numbers in the TCP header. The full session handover is depicted in the sequence diagram in Figure 1. Figure 1 is divided into 4 sections.

To make this solution work we must synchronize the sequence and acknowledgement numbers in the TCP headers too. Such modification could be implemented on a data plane as an experimenter action. The action must be able to increment these numbers to map the other TCP session's values. Once the sequence/acknowledge number reaches the maximum value of a 32bit number, it overflows and starts from 0. Upon a successful installation of actions to the data plane, the client consumes the response message from the surrogate server, which was originally generated for the request router (1st TCP session), while the client âĂIJthinksâĂİ the response is coming directly from the request router (2nd TCP session). In the end, to finish the handover procedure, the SDN controller must create reset (RST) or FIN messages to mitigate both sessions which were established with the request router. In order to be able to perform the handoff we have to keep track of the existing TCP sessions in

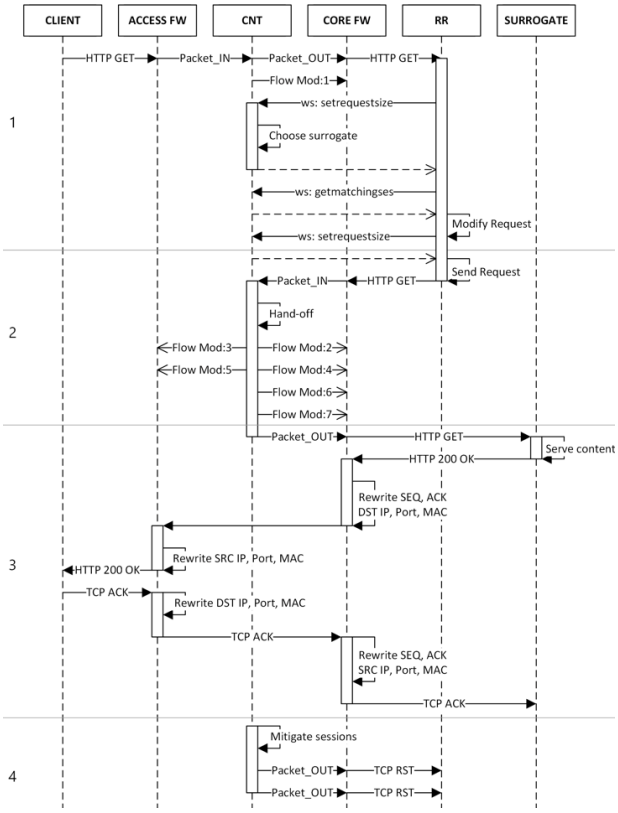

Figure 1: TCP session handover flow diagram.

the network. This can be achieved by intercepting the initial TCP handshakes with the request router and the surrogate servers. The request router prepares a pool of established TCP sessions towards the surrogate servers, which are reused for the handoff. Once the client establishes a TCP session with the request router a valid HTTP request message is expected. From this point the handoff procedure begins. The solution is designed to work with the standardized HTTP 1.1. In the 1st section of figure 1 the clients send an HTTP GET request including the URL and additional request headers. This request is sent to the controller (CNT) and then forwarded to the request router. Once the full request is sent to the request router the controller blocks the communication from the request router towards the client (Flow Mod:1). The request router parses the request and reports the request size to the controller over over the Northbound API via the setrequestsize message. This confirms the controller, that a full request was received. The request router requests the surrogate server to which the session must be handed off via the getmatchingsess message. Based on the returned information the request router modifies the request by updating the Host header in the HTTP request and reports the request's size to the controller via the setrequestsize message again. In the section 2 of figure 6 the request router sends out the request towards the surrogate server via an already pre-established TCP session (this session was chosen by the controller). At this point the controller has all the required information to perform the handoff. The controller installs flow mods to the core forwarder, where the surrogate is connected and to the access forwarder where the client is connected. The following flow mods are installed:

- Flow mod 2: rewrites destination IP, port and MAC address for the communication from the surrogate server towards the client, where the IP, port and MAC address is rewritten to mimic the request router. Sequence number and Acknowledge number is incremented to synchronize with the TCP session established between the client and request router
- Flow mod 3: rewrites the source IP, port and MAC address to match the parameters of the client. The packet sent out from the surrogate server is originally destined to the request router
- Flow mod 4: rewrites source IP, port and MAC address to match the parameters of the request router for the communication from the client towards the surrogate server. Sequence and acknowledge numbers are incremented to synchronize with the TCP session between the request router and the surrogate server.
- Flow mod 5: rewrites the destination IP, port and MAC address to match the parameters of the surrogate server as the outgoing packet from the client is destined to the request router.
- Flow mod 6 and 7 blocks any further communication from the request router towards the client or surrogate server over the two established TCP sessions.

Intermediary forwarders on the paths have simple match and output actions to forward the packets to the desired destination. In section 3 of figure 6 the modified request is sent out from the controller to the surrogate server, which parses the request and sends a response upon a valid request. The response and acknowledgement packets are forwarded through the network and the parameters are rewritten on the core and access forwarders based on the flow mods from section 2. The original TCP sessions established with the Request Router are mitigated by the controller by sending out TCP RST packets. Once the sessions are mitigated, the request router establishes a new TCP session towards the surrogate server to keep the pool of available TCP sessions alive. The following formulas are used to calculates the differences between the two TCP sessions:

$$Sinc_cs = ((2^{32}) + (Srs - Scr) + (Rrs - Rcr))mod(2^{32}) (1)$$

$$Ainc_sc = ((2^{32}) - Sinc_cs)mod(2^{32}) (2)$$

$$Sinc_sc = ((2^{32}) + (Src - Ssr))mod(2^{32}) (3)$$

$$Ainc_cs = ((2^{32}) - Sinc_sc)mod(2^{32}) (4)$$

Where:

- Srs: Initial sequence number from request router to the surrogate server
- Scr: Initial sequence number from client to the request router
- Rrs: Request size from the request router to the surrogate server
- Rcr: Request size from client to the request router
- Src: Initial sequence number from the request router to the client
- Ssr: Initial sequence number from the surrogate server to the request router

As the sequence and acknowledge numbers are 4 byte numbers, modulo 32 is applied to prevent overflow of this number. SeqCS (1) is applied in direction from client to the surrogate server as `inc_seq`(SeqCS) on the core forwarder (Flow mod 4). AckSC (2) is applied in direction from the surrogate server towards client as `inc_ack` (AckSC) on the core forwarder (Flow mod 2). SeqSC (3) is applied in direction from surrogate to client as `inc_seq` (SeqSC) (Flow mod 2) and AckCS (4) is applied in direction from client to surrogate as `inc_ack` (AckCS) on the forwarder where the surrogate server is attached (Flow mod 4). Action `inc_seq` and `inc_ack` are interpreted as datapath actions on the forwarders, which increment sequence and acknowledge numbers respectively.

4. Testbed

To evaluate the proposed solution we implemented a forwarding element, which is capable of incrementing the sequence and acknowledge numbers on the data path. We have chosen OpenVswitch for this purpose. The forwarder consists of two components. A fully userspace implementation of the forwarding datapath called ovs-switchd and a datapath kernel module written specially for the host operating system achieving high performance [18]. In addition to the modified SDN Forwarder we also needed our own SDN Controller that we could easily modify. We chose SDN Controller RYU, which is open-source, well documented and easy to modify. The controller right now support OpenFlow protocol up to 1.5, however as the data plane runs on version 1.3, we had to fall back to this version. The controller is capable of serving a single L2 network and the surrogate servers for the client requests are chosen based on the hop count. We implemented our custom Request Router, which was connected to the network to be able to terminate TCP sessions. The request router communicates with the SDN controller over the Northbound API during the redirection procedure. The testbed allowed us to test the implemented solution. We decided to use a custom build open source CDN solution which is based on Nginx. The implemented prototype allowed us to test the TCP session handover in an SDN network to enable Transparent CDN Redirection. As a result, we have achieved a fast transparent redirection method, which did not expose the physical topology (IP addresses) towards the clients. Some TCP Options (Timestamps, WSCALE, SACK) had to be turned off to make this kind of TCP session handoff work.

5. Evaluation

As next we made delay and bandwidth measurements on a network with 2 hops between the client and the surrogate server and we compared the proposed transparent redirection with the 302 redirection side by side. The topology used is shown in figure 2.

As already mentioned, the transparent redirection requires to be sent over the SDN controller, so the TCP session handshake traverses to the control node via the Packet IN and Packet OUT messages. This add some initial delay to the 3-way handshake. The time required

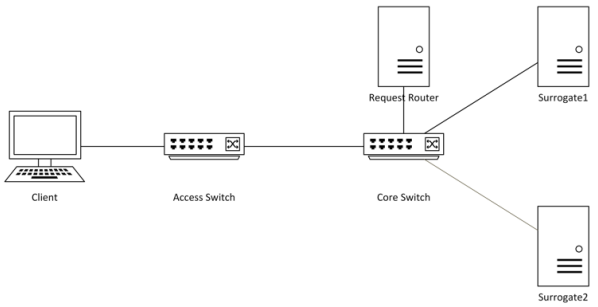

Figure 2: Topology for measuring overall redirection delays.

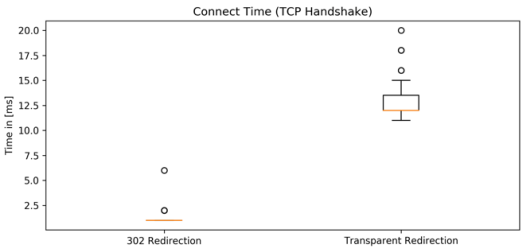

Figure 3: 3-way handshake delay.

to establish a TCP connection with the request router is shown in figure 3.

Once the TCP session is established, the request router expects a HTTP GET message from the client. Upon a valid request the client is redirected using a 302-redirection method toward the surrogate server, from where the content is served. Using the transparent redirection, a TCP session handover occurs to the surrogate server. Figure 4 shows the time required to process the redirection methods. The time between the sent HTTP GET message and receiving the first data bytes is measured.

To give some insight into the performance of the data plane, we did measurements on a large file (1GB). This size of the file is enough to reach high performance on the data plane due to the increasing window size. Figure 5 shows the whole time required to download the 1GB file from the surrogate server including TCP session establishment and redirection.

After each test we reported the mean bandwidth reached during the file transfer, which we put on a boxplot to

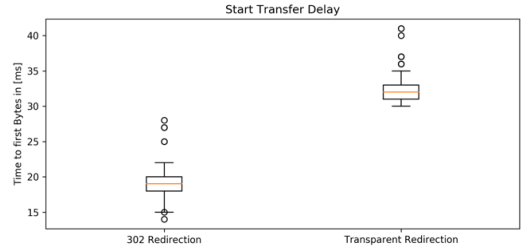

Figure 4: Time required to process redirection.

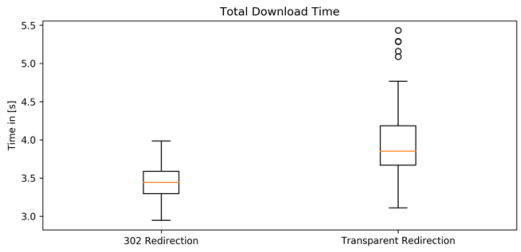

Figure 5: Total time required to download file.

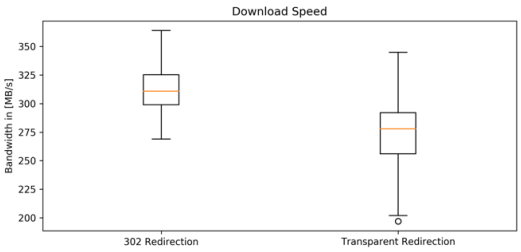

Figure 6: Average bandwidth reached.

evaluate the performance of the data plane itself. Figure 6 shows the average bandwidth reached during the file transfers. We have to note, when 302 redirection is used the forwarding elements only match the packets and forward them to the next hop based on output action in the flow table. No any other modifications are done on the packets. When transparent redirection is used, due to the TCP session synchronization, source IP, destination IP, source port and destination port are rewritten and sequence and acknowledge numbers are incremented. As we made our tests on a L2 network, source and destination hardware (MAC) addresses were rewritten too. This comes to a fact, that transparent redirection does 8 modifications on the packets, while the 302 redirection does not do any changes. Any change applied to a TCP or IPv4 Header also triggers a recalculation of the CRC checksum. Based on the achieved bandwidth (280 Mbit/s compared to 320 Mbit/s), we can state, that the modifications made on the Open vSwitch forwarder did not degraded the overall performance.

6. Conclusions

The paper proposes a TCP session handoff mechanism as a transparent redirection which synchronizes two existing TCP connections with each other. The main contribution of this project a prototype, which allows to hand off existing TCP sessions from one endpoint to the other without any further modifications to the endpoints. The transparent redirection ensures that the TCP session is handed off to a CDN surrogate from the request router, while the procedure stays unnoticeable to the endpoints. The redirection decision is based on the network topology and the location of the surrogates and the users. The results acquired show that current redirection methods (302 Redirection) are faster than the proposed solution, however, this performance could be enhanced by using a more mature and faster controller. Currently, the controller represents a single point of failure, does not scales and represents our performance bottleneck in the proposed solution. There were several issues found,

which should be solved in the future. One of them is the issue of pre-establishing TCP sessions form the request router to the surrogate servers with various WSCALE options to be able to make exact matches when we synchronize the TCP sessions. The incrementation of the acknowledgement numbers in the selective acknowledgement TCP Option should be implemented on forwarding elements too. Currently, these TCP Options including TCP Timestamps are turned off, which could degrade the performance in challenging environments. Furthermore, in the future, the possibility of using secured TLS based TCP sessions should be elaborated.

Acknowledgements. This work was supported by the projects ITMS 26240120029, ITMS 26240120005, ITMS NFP313011T570 (Grant agreement 035/2019/OPVaI/DP), and the grant 2018/14427:1-26C0.

References

[1] Cisco visual networking index: Forecast and methodology, 2016-2021, white paper.

[2] A. Acharya and A. Shaikh. Using mobility support for request-routing in ipv6 cdns. In *Proceedings of Proc. 7th International Workshop on Web Content Caching and Distribution 2012*, 2012.

[3] A. Barbir, B. Cain, R. Nair, and O. Spatscheck. *Known Content Network (CN) Request-Routing Mechanisms*. RFC Editor, United States, July 2003.

[4] A. Binder and I. Kotuliak. Content delivery network interconnect: Practical experience. In *2013 IEEE 11th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, pages 29–33, Stara Lesna, Oct 2013.

[5] A. Bondkovskii, J. Keeney, S. van der Meer, and S. Weber. Qualitative comparison of open-source sdn controllers. In *in NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, pages 889–894, 2016.

[6] R. T. Braden. Rfc 1379 - extending tcp for transactions – concepts, 1992.

[7] R. T. Braden. Rfc 1644 - t/tcp–tcp extensions for transactions functional specification, 1994.

[8] R. Buyya, M. Pathan, and A. Vakali. *Content Delivery Networks*. Number 5. November 2010.

[9] M. Cooney. Getting corporate intranets under control. *Network World*, 13(51):17–17, 1996.

[10] J. Corbet. Tcp connection repair. https://lwn.net/Articles/495304/, 2012. Accessed: 14-Apr-2019.

[11] C. Hannum. Security problems associated with t/tcp, 1996.

[12] C. Hannum. T/tcp vulnerabilities. *Phrack Magazine*, 8(53), 1998.

[13] Y. Hayakawa, L. Eggert, M. Honda, and D. Santry. Prism. In *in Proceedings of the 2017 Symposium on Cloud Computing - SoCC '17*, pages 181–188, 2017.

[14] M. Honda, F. Huici, G. Lettieri, and L. Rizzo. mswitch. In *in Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research - SOSR '15*, pages 1–13, 2015.

[15] G. Hunt, E. Nahum, and J. Tracey. Enabling content-based load distribution for scalable services. 09 2002.

[16] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, Jan 2015.

[17] J. Otto, M. Sánchez, J. Rula, and F. Bustamante. Content delivery and the natural evolution of dns: Remote dns trends, performance issues and alternative solutions. In *IMC 2012 - Proceedings of the ACM Internet Measurement Conference*, Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC, pages 523–536, 12 2012.

[18] B. Pfaff et al. The design and implementation of open vswitch. In *NSDI'15 Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation*, pages 117–130, 2015.

[19] J. Postel et al. Rfc 793: Transmission control protocol, 1981.

[20] V. Šulák, P. Helebrandt, and I. Kotuliak. Performance analysis of openflow forwarders based on routing granularity in openflow 1.0 and 1.3. In *2016 19th Conference of Open Innovations Association (FRUCT)*, pages 236–241, Nov 2016.

[21] M. Wichtlhuber, R. Reinecke, and D. Hausheer. An sdn-based cdn/isp collaboration architecture for managing high-volume flows. *IEEE Transactions on Network and Service Management*, 12(1):48–60, March 2015.

Selected Papers by the Author

A. Binder, T. Boros, I. Kotuliak. . A SDN based method of TCP connection handover. In *Information and communication technology : Third IFIP TC 5/8 Int. Conf. (ICT-EurAsia 2015), and 9th IFIP WG 8.9 Working Conf. (CONFENIS 2015), Held as Part of WCC 2015*, pages 13–19, Daejeon, Korea, 2015. Springer.

T. Boros, P. Zuraniewski, R. Hindriks, N. van Adrichem, E. Thomas, L. D'Acunto. Enabling superior and controllable video streaming QoE with 5G network orchestration. In *2019 22nd Int. Conf. on Innovation in Clouds, Internet and Networks (ICIN 2019)*, pages 124–129, Paris, France, 2019. IEEE.

T. Boros, I. Kotuliak. SDN-based transparent redirection for content delivery networks. In *TSP 2019 : 42nd Int. conf. on telecommunications and signal processing*, pages 373–377, Budapest, Hungary, 2019. IEEE.

Optimization of Network Monitoring

Ivana Palúchová*
Institute of Computer Engineering and Applied Informatics
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 2, 842 16 Bratislava, Slovakia
ivana.paluchova@stuba.sk

Abstract

The monitoring of a computer network is often challenging. On one hand, monitoring of every node and link gives us all the needed information. On the other hand, this approach generates huge amounts of data that need to be collected, analyzed and processed. In our work, we propose a monitoring model based on a selection of critical nodes in the network. The network monitoring can be applied only on a subset of network elements and therefore reduce the amounts of monitored, collected and processed data. Our proposed solution was verified in a simulation environment of Matlab R2018b. The testing shows that with monitoring of 64% of all nodes in the network we are able to gain 86.7% knowledge of all network elements. In terms of artificial traffic, we proved the reduction of 15.7% - 57.4% of bandwidth consumption in simulated SDN topologies.

Categories and Subject Descriptors

C.2.0 [**Networks**]: General; C.2.1 [**Networks**]: Network Architecture and Design; C.2.3 [**Networks**]: Network Operations

Keywords

network monitoring, performance monitoring, critical nodes, monitoring overhead, software-defined network

1. Introduction

Accurate knowledge of the status of the network is the most vital information for a network administrator. Up-to-date information about the traffic load, performance parameters or potential problems is crucial for everyday operation of the network. Monitoring of multiple network parameters on every link or node might create an issue, mainly in large and dense networks [1]. Time needed for measurement, collection and analysis of data may be too long compared to rapidly changing network conditions. The results might be outdated and selected reaction not adequate [2].

The new concept of networking brought by SDN changed also the approach to network monitoring [3]. The standard monitoring solutions designed for traditional IP networks are often not flexible enough to cover new opportunities of SDN architecture [4]. With the expansion of network scale grows the amount of data needed to be measured, collected and analyzed. In the context of SDN the performance of the controller itself can become the bottleneck of the requirements [5].

In our work we focus on problems in performance monitoring in large networks – both traditional IP networks and SDNs. Our goal is to propose a novel monitoring model, which could provide necessary information about the network status, while lowering the monitoring overhead and computational complexity. With the approach of selecting only important nodes for the monitoring we propose a solution to lower the amount of artificial traffic needed for monitoring and therefore fasten the collection, analysis and decision making.

The structure of this extended abstract is as follows. We present existing monitoring solutions for traditional IP networks as well as SDNs in the second section. The third section is dedicated to the proposed solution and used methodology. The evaluation and testing results are described in the fourth section. Last fifth section contains the conclusions of our work.

2. Related Work

In the following sections we provide an overview of the current state of the art in the area of network monitoring in general and in the area of SDN monitoring. The problems and complications known in traditional IP network monitoring appear also in the monitoring of SDNs. Furthermore, new challenges are rising in SDN, where the solutions and approaches need to be adjusted due to different network architecture and principles.

All of the analyzed proposals to network monitoring have one common feature – the need to find a balance between monitoring all details in the whole network and keeping the monitoring overhead low while obtaining necessary information about the network status. Applying moni-

*Recommended by thesis supervisor: Prof. Pavel Čičák Defended at Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava on June 26, 2020.

Palúchová, I. Optimization of Network Monitoring. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 12, No. 1 (2020) 7-12

toring to every network element may become an issue, mainly in large networks. Increased CPU overhead on the monitoring nodes, higher bandwidth utilization and huge amounts of monitored data increase the time and costs for the monitoring [6, 7].

2.1 Traditional IP Network Monitoring

We can distinguish two main categories of network monitoring – availability monitoring and performance monitoring. The first is concerned with the accessibility of the network devices such as application servers while the second type of monitoring deals with the network performance parameters on the network links and nodes, such as available bandwidth, delay, jitter or packet loss.

For both types of monitoring there is number of existing solutions on the market, such as [8, 9, 10, 11, 12]. These systems differ in depth of the analysis, features and graphical representations of the monitored data. Along with these "standard" solutions there are many new approaches being proposed by the academics.

In the area of availability monitoring researchers address various challenges such as heterogeneity and complexity in large size industrial networks, inefficiency of troubleshooting or need for a centralized monitoring solution. Authors in [13] proposed a novel and flexible monitoring solutions based on existing Nagios software. Authors in [14] focused on the collection and visualization of important information that characterizes the functionality and operating conditions of the network and network services.

The monitoring of device performance is crucial in the network along with the status and health monitoring. The question of performance monitoring is widely analyzed in the academic field, covering different areas of application. The common challenge, however, still remains - the possibility to monitor the network effectively, accurately and flexibly.

The question of QoS monitoring was analyzed and new approaches were proposed by authors in [15, 16]. Another area of interest is focused on monitoring of traffic flows in the network. Authors in [17] address the problem of local traffic monitoring overloads, publications [18, 19] proposed a parallel monitoring of flows to measure end-to-end delay in the network.

2.2 SDN Network Monitoring

The new concept of networking brought by SDN changed also the approach to network monitoring. The SDN architecture provides new possibilities to measure various network parameters, monitor link failures or topology changes. OpenFlow managed nodes usually report their status periodically to the SDN Controller. Furthermore, the Controller can request specific information such as flow statistics, interface statistics, etc.

The standard monitoring solutions designed for traditional IP networks are often not flexible enough to cover new opportunities of SDN architecture. This drives a huge number of academics into research in this area.

Authors in many publications addressed the challenges of QoS monitoring in SDNs: [20] proposed a passive approach to bandwidth measurement utilizing existing OpenFlow messages; [4] analyzed the possibilities of latency monitoring and potential bottleneck of the SDN controller, while proposing an optimization algorithm to reduce the time needed for delay measurements; [21] proposed a two-way link-level packet loss measurement solution using active measurement.

Traffic flow monitoring in SDN is challenging as it requires to install an entry per flow in the flow tables of every switch and the controller might become a bottleneck. Many researchers focused on this area and proposed novel solutions: [22] proposed a flow monitoring solution with classification where the measurements are maintained in the switches and are sent to the controller asynchronously; an interesting flow monitoring scheme is described in [23] where a new polling mechanism called poll-some is introduced. As compared to the two existing mechanisms, poll-single and poll-all, poll-some aims at collecting the statistics of multiple not-yet-covered flows at a switch.

Monitoring a large number of nodes can generate unsustainable loads on the central controller due to the increasing amount of measurement traffic converging to it [24]. To overcome this problem authors proposed number of different distributed solutions of the network monitoring: a distributed controller scenario was proposed in [25] which tries to minimize the measurement overhead, message interactions and CPU utilization on the SDN controller; in [24] the authors introduced a configurable interface for the synchronization of monitoring data between local managers operating within a distributed management environment.

3. Proposed Solution

The goal of our work was to design a new network monitoring model which will select the key network elements and identify their relationship with the rest of the network. The network monitoring can be then focused on these identified parts of the network instead of the whole network. In the end, the proposed model will provide information about all network elements without the need of extensive network monitoring. In result the proposed solution will cause less monitoring overhead, while still receive all needed information about the network status. Less bandwidth usage and less monitored data to be analyzed will provide faster decision making for the network manager and more available bandwidth for the end users.

An overview of the proposed model structure is shown in Figure 1. The primary input for the model consists of:

- network topology which models the real network
- user input such as type of NPPs (network performance parameters) to monitor

The primary input will be used for the calculations in our model. The calculations will produce following primary outputs which are also used as secondary inputs for the proposed model:

- identification of the important elements which should be monitored
- identification of the influences among network elements

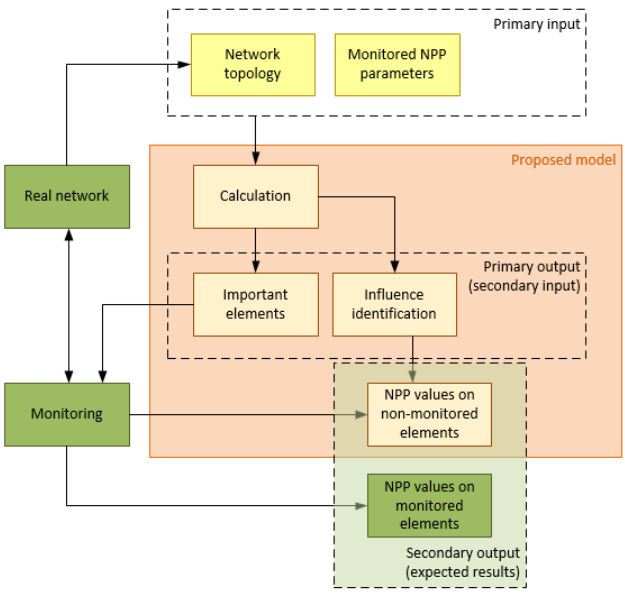

Figure 1: Proposed model structure.

The monitoring probes will be set to the identified important elements. Values obtained by monitoring together with the identified influence will result into NPP values estimation on elements which are not monitored.

3.1 Selection of Critical Nodes

The proposed approach lays in identifying set of *critical nodes* which will serve as main sources of monitoring data in the network. Overall network status will be calculated and estimated based on information gathered from these *critical nodes*. The identification of *critical nodes* is based on weights we assign to each node and link. The weight of node n_p is defined in Equation 1 and it is statically assigned based on the status of the node. The weight of link l_{ij} is defined in Equation 2 and it is dependent on the weights of nodes a, b which are connected by this link where node a represents the source node and node b represents the destination node of the link.

$$w_{n_p} = \begin{cases} 1 & \text{for critical node} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$$w_{l_{ij}} = w_{n_a} + w_{n_b}/2 \tag{2}$$

The goals for the selection of *critical nodes* are:

- minimize the number of selected *critical nodes*
- maximize the number of links with *full knowledge*
- cover all links in the network (no link can be left unmonitored)

Although an optimal solution for *critical nodes* selection can be found easily for small networks, it can be, however, a serious time-consuming and resource-demanding task for larger networks. Therefore, we formulate this problem as a multi-objective optimization problem consisting of two objectives as described in Equation 3 and Equation 4 and one constraint described in Equation 5.

$$min \sum_{p=1}^{S} w_{n_p} subject \begin{cases} \forall w_n & \text{are integer} \\ \forall n \in N \end{cases} \tag{3}$$

$$max \sum_{i,j=1}^{S} w_{l_{ij}} subject \begin{cases} \forall w_l & \text{are integer} \\ \forall l \in L \end{cases} \tag{4}$$

$$\forall l \in L : w_l > 0 \tag{5}$$

We used Matlab integer linear programing (*intlinprog*) function to solve the multi-objective optimization problem. The function as described in [26] finds the minimum of a problem specified by Equation 6 where f, x, $intcon$, b, beq, lb, and ub are vectors, and A and Aeq are matrices:

- f represents the objective function – sum of all node's weights as described in Equation 3
- x represents the decision variable – node's weight w_n
- $intcon$ specifies the components of x that are integer – all of the nodes are integer variables
- b represents the constant vector for the inequality constraint – described in Equation 7
- beq represents the constant vector for the equality constraints – not used, empty vector
- A represents the linear coefficients for the inequality constraint – described in Equation 7
- Aeq represent the linear coefficients for the equality constraints – not used, empty matrix
- lb, ub represent the lower and upper bounds of the decision variable – lower bounds set to edge nodes, upper bounds set to all nodes in the network

$$minf^T xsubject \begin{cases} x(intcon) & \text{are integers} \\ A.x \leq b \\ Aeq.x = beq \\ lb \leq x \leq ub \end{cases} \tag{6}$$

$$-2.w_{n_a} - w_{n_b} \leq -1 \tag{7}$$

3.2 Bandwidth Utilization and Packet Loss Calculation

With the *critical nodes* identified we can apply the network monitoring on the most important parts of the network. The results of NPPs monitoring will be combined with identified influence among the network elements. For the calculation we create set of equations and inequalities based on the following assumptions:

- packet loss on link can be expected if the link's load has reached the value of link's bandwidth
- the amount of incoming traffic is equal to the amount of outgoing traffic and packet loss for every node
- the traffic flow entering the node via interface i must be forwarded out via another interface
- the amount of traffic flowing through a link cannot be greater than the links' bandwidth

The number of equations is dependent on the density of the network. Equations are created for each node separately and for each possible path combination through the node. The inequalities are created for each link in the network. In general, the set of equations and inequalities are depicted in Equation 8 and Equation 9 respectively.

$$\forall n_p \in N : \forall l_{ip} \in L : LD_{l_{ip}} = \sum_{j=1}^{S} LD_{l_{pj}} + PL_{l_{pj}}; j \neq p \quad (8)$$

$$\forall l_{ij} \in L : LD_{l_{ij}} \leq BW_{l_{ij}} \quad (9)$$

4. Results

Our proposed solution was verified in a simulation environment of Matlab R2018b with the Optimization toolbox installed and in a virtual machine with Ubuntu operating system. We created five testing topologies and for each one we created three versions with different characteristics to represent sparse, medium and dense networks. For every combination we decided to calculate with four different setups regarding the number of *edge nodes*. With this approach we created 60 different network setups on which our proposed solution was tested as described in Table 1.

The verification of our proposed solution was divided into three sections:

- selection of critical nodes
- calculation of bandwidth usage and packet loss
- application of acquired results into SDN scenario

The validation results for selection of *critical nodes* are depicted in Figure 2 – Figure 4. The proposed model took into consideration all possible combinations of the *edge nodes*. We focused on the number of selected nodes in different networks and the influence of various network characteristics on the final number of *critical nodes*. The results are represented in percentual values where 100% defines all nodes in the network.

The average amount of needed nodes for effective monitoring is around 64% of all nodes in the network. This shows 36% reduction of needed network nodes for overall monitoring, not depending on the size of the network as described in Figure 2. The density of the network slightly influences the amount of needed network nodes as described in Figure 3. Figure 4 depicts the direct impact of number of edge nodes on the amount of needed network nodes.

Table 1: Parameters of Testing Networks

Topology	S	avg(d)	Number of edge nodes			
			2	S/4	S/3	S/2
1	8	2.3 3.3 4	2	2	3	4
2	10	2.4 3.2 4.2	2	3	3	5
3	12	2.3 2.8 3.8	2	3	4	6
4	20	2.3 2.9 4.2	2	5	7	10
5	30	2.3 3.1 4	2	8	10	15

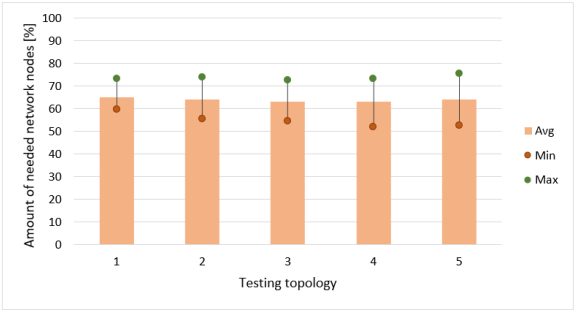

Figure 2: Nodes for monitoring per testing topology.

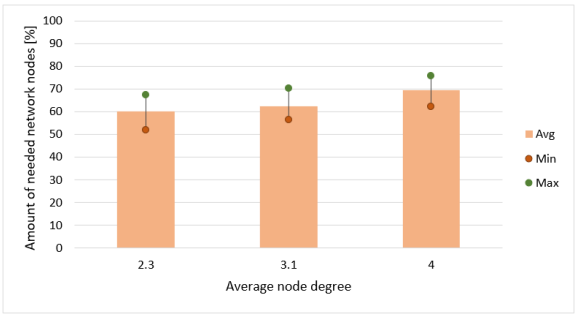

Figure 3: Nodes for monitoring per average node degree.

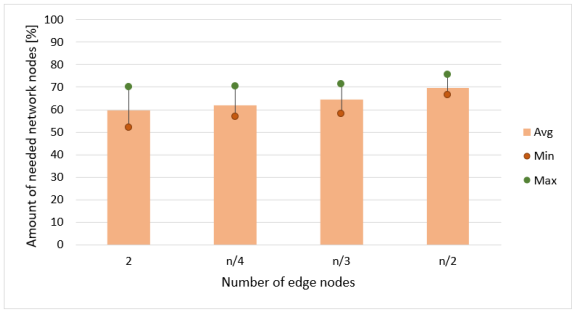

Figure 4: Nodes for monitoring per number of edge nodes.

The packet loss calculations were done on the same topologies as the testing of *critical nodes* selection. Ten scenarios of traffic flows for each of the networks were created and the bandwidth usage and potential packet loss was calculated. For each scenario different set of source-destination pairs for traffic flow was used and different amount of data was transferred, each of these generated randomly. The results of the testing scenarios for every testing topology are shown in Figure 5. "Calculated loss" represents the exact value of packet loss calculated by our model based on information from monitored *critical nodes* and network topology. "Estimated loss" represents an estimation of packet loss with lower and upper bounds. The average knowledge about the bandwidth usage and packet

Figure 5: Packet loss calculation.

loss in the network was 86.7%. We didn't find any correlation with specific network type, network size or number of edge nodes in the network.

For every testing scenario used for selection of *critical nodes* we created an SDN topology with Floodlight controller using OpenFlow 1.3. We simulated monitoring of interfaces on each network node and compared the results with monitoring of selected *critical nodes*. Figure 6 shows the results per testing topology. It is obvious that the amount of artificial traffic is not dependent on the size of the network since the average amount of monitoring traffic is lowest in the biggest topology. The test results grouped by an average node degree of the network, depicted in Figure 7, show that the growing density of the network results in a higher percentage of needed artificial traffic. Figure 8 shows the results of testing grouped by the number of edge nodes in the network. As expected, the amount of artificial traffic needed for monitoring grows with the number of edge nodes in the network.

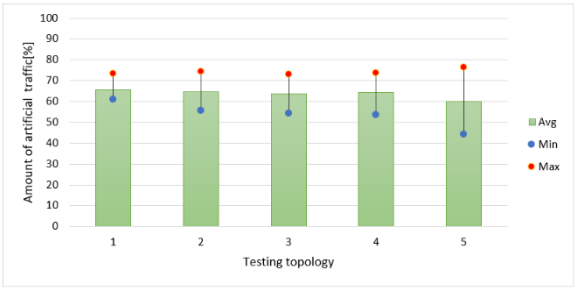

Figure 6: Artificial traffic per topology.

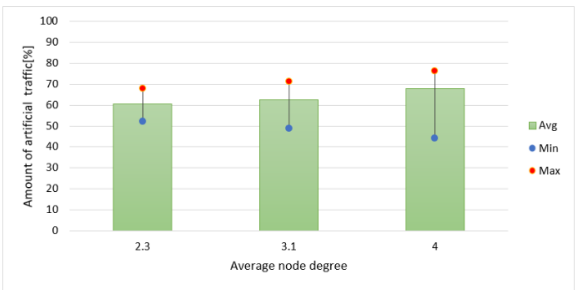

Figure 7: Artificial traffic per average node degree.

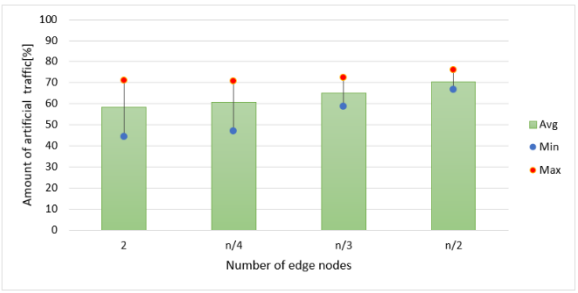

Figure 8: Artificial traffic per number of edge nodes.

5. Conclusions

The goal of our work was to design a novel network monitoring model to lower the monitoring overhead in the network while getting the overall knowledge of the network performance parameters. The proposed monitoring model analyzes the network, relationships among its elements and selects the most important elements. The selection is based on multi-objective optimization problem solving. The main objective was to cover all links in the network while positioning the monitoring probe at as few nodes as possible. The network monitoring may be then applied on much smaller number of nodes and links which will, in turn, decrease the amount of time and resources used.

The functionality of proposed model was verified in simulation environment of Matlab R2018b using 60 different network topologies. The results show that the number of nodes needed to be monitored ranges between 50-86.7% of all nodes in the network. The exact number is dependent of the density of the network and on the number of *edge nodes*. The possibility to calculate the remaining information about non-monitored elements was tested on ten different test scenarios for each of testing topologies. The average amount of calculated knowledge about the bandwidth usage and packet loss in the network was 86.7% while the remaining 13.3% were correctly estimated with lower and upper boundaries set. The amount of needed artificial monitoring traffic ranges from 42.6% up to 84.3%. The bandwidth consumption is therefore lowered by 15.7% - 57.4% which can be considered as a significant reduction of bandwidth consumption.

Acknowledgements. This research was supported by the Ministry of Education, Science, Research and Sport of the Slovak Republic, Incentives for Research and Development, Grant No.: 2018/14427:1-26C0. It was also partially supported by the Slovak Cultural and Educational Grant Agency (KEGA 011STU-4/2017) and the Slovak Science Grant Agency (VEGA 1/0676/12).

References

[1] Sihyung Lee, Kyriaki Levanti, and Hyong S. Kim. Network monitoring: Present and future. *Computer Networks*, 65, 2014.

[2] M. Hrubý. Optimization of network traffic flow. dissertation thesis. slovak university of technology in bratislava. 2013.

[3] Software-defined networking (sdn) definition. [online]. available at: https://www.opennetworking.org/sdn-definition/.

[4] W. Zhang, X. Zhang, H. Shi, and L. Zhou. An efficient latency monitoring scheme in software defined networks. *Future Generation Computer Systems*, 83, 2018.

[5] W. Queiroz, M. A. Capretz, and M. Dantas. An approach for sdn traffic monitoring based on big data techniques. *Journal of Network and Computer Applications*, 131, 2019.

[6] Z. Yu, Y. Zhang, Y. Zhu, D. Zhu, and P. Lin. Performance monitoring nodes deployment strategies for power wireless private networks based on improved mixed greedy algorithm. *IEEE International Conference on Energy Internet (ICEI)*, 2018.

[7] R. Umair, K. Shahid, and R. L. Olsen. Information reliability in smart grid scenario over imperfect communication networks using iec-61850 mms. *IEEE EUROCON 2017 - 17th International Conference on Smart Technologies*, 2017.

[8] Solarwinds network performance monitor. [online] available at: http://www.solarwinds.com/solutions/network-availability-monitor.aspx.

[9] Nagios. [online] available at: https://www.nagios.com.

[10] Total network monitor. [online] available at: http://www.softinventive.com/total-network-monitor.

[11] Zabbix: The enterprise-class open source network monitoring solution. [online] available at: https://www.zabbix.com.

[12] Paessler prtg network monitor. [online] available at: https://www.paessler.com/prtg.

[13] R. Khan and S. U. Khan. Design and implementation of an automated network monitoring and reporting back system. *Journal of Industrial Information Integration*, 9, 2018.

[14] M. Ljubojevic, A. Bajic, and D. Mijic. Centralized monitoring of computer networks using zenoss open source platform. *17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, 2018.

[15] P. Zhang, H. Jin, Z. He, H. Leung, W. Song, and Y. Jiang. Igs-wbsrm: A time-aware web service qos monitoring approach in dynamic environments. *Information and Software Technology*, 96, 2018.

[16] A. Villegas, P. Perez, J. J. Ruiz, and J. Lopez-Poncela. Scalable monitoring of end-to-end delay in live video services. *IEEE International Conference on Consumer Electronics (ICCE)*, 2018.

[17] V. Demianiuk, S. Gorinsky, S. Nikolenko, and K. Kogan. Robust distributed monitoring of traffic flows. *IEEE 27th International Conference on Network Protocols (ICNP)*, 2019.

[18] K. Watabe, S. Hirakawa, and K. Nakagawa. Accurate delay measurement for parallel monitoring of probe flows. *13th International Conference on Network and Service Management (CNSM)*, 2017.

[19] K. Watabe, N. Murai, S. Hirakawa, and K. Nakagawa. Accurate measurement technique of packet loss rate in parallel flow monitoring. *28th International Conference on Computer Communication and Networks (ICCCN)*, 2019.

[20] P. Megyesi, A. Botta, G. Aceto, A. Pescapé, and S. Molnár. Challenges and solution for measuring available bandwidth in software defined networks. *Computer Communications*, 99, 2017.

[21] X. Zhang, Y. Wang, J. Zhang, L. Wang, and Y. Zhao. A two-way link loss measurement approach for software-defined networks. *IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, 2017.

[22] J. Suárez-Varela and P. Barlet-Ros. Flow monitoring in software-defined networks: Finding the accuracy/performance tradeoffs. *Computer Networks*, 135, 2018.

[23] Z. Yang and K. L. Yeung. Flow monitoring scheme design in sdn. *Computer Networks*, 167, 2020.

[24] G. Tangari, D. Tuncer, M. Charalambides, Y. Qi, and G. Pavlou. Self-adaptive decentralized monitoring in software-defined networks. *IEEE Transactions on Network and Service Management*, 15(4), 2018.

[25] H. Tahaei, R. B. Salleh, M. F. A. Razak, K. Ko, and N. B. Anuar. Cost effective network flow measurement for software defined networks: A distributed controller scenario. *IEEE Access*, 6, 2018.

[26] intlinprog, mixed-integer linear programming (milp). [online] available at: https://www.mathworks.com/help/optim/ug/intlinprog.html.

Selected Papers by the Author

I. Hucková, P. Čičák. Advanced network monitoring using the selection of critical nodes. In *TSP 2019: 42nd International conference on telecommunications and signal processing*, pages 290–293, Budapest, Hungary. July 1–3, 2019. IEEE.

I. Hucková, M. Hrubý. QoS-Based optimization of data flow in MPLS networks. In *SAMI 2015. IEEE 13th international symposium on applied machine intelligence and informatics*, pages 83–88, Herľany, Slovakia. January 22–24, 2015. IEEE.

I. Hucková, P. Čičák. Advanced network monitoring and performance prediction. In *IDS 2015: 10th International Doctoral Seminar Proceedings*, pages 53–57, Varaždin, Croatia. September 24, 2015. Faculty of Organization and Informatics, University of Zagreb.

Instructions to the authors

Publishing procedure

All contributions are web-published. A contribution is published without unnecessary delay right after it has been accepted. Contributions are published on the fly in the current issue. It is at the discretion of the Editor-in-chief to determine, when the current issue is closed and a subsequent new one is open. There will be at least two issues in a year but it is left up to the Editor-in-chief to adjust periodicity of the Bulletin to actual needs.

Extended abstracts of theses is the primary type of article in the Bulletin. Each extended abstract will by annotated by identifying the thesis supervisor, who must recommend it for publication and stands for the Editorial Board in a role similar to a reviewer. We offer publishing extended abstracts on the Bulletin's web before the thesis is defended. This preliminary publishing is a specific service to the academic community. As soon as we learn about successful defence, the extended abstract gains the status of accepted paper and will be included in the forthcoming issue. The accepted paper will be annotated with the date of successful defence and name of the insitution where the defence took place.

It is the policy of the Bulletin to offer a free access to all its articles on the web. Moreover, the publisher will seek opportunities to promote as wide as possible access and/or indexing of the articles. All the past issues remain accessible on the web as part of the web portal of the Bulletin. Closed issues will be made available also in a printable form, free for downloading and printing by anyone interested.

Policy on Originality

It is the policy of the Bulletin that Slovak University of Technology be the sole, original publisher of articles. Manuscripts that have been submitted simultaneously to other magazines, journals or to conferences, symposia, or workshops without the prior written consent of the Editor-in-Chief will be rejected outright and will not be reconsidered. Publication of expanded versions of papers that have been disseminated via proceedings or newsletters is permitted only if the Editor-in-Chief judges that there is significant additional benefit to be gained from journal publication. A conference chairperson can arrange with the Editor-in-Chief to publish selected papers from conferences, symposia, and workshops, after suitable reviewing. The papers must meet the editorial requirements for research articles. Acknowledgement of the originating conference will appear as a credit when the paper is published in the Bulletin.

Manuscript information for extended abstracts of doctoral dissertations

All contributions are submitted electronically. Send your manuscript as LaTeXsources and .pdf files by e-mail to `editor.acm@fiit.stuba.sk`. Paper's length should be 6-12 pages. Please, use LaTeXstyle, which is available to download at bulletin web-page `http://slovakia.acm.org/bulletin/`.

Some remarks to the provided style:

- **Headings and Abstract**
  The heading must contain the title, full name, and address of the author(s), thesis supervisor, abstract of about 100-200 words.
- **Categories and Subject Descriptors**
  Define category and subject descriptors according to ACM Computing Classification System (see `http://www.acm.org/about/class/1998/`).
- **Keywords**
  Please specify 5 to 10 keywords.
- **Equations**
  You may want to display math equations in three distinct styles: inline, numbered or non-numbered display (we recommend the numbered style). Please make sure that your equations are clearly formulated and described.
- **Figures and tables**
  Figures and tables cannot be split across pages, the best placement for them is typically the top or the bottom of the page nearest their initial cite. To ensure this proper "floating" placement of figures/table, use the environment figure/table to enclose the figure and its caption.
- **References**
  Please use BibTeX to automatically produce the bibliography. If possible use abbreviations like: Proceedings – Proc., International – Int., Conference – Conf., Journal – J.
- **Selected papers by the author**
  This section is used for thesis extended abstracts. Please write down all publications which are related to your thesis.

Published by Slovak University of Technology Press,
Vazovova 5, 812 43 Bratislava, IČO: 00397687
on behalf of the ACM Slovakia Chapter
ISSN 1338-1237 (printed edition)
ISSN 1338-6654 (online)
Registration number: MK SR EV 3929/09