IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 27, No. 1, February, 2022, pp. 135-141
Journal homepage: http://iieta.org/journals/isi

Transmission Control Protocol Analysis Using NS3

Ahmed B. Abdulkareem

University of Anbar, Continuous Learning Centre, Ramadi 31001, Iraq

Corresponding Author Email: Ahmedalnakep3@uoanbar.edu.iq

https://doi.org/10.18280/isi.270116

Received: 15 October 2021
Accepted: 22 February 2022

Keywords:
TCP, active queue management, network simulator, NS-3, NS-2, end to end delays

ABSTRACT

The NS-3 Test Framework provides a focus on multidisciplinary development and high-level design, and is now being used by various experts around the world. Surprisingly, its Data Collection Protocol (DCP) implementation is late and is not planned to be used as a reference point for Transmission Control Protocol (TCP), focused research, where the Herald has provided NS-3 to accept NS-2 submissions. The latest-best configuration consists of a number of PCs connected to each other by two sweets and switches. As a result of the overlapping character, being a significant number of PCs, channels can create a complex problem. If a case of sensitivity is sent to the TCP bundles, each word flow should be taken with the appropriate treatment when there is a deadline. Drop-tail is a common scheme for using the power of measurement, and with this provision, the feeling cannot be limited to the illumination of long lines, which increases the delay in these methods. As a result, the flow efficiency decreases. The Board Active Queue Management (AQM) is largely oblivious to the fact that conspiracy and sensitivity are not the basic mechanisms that must be considered in solving this major problem. Another area to look for is depression, and the range of serious problems. In this work, a framework was developed based on drop- tail and different active queue management schemes such as DLP, Drop-Tail delivers, Packet First in First Out (PFIFO), Random Early Detection (RED) and Code Exploration Introduction. The evaluation is performed using an open-framework framework (NS-3) assessment framework. In the end, the results of the review suggest that Code and RED, a unique line of management figures, have shown much needed performance.

1. INTRODUCTION

The size and detailed design of the components of the current system are increasing rapidly. Most importantly, that the latest manufacturing industries with a gadget engine needs more clarity and better performance [1]. To meet these needs, a flexible study of compromise and communication between different domains has been prepared. Courses on Ethernet or Persistent Ethernet (RTE) have been improved [2]. RTE meetings include Ether CAD, Xerox, Power Link, Project IRT and Mudbugs. The RTE conference is a standard adopted by the Modicums Association [3]. Mudbugs interface is often used for communication between lover's devices, for example, downloadable systems (PLC) and micro-sized objects (MCUs), PC-type computer gadgets (CNC), and remote data controls. Acquisition Properties (SCADA). The SCADA framework is used to consider complex systems that may affect risk or malfunction. Along these lines, Mudbugs / TCP meet the requirements to ensure maximum reliability [4]. The most frustrating number of points hosted in Mudbugs / TCP programs is the number of IP addresses sent within the allowed range as a result, we do not know about the worst organizational points if the merger is not good [5]. This paper presents configuration data in Mudbugs / DCP and Framework Testing Framework 3 (NS-3) and shows how to create a Mudbugs / DCP interface with NS-3 [6]. In a given case, we simulate the Mudbugs / TCP scenario and measure the function of documents, for example, the number of focal points, topography, differences and response times based on the tic as shown in the Figure 1. To demonstrate the introduction, we use a literature search program called Diacritic [7]. With these deliberate frameworks, it is possible to determine which program is compatible with Mudbugs / TCP and when the building collapses in communications [8]. The Figure 1 shows the architecture and layer of TCP/IP and OSI Model IOT model [9].

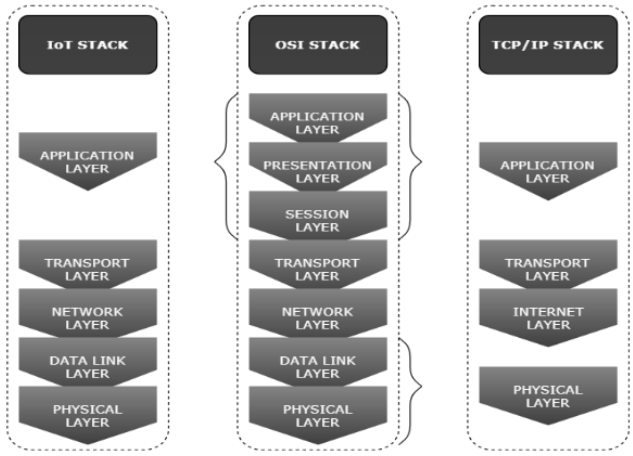


Figure 1. Layes of OSI model and TCP/IP model IOT model [10]

The significance of this research for versions makes the practical evaluation a challenge as well as the interpretation of results, because the NS-3 and TCP implementation significantly evolved in the meantime. In addition, the goal of this research is to allow researchers develop and evaluate new features of TCP by using our simulator in a much faster way than they would with a NS3 implementation.

2. BACKGROUND AND RELATED WORK

In this section, we begin by summarizing the domain of the TCP compilation control documentation and the TCP extensions for measuring windows and timelines. After that, we present an assessment of related activities.

2.1 Background for TCP

In RFC 793 the 16 bit is specified as the header field for the TCP clock. Thus, the largest representation window is therefore 65535 bytes: This restriction may be problematic for high latency, delay information channels as it restricts the capacity for these communication [10]. To avoid this problem, RFC 1323 introduced a window-like extension. This extends the TCP Windows definition to 32 bits, using the 32-map enlargement feature of the Windows 16 service field. This calibration item is used in the new TCP window selection process, set only in SYN stages: therefore, the window scaling feature is adjusted for each location when the link is opened. Other extensions brought by the Timestamp option [11]. It describes a method that allows each component (including transactions) to be performed at the lowest computational cost. In this way, time cycle calculation (RTT) and time transfer (RTO) calculation can be very accurate, which is often a necessary component of TCP efficiency. Timers also have other uses, such as protecting serial numbers from folding [12-14]. In addition to the options, the researchers also developed a way to control the initial TCP encounter. The communication is applicable when use the high-speed communication (Ex. satellites).

2.2 High speed TCP

TCP Highspeed is built by wide interconnect windows for high voltage channels or, more generally, TCP connections. In addition, cWnd highspeed accelerates test measures in relation to the normal TCP and accelerates cWnd recovery in loss.

This behaviour only happens when the window reaches a certain level, meaning that TCP High-speed is nice to the normal TCP in a heavily populated setting, without causing new congestion risks [15].

2.3 TCP Bick

Variation of TCP bike Reducing RTT damage (e.g., disadvantages between streams, with different RTTs, striving for a single blocking link) can be classified as high speed TCP. In particular, the congestion control problem with TCP bid is considered a search problem [16]. The binary search approach will be used to update the cWnd value (returned as cWndmin) of the current Windows value as the start and the last Windows cWndmax value (e.g. the cWnd value before the crash event) as the target point. When the gap is too wide from the current period, using the same growth strategy between these. Therefore, when the time of loss, stroke and windows association approaches cWndmaxuntil, the difference between cWndmaxand cWnd is below the predetermined limit. Once such a value is reached (or high windows are unknown, e.g. binary search does not enable at all) the algorithm changes to look for the new top window with the "high startup" scheme. If these two steps are damaged, the current window (before damage) is considered to be the new upper and then the reduced (double) window size will then be regarded as the new minimum.

2.4 TCP Cubic

TCP Cubic is an agreement that maintains the BIC's market potential and its strength. The main feature is that the window expansion function is punished in real time to be independent of RTT. Further, Cubic convex window is calculated by the function C (t-K) 3 + cWndmax, where C is a factor, Period (Time) elapsed since the last window is decreased, cWndmaxis window size is the final decrease of win3 cWndmaxβdow, and K=C, sufficient substance for permanent deterioration. Its goal is to overcome the problems of standard saturation control algorithms and optimize output over multiple BDP channels. Noordwijk replaces the standard "window-based" transmission with "basic" transmission [17]. The packet explosion distribution is divided into two categories: the size of the explosion, the number of components to be shipped during the shoot, and the duration of the explosion transmission, which is the time between the two deliveries. Both of these variables are updated based on ACK-based measurements, e.g. ACK amplitude and RTT variability. It also has some drawbacks: it operates under the hood of a controlled environment with known features, and does not guarantee the right behavior with competitive streams, and active resource sharing in the case of short transfers. Also, the use of its line (with outgoing delays) is very large [18]. On the other hand, the TCP services on a per connection basis such as in-order delivery. Ordering is indeed unnecessary when downloading an archive, because head-of-line blocking may slow down the connection.

2.5 TCP Tahoe

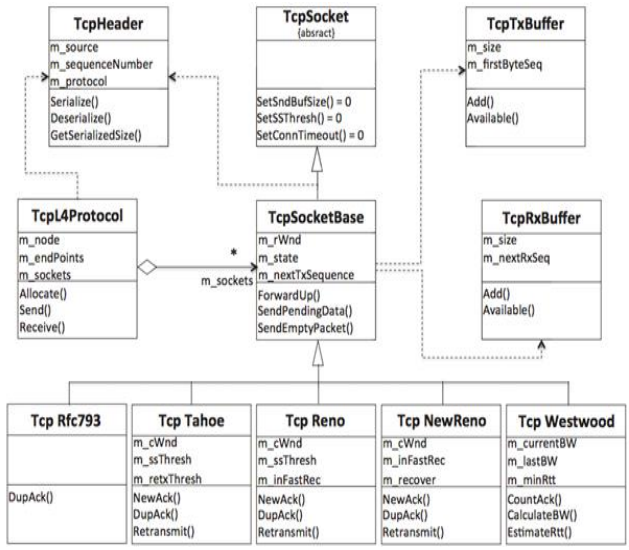


Figure 2. TCP class diagram

TCP Tahoe is the first version of TCP, an integration control system developed by von Jacobson with three advanced algorithms: slow start, focus block and fast distribution. With a TCP (sender) source, the idea is to routinely track the route through the ink as shown in the Figure 2, without knowing the state of the network [19].

2.6 Related work

Practical TCP integration control methods can be distinguished from risk-based control methods and delay-based control methods. Lost loss control coordinates (such as BIC-TCP or TCP-New Reno) use packet loss as a connection signal [20]. Delay-based compression controls (such as Vegas-DCP) use network delays to indicate congestion. Hardik investigated the impact of the original PBR and fixed-flow policies on the various risk factors used by the risk factors CCA and Cubic. Continuing with RTD, compared to or compared to Cubic, shows that PPR can achieve better performance than most CCAs proposed on the network. In this study, we compare the performance of BIC and BBR and analyze the four IB regions in the experiment [21]. Goal-based loss control or delay-based delay favors less RTD travel, but PPR offers more resistance compared to less-effective RTD. In-depth analysis reveals the cause of the PPR dependence of the slow flow of RTD.

The PBR wavelength energy is supported by the model and shows a closed melatonin form for the PBR RTD integrity of the PBR. In opposition to the authors above, we explain the reason for the logical problem of PBR and other TCP policies that arise from the merger control policy. While AQM modes share the common goal of reducing TCP congestion, the most effective user control is when AQM and TCP work together. This experiment is analyzing the performance of different pairs of TCP and AQM algorithms [22]. The experiment extended towards the confirming that Codel provided better performance of the equations with higher ordering violence. The authors used two PBR streams that compete on the same trajectory with different RED parameters and enable short RD routing to return short bandwidth role. To the best of our knowledge, we are the first to evaluate BBR in accordance with various AQM methods. We found that the PBR and Codel pair could achieve better performance [23].

Much research work has been done on developing methods and methods for statistical control over similarities, which is a summary of such research under Table 1.

3. PROPOSED METHODOLOGY

The NS3 (Network Simulator 3) it's an event network simulator that uses simulation architectures and models in C ++ and also Python. This is a free program, designed primarily for research and educational use, that runs in a Linux environment [24]. Ns-3 software company was built with base units. Main unit contains some basic code. E.g., smart directions, organizers and more. This unit communicates with the network unit, which includes (code, packet, address types (IPv3and Mac) and network device types).

3.1 TCP socket communication with the interface via the system layer

The following is an analysis of TCP socket's general appearance, and how it can be used to communicate with the socket. Analysis of socket foot chassis is also carried out. For clarification, please notice that we will use the words (sender and receiver) to specifically define the socket feature as shown in the Figure 3. However, these two functions should be used concurrently in DCP (i.e. socket sender and recipient concurrently): our discrepancies are limitless, as the same definition refers to all sites because the entire is complete [25].

Table 1. Summary of related works

Authors	Main objectives	Scenario	Constraints
[26]	The performance improvement mechanism for TCP in NS-3.	Cutting payload (CP) and congestion control.	TCP protocol is not use the actual channel bandwidth and congestion occurrence during the data transmission.
[27]	To enhance networks running multiple TCP variants, parameters of any of the TCP variants can influence local stability.	Simulator (NS3) exhibit limit cycles in the queue size.	We analyze local stability, and performance of TCP/NC coexistence.
[28]	Focused on compound TCP as it is the default protocol in the Windows operating system and started by conducting a local stability analysis for the underlying fluid models.	Analyzed a model for transport control protocol (TCP).	Evaluates the performance using TCP.
[29]	To improve performance of TCP over protocol networks.	Analyzed a model for user defined protocol (UDP).	Found that MX-TCP is more appropriate for lossy-links such as driving on highway or low bandwidth and accessing public hotspots or Internet.
[29]	To Provide the omultipath TCP implementation using NS-3 model.	Used the Network Traffic Congestion Control scheme, called SB-CC, which leverages ECN (Explicit Congestion Notification) mechanism to detect shared bottlenecks among subflows.	The multipath network implementation and estimate the congestion degree of each subflow.

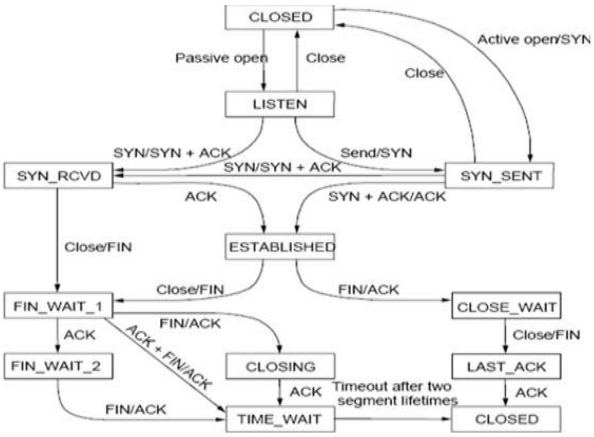


Figure 3. TCP step machine [25]

3.2 Modbus implementation on NS-3

In this section, we will focus more on defining flow and source relationships instead of source definition because all resources are written in the popular C ++ programming language. We create and modify bus-level, star-level, Modbus-Assist, Modbus-PT, 'Modbus-Client', 'Modbus-Server', 'Modbus-MGR' and 'MPAP'. The organizational framework of these groups is. For lower layer model, here is the entire phase of the whole sub-plane and the upper(highest) layer. All functions in the Modbus / TCP protocol are designed for the sample phase. This means that Modus runs on the C ++ language available in this class, including data model resources, implementation code, data and data-parsing functions. Modbus client and server program that sends a request or response message to this class. Modbus introduces a client and server application for environments created in the context class. It is also worthwhile to use the model phase functions [30]. In fact, object models and helpers have a certain amount of IP address, work code, start address, and number of locations. They were used to create the Modbus / TCP data structure if it had some value. The Position class is the phise which converts a particular value from the operator of the helper class. Display source code for the 'Modbus / TCP' protocol management. we use NS-3 simple modules like 'core', 'network', 'internet', 'contractor' and 'implementation'. With these modules, you can create network topology, data rate, and assign an IP address, location and usage. After setting a specific topology, date rate, number of nodes, and IP addresses in the status class, Mbbus uses Modbus-Assist and Model models to send queries and response messages to specific locations [30].

3.3 Setup and configuration

The setups of the NS-3 model towards the implementation of TCP model are inherited from common header class. The inherited class is stored in the src/network directory. The code from this directory is swap out with the minimal implementation of the script. Two important abstract classes are supported form the setup and implementation like class TcpSocket and class TcpSocketFactory [19]. The TcpSocket class is configured with the feature src / internet / model / tcp-socket. {cc, h}. The existing class hosted the attributed of the implemented class. InitialCwnd object parameter is used for the implementation of this class. The TCP socket implementation using the four layer protocol is done using the TcpSocketFactory class. The TCP model in NS-3 is implemented using native implementation and simulation based model. For the implementation of TCP the virtual machine is combining diverse ways. The NS-3 provides the port of TCP towards the setup and implementation. The TCP is implemented with the help of GTNets. The full model of TCP is the bidirectional and attempt for connection setup and logic. The setup of this model framework is done using the following files of code.

```
src/internet/model/tcp-header.{cc,h}
src/internet/model/tcp-l4-protocol.{cc,h}
src/internet/model/tcp-socket-factory-impl.{cc,h}
src/internet/model/tcp-socket-base.{cc,h}

rc/internet/model/tcp-tx-buffer.{cc,h}
rc/internet/model/tcp-rx-buffer.{cc,h}
rc/internet/model/tcp-rfc793.{cc,h}
rc/internet/model/tcp-tahoe.{cc,h}
rc/internet/model/tcp-reno.{cc,h}
rc/internet/model/tcp-newreno.{cc,h}
rc/internet/model/rtt-estimator.{cc,h}
rc/network/model/sequence-number.{cc,h}
```

The various TCP congestion control are supported for implementation of subclass for the TcpSocketBase. Tahoe, Reno, and NewReno. NewReno is used by default manner as a subclass for the setup.

The various cases the set up application layer in ns-3 application are used the socket function like src/applications/helper and src/network/helper [31].

```
// Create a packet sink on the star "hub" to receive these packets
uint16_t port = 50000;
Address sinkLocalAddress(InetSocketAddress (Ipv4Address::GetAny (), port));
PacketSinkHelper sinkHelper ("ns3::TcpSocketFactory", sinkLocalAddress);
ApplicationContainer sinkApp = sinkHelper.Install (serverNode);
sinkApp.Start (Seconds (1.0));
sinkApp.Stop (Seconds (10.0));

// Create the OnOff applications to send TCP to the server
OnOffHelper clientHelper ("ns3::TcpSocketFactory", Address ());
```

For the configuration of this framework below snippet is used. This source configures the traffic source of TCP. For the configuration of the TCP number of parameter are used which are exported using Attribute class for the socket. The internet objects are created related to socket stack. These created objects are put on the top statement for the simulation program.

```
Config::SetDefault ("ns3::TcpL4Protocol::SocketType", StringValue
("ns3::TcpTahoe"));
```

The pointer of the actual socket is utilized by setting the socket option. The method Bind is used for the actual socket setup. The Socket CreateSocket method is used for the socket configuration [24].

```
// Create and bind the socket...
TypeId tid = TypeId::LookupByName ("ns3::TcpTahoe");
Config::Set ("/NodeList/*/$ns3::TcpL4Protocol/SocketType", TypeIdValue (tid));
Ptr<Socket> localSocket =
  Socket::CreateSocket (n0n1.Get (0), TcpSocketFactory::GetTypeId ());
```

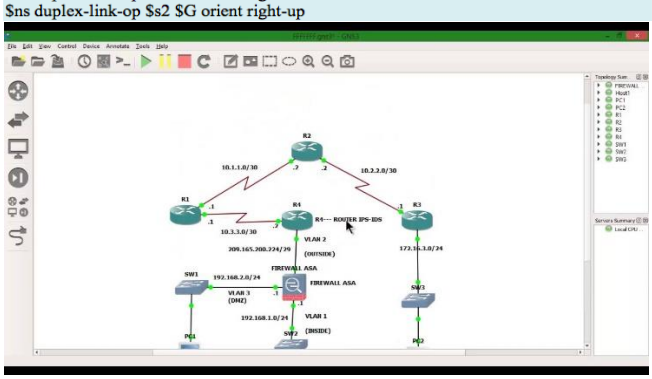
In particularly, the describe our motivation and the technical aspects of our implementation, present a few tools that developed to ease testing and analysis of related TCP

traces. The evaluation and challenge, hence tried to choose tools that would allow for seamless testing and analysis between the kernel and NS-3 stacks to lighten the burden analysis.

3.4 Topology layout

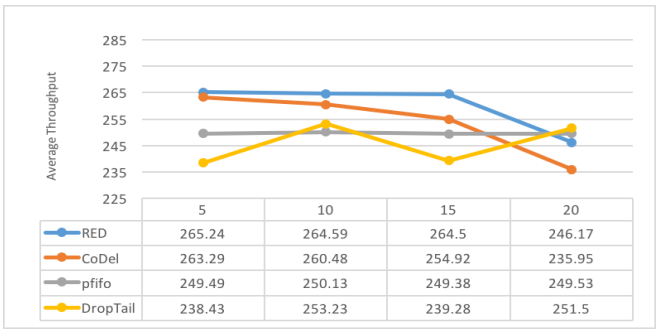
The simulation scenario of ns3 for successful implementation there needs a layout of network topology. This layout topology comprises of collection of link and nodes [31]. The code snippet for the development of the topology layout using the nodes is shown below.

```
#Define the layout of the topology
$ns duplex-link-op $s0 $G orient right-down
$ns duplex-link-op $s1 $G orient right
$ns duplex-link-op $s2 $G orient right-up
$ns duplex-link-op $G $r orient right
$ns duplex-link-op $s1 $G orient right
$ns duplex-link-op $s2 $G orient right-up
```



4. RESULT AND DISCUSSION

Throughput: The aggregate throughput is regarded as the total amount of throughput that was achieved during the simulation of network. Thus, the through put of each flow is aggregated so as to evaluate them as shown in the Figure 4.

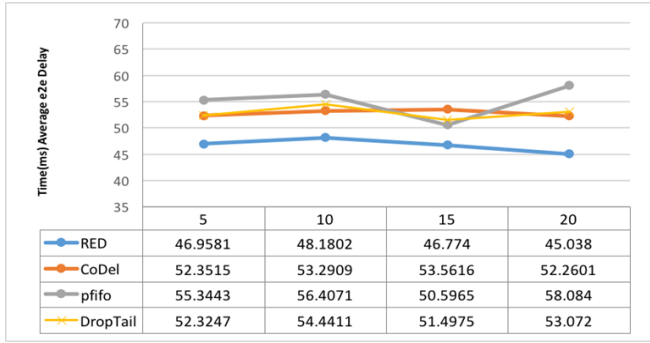


	5	10	15	20
RED	265.24	264.59	264.5	246.17
CoDel	263.29	260.48	254.92	235.95
pfifo	249.49	250.13	249.38	249.53
DropTail	238.43	253.23	239.28	251.5

Figure 4. The throughput of the scenario

When the four schemes were compared, it was found that the highest throughput was achieved by CoDel, while that of the RED is still approximately the same as the previous one. The worst aggregate is demonstrated by the pfifo and Droptail. Even though, the through of DropTail is good, it is not considered as good enough.

End to end delay: which is also referred to as latency, is reflective of aggregate of delay over transmissions from source to destination. The end-to-end delay demonstrated by the CoDel is very minimal, and as such considered as one of the advantages of CoDel. The same end-to-end delay is demonstrated by the RED.



	5	10	15	20
RED	46.9581	48.1802	46.774	45.038
CoDel	52.3515	53.2909	53.5616	52.2601
pfifo	55.3443	56.4071	50.5965	58.084
DropTail	52.3247	54.4411	51.4975	53.072

Figure 5. End to end delays

On the other hand, the latency exhibited by both Fifi and Drop Tail is high, and therefore not suitable for emergency and high-speed network. Fifi's latency may be attributed to the management of priority of the packets (the processing delay), as shown in the Figure 5. That of Drop tail can be attributed to high congestion over the channel.

Finally, we present in the following a few simulations to compare the Linux kernel implementation to our NS-3 implementation. In order to minimize the differences due to the environment and for ease of reproducibility, we chose to compare the through put and the end-to-end delay for the Linux NS-3 implementations within the TCP framework. This means that nodes, roters and links are created by NS-3. That, every node can be configured with a specific network stack. We always Linux stacks in the routers.

5. CONCLUSIONS

The effect of node velocity and packet size on the TCP network is shown in this study. Test algorithms are evaluated for intermediate performance (TP) and end-delay (E2E) as a measure of performance using an NS3 simulator. In this simulation project there is a simulation model with various applications. Sequence projects and their production files are done. The use of these files is important for monitoring network performance in terms of delays, performance, and TCP. Simulation of Drop Tail, RED, Codel, and PFIFO designs were carried out with the help of a unique sim-NS3 event. The performance Results of 'RED' and 'CodeL' indicate that their performance is good and low in comparison with drop tail and PFIFO.

REFERENCES

[1] Gupta, R., Tanwar, S., Tyagi, S., Kumar, N. (2019). Tactile internet and its applications in 5G era: A comprehensive review. International Journal of Communication Systems, 32(14): e3981. https://doi.org/10.1002/dac.3981

[2] Babaeinejadsarookolaee, S., Birchfield, A., Christie, R.D., et al. (2019). The power grid library for benchmarking ac optimal power flow algorithms. arXiv preprint arXiv:1908.02788. https://arxiv.org/abs/1908.02788

[3] Vitturi, S., Zunino, C., Sauter, T. (2019). Industrial communication systems and their future challenges: Next-generation Ethernet, IIoT, and 5G. Proceedings of the IEEE, 107(6): 944-961.

https://doi.org/10.1109/JPROC.2019.2913443
[4] Felser, M., Rentschler, M., Kleineberg, O. (2019). Coexistence standardization of operation technology and information technology. Proceedings of the IEEE, 107(6): 962-976. https://doi.org/10.1109/JPROC.2019.2913443
[5] Chanu, A.D., Sharma, B. (2019). Detection of routing infrastructure attack in TCP connection. In International Conference on Computational Intelligence, Security and Internet of Things, pp. 123-131. https://doi.org/10.1007/978-981-15-3666-3_11
[6] Luo, J., Yang, X., Zhang, C. (2019). CCMA: A dynamical concurrent-connection management agent to mitigate TCP incast in datacenters. IEEE Access, 7: 63303-63320. https://doi.org/10.1109/ACCESS.2019.2917336
[7] Maji, S., Arora, S. (2019). Decision tree algorithms for prediction of heart disease. in Information and Communication Technology for Competitive Strategies, Springer, 447-454. https://doi.org/10.1007/978-981-13-0586-3_45
[8] Dab, B., Aitsaadi, N., Langar, R. (2019). Joint optimization of offloading and resource allocation scheme for mobile edge computing. In 2019 IEEE Wireless Communications and Networking Conference (WCNC), Marrakesh, Morocco, pp. 1-7. https://doi.org/10.1109/WCNC.2019.8885537
[9] Howser, G. (2020). The OSI seven layer model. In Computer Networks and the Internet, pp. 7-32.
[10] Chen, Y.R., Liu, I.H., Chang, K.H., Liu, C.G., Li, J.S. (2019). Selection strategy for VM migration method. Journal of Robotics, Networking and Artificial Life, 6(1): 66-70. https://doi.org/10.5954/ICAROB.2019.OS1-3
[11] So, A., Al-Sharif, L. (2019). Calculation of the elevator round-trip time under destination group control using offline batch allocations and real-time allocations. Journal of Building Engineering, 22: 549-561. https://doi.org/10.1016/j.jobe.2019.01.013
[12] Hagos, D.H., Engelstad, P.E., Yazid, A., Griwodz, C. (2019). A deep learning approach to dynamic passive RTT prediction model for TCP. In 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC), London, UK, pp. 1-10. https://doi.org/10.1109/IPCCC47392.2019.8958727
[13] Jiang, H., Dovrolis, C. (2002). Passive estimation of TCP round-trip times. ACM SIGCOMM Computer Communication Review, 32(3): 75-88. https://doi.org/10.1145/571697.571725
[14] García, M., Agüero, R., Muñoz, L. (2004). On the unsuitability of TCP RTO estimation over bursty error channels. In IFIP International Conference on Personal Wireless Communications, pp. 343-348. https://doi.org/10.1007/978-3-540-30199-8_28
[15] Sulaiman, S.O., Al-Dulaimi, G., Al Thamiry, H. (2019). Natural rivers longitudinal dispersion coefficient simulation using hybrid soft computing model. Proc. - Int. Conf. Dev. eSystems Eng. DeSE, pp. 280-283. https://doi.org/10.1109/DeSE.2018.00056
[16] Kumar, S., Andersen, M.P., Kim, H.S., Culler, D.E. (2020). Performant {TCP} for {Low-Power} wireless networks. In 17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20), pp. 911-932.
[17] Ali, A.M., Kadry, S. (2019). Performance evaluation of TCP congestion control algorithms using a network simulator. In Automatic Control, Mechatronics and Industrial Engineering: Proceedings of the International Conference on Automatic Control, Mechatronics and Industrial Engineering (ACMIE 2018), October 29-31, 2018, Suzhou, China, p. 317.
[18] Abdulrazzak, F., Abdulaziz, E., Al-Hussaini, K. (2019). Performance analysis for TCP protocols over mm wave in 5G cellular networks. In 2019 First International Conference of Intelligent Computing and Engineering (ICOICE), Hadhramout, Yemen, pp. 1-6. https://doi.org/10.1109/ICOICE48418.2019.9035145
[19] Lu, Y., Li, Y. (2019). TS-TCP: Two-stage congestion control algorithm for high concurrency TCPs in data center networks. In 2019 28th International Conference on Computer Communication and Networks (ICCCN), Valencia, Spain, pp. 1-9. https://doi.org/10.1109/ICCCN.2019.8847083
[20] Shurman, M., Taqieddin, E., Oudat, O., Al-Qurran, R., Al Nounou, A.A. (2019). Performance enhancement in 5G cellular networks using priorities in network slicing. In 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan, pp. 822-826. https://doi.org/10.1109/JEEIT.2019.8717469
[21] Oliveira, A.A., Batista, D., Hirata Jr, R. (2019). Exploring the NS-3 mmWave module.
[22] Wei, W., Xue, K., Han, J., Wei, D.S., Hong, P. (2020). Shared bottleneck-based congestion control and packet scheduling for multipath TCP. IEEE/ACM Transactions on Networking, 28(2): 653-666. https://doi.org/10.1109/TNET.2020.2970032
[23] Paliwal, G., Sharma, K.P., Taterh, S., Varshney, S. (2019). A new effective TCP-CC algorithm performance analysis (NS3). In 2019 4th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, pp. 630-635. https://doi.org/10.1109/ISCON47742.2019.9036301
[24] Tazaki, H., Nakamura, R., Sekiya, Y. (2015). Library Operating System with Mainline Linux Network Stack. Proceedings of Netdev.
[25] Supaporn, L., Le Jun, Z. (2017). Applying relief algorithm for feature selection in sentiment classification for movie reviews. J. Comput. Theor. Nanosci., 14(11): 5418-5423. http://dx.doi.org/10.1166/jctn.2017.6964
[26] Pokhrel, S.R., Singh, S. (2020). Compound-TCP performance for industry 4.0 WiFi: A cognitive federated learning approach. IEEE Transactions on Industrial Informatics, 17(3): 2143-2151. https://doi.org/10.1109/TII.2020.2985033
[27] Gamess, E., Smith, B., Francia III, G. (2020). Performance evaluation of modbus TCP in normal operation and under a distributed denial of service attack. International Journal of Computer Networks & Communications, 12(2): 1-21. https://doi.org/10.5121/ijcnc.2020.12201
[28] Bosch, P., Latré, S., Blondia, C. (2020). An analytical model for IEEE 802.11 with non-IEEE 802.11 interfering source. Computer Networks, 172: 107154. https://doi.org/10.1016/j.comnet.2020.107154
[29] Campanile, L., Gribaudo, M., Iacono, M., Marulli, F.,

Mastroianni, M. (2020). Computer network simulation with NS-3: A systematic literature review. Electronics, 9(2): 272. https://doi.org/10.3390/electronics9020272

[30] Gharamaleki, M.M., Babaie, S. (2020). A new distributed fault detection method for wireless sensor networks. IEEE Syst. J., 14(4): 4883-4890. https://doi.org/10.1109/JSYST.2020.2976827

[31] Saini, J.S., Sohi, B.S. (2020). Performance evaluation of interference aware topology power and flow control channel assignment algorithm. International Journal of Electrical and Computer Engineering, 10(3): 2503. http://doi.org/10.11591/ijece.v10i3.pp2503-2512