ESTIMATING END-TO-END DELAY ON A NETWORKING ENVIRONMENT USING A DEVELOPED FRAMEWORK

Ahmed Alsheikhy *

Department of Electrical Engineering, College of Engineering,
Northern Border University, Arar
Saudi Arabia

* Corresponding Author, e-mail: aalsheikhy@nbu.edu.sa

Abstract: Any designer wants to perform performance analysis to calculate and exploit performance, in this paper, the main aim is to compute end-to-end delay which also known as "latency" at beginning phase before start working on the implementation to ensure that the proposed design is efficient. This analysis is especially needed in real-time embedded systems and considered to be crucial. Herein, an implemented model, framework, is used to link between a Finite State Machine and a Mathematical Analytical method to compute the expected average end-to-end delay for different layers of abstractions on any network topology using a network simulator. A Hierarchical Generic FSM "HGFSM" was developed earlier to estimate the desired performance metric which is represented by the end-to-end latency in this paper. Several modifications and adjustments were made to the HGFSM and the mathematical model to be suitable for this research. Cisco Packet Tracer ver. 7.3 was used in this paper to show how that performance metric, which is latency, was computed using the implemented framework. In addition, MATLAB was used as a simulator tool along with Cisco Packet Tracer to show the effectiveness of using that framework in finding the desired metric.

Key words: end-to-end delay, Finite State Machine (FSM), Hierarchical Performance Model (HPM), Real-time embedded systems, Performance.

1. INTRODUCTION

Nowadays, embedded systems have become the core of our daily life. They can be found everywhere. Very often, any designer or a developer depends on an instrument to find a design that satisfies its requirements. Occasionally, this process consumes more time and also can be insufficient regarding to its outputs. As a result of using embedded systems, it becomes crucial to calculate end-to-end delay "latency" or any other desired metrics. For simplicity, the end-to-end delay is defined as a time needed for a packet to

be transmitted or sent over a network from its source to its destination. In this paper, OPENWRT and Cisco Packet Tracer are used to build a virtual environment which consists of several network topologies and networking devices. OPENWRT is an operating system which targets embedded devices and builds on a Linux platform; its role and service is to simulate network packets and their movements. In addition, it has become an essential component in routers which are being used in today's life.

In this paper, focusing for analysis of timing aspects in networking environment is considered [4, 5, 6, 13, 14, 17]. Specifically, the concerned team aims to decide which design provides less end-to-end delay and ensuring it guarantees its functions [4, 5, 13, 14]. Yet, the concerned team must propose a scheme to predict the needed performance metrics when there is no performance evaluation method and also that scheme is capable to spot weaknesses if possible. Using performance analytical method to estimate end-to-end latency from the first stage of design in the desired networking platform reveals well outcomes than using a technique called "fix-it-later" [4, 13, 14].

Presently, there are three mechanisms which are used to predict performance and analysis and can be classified as follows:

1. Simulation Based Method.
2. Analytical Based Method.
3. Direct Measurement [4, 5].

In this research, direct measurements and a mathematical analytical method are compact together to find the average value of delay.

Contribution in this paper is done by using the framework developed in [4] to compute end-to-end delay only. HGFSM is modified to be used for this purpose as several factors need to be included in this research. Moreover, mathematical model is adjusted too for the objective functions to be easily constructed. After that, objective functions are mixed with the performance parameters to compute the latency. Readers can refer to [4, 5] for more information. The developed framework was applied on different scenarios and circumstances using Cisco Packet Tracer as the tool along with MATLAB and presented within this paper to estimate the average latency. Lastly, finding actual end-to-end latency is conducted to compare the results between the actual and the estimated values.

Related work on the delay estimation approach is presented in the coming Section. In section 3, a complete explanation about the developed model in [4] and its associated hierarchical performance model to calculate the needed metric which is the "delay" in this research. After that, two cases that apply the implemented approach in [4] with the support of the simulation tool to evaluate the desired latency are shown in section 4. Lastly, the conclusion of the paper is presented in section 5.

2. RELATED WORK

G. Lentaris et al in [1] performed a tradeoff analysis for power and performance parameters on Vision-Based Navigation (VBN) platform. They used a set of benchmarks on image processing and computer vision algorithms on different scenarios and then collected the obtained results. Those results were considered based on connectivity, size/mass and lastly watt usage. This work involved a number of CPUs from small

devices to bigger ones like those put in embedded systems in aerospace machines. In addition, FPGA was used in their work to perform in-house evaluation. This work consumed time whereas the developed framework in [4] requires only paper and pencils to compute the needed performance metrics. Readers can refer to [4] and [5] for more information.

M. Sreenath and Dr. P. A. Vijaya [2] conducted performance evaluation on an embedded system using several scheduling algorithms to reduce the execution time during the stipulated or given time slot to each task. Their aim was to allocate sufficient processor time to existing tasks based on scheduling algorithms. They used TORSCHE scheduling toolbox in MATLAB to schedule several jobs on multi processors. The developed model in [4] is suitable to be used to evaluation end-to-end delay as it proved it in [6]. Furthermore, this model can be used in many applications such as smart devices radars to calculate the expected average latency. Currently, this framework is being utilized to find latency and energy dissipation in robotics and cars. Interested readers can refer to [4], [5] and [6].

S. Sundaresan et al in [7] measured the performance of user traffic in home wireless networks using passive measurement technique. That tool was deployed on product of numerous network devices to estimate several performance metrics after correlating with TCP segmentation. Those performance metrics were bitrate, retransmission rate and Received Signal Strength Indication (RSSI). More than 66 homes were used to collect data for a month. Their results indicated that 5GHZ band yields better performance than 2.4GHZ band and the performance of several hosts inside the similar network in a designated place has a significant variation. This paper measures the end-to-end latency as the desired performance metric using the analytical method. This method uses different level of abstractions to capture several performance parameters to find the estimated average performance metric.

In [9], G. Kremer et al developed a tool that allowed them to manage the noise altitude on a WIFI communication channel in a secure environment in side a room while that channel provided signal in one direction only. Their motivation was to estimate or predict the performance of wireless networks w.r.t. a physical channel condition since it is considered the source of the errors and performance drops through communication peers. Several medium conditions were generated to collect different data sets in order to estimate or predict the throughput using various PHY layer parameters on that link. Two algorithms were used to predictively estimate the throughput from measured physical medium, those algorithms were Support Vector Regression “SVR” and K-Nearest Neighbors “K-NN”. The purpose from using two schemes was to study the trade-off between the accuracy of their estimation and the complexity. In this paper, the average value of the end-to-end delay is estimated using the developed framework in [4] through different level of abstractions. The estimation procedures are computed using the analytical analysis with the help from the Cisco Packet Tracer simulator to estimate several performance parameters. Readers can obtain more information from [9].

L. Xue et al in [10] developed a network assessment tool, OMware, in order to increase the steadiness of a domestic broadband frequency. The developed ware worked by implementing the send and receive features for packets measuring in the kernel. Their main aspect was to provide stable throughput when compared a typical socket-based

measurement at a user level. A Netgear router was used to test the developed tool using two sets of experiments. The experiments were done with and without cross traffic. The cross traffic rate was 9000 packets per second with a packet size uniformly distributed between 100 to 1500 Bytes. More information can be found in [10]. The framework in this paper is capable of estimating the average end-to-end latency regarding the size of the packets being sent since their size values are included in the analysis.

Implementing a common FSM to use it in calculating performance for any system and especially in real-time embedded systems was presented in [4] and will be used in the coming section. Several methods for evaluating the delay at different levels such as gate-level and beyond were developed earlier. Numerous simulation tools were developed in order to achieve that goal. However, those approaches are impractical to evaluate the desired performance metric (delay) because of the absence of information readiness of both levels of a system being considered.

When HPM is being used, various factors are essential for each layer in the HPM; those factors circulate from down to up levels [4, 5, 6]. This paper uses the model implemented in [4] to find objective functions which will be used later for performance analysis in order to compute the average delay. Fig 1 from [4] shows an overall summary of developed framework which uses FSM and HPM to calculate the needed average latency.

Hierarchical Generic FSM. Also known as HGFSM, is developed, adjusted and then transformed into a well stochastic model which is also known as Markovian model. Every individual state in that stochastic representation is broken down to another graph which relates to the original Markovian model. Moreover, Hierarchical Performance Model is utilized and practiced in the entire levels for the sake of finding the performance equations using bottom-up procedures to estimate construct the objective functions or equations.

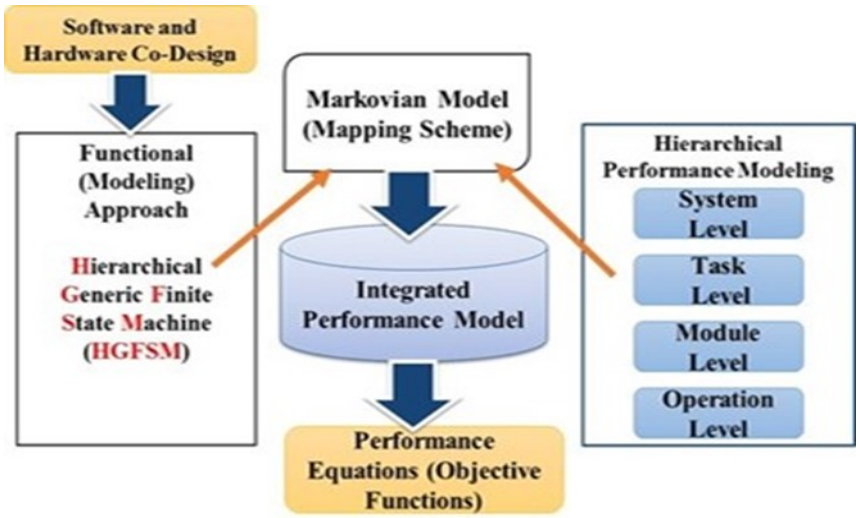

Fig. 1. Developed framework

3. FINITE STATE MACHINE AND HIERARCHICAL PERFORMANCE MODEL

Any common or classical Finite State Machine representation is consisted of 5-pillars which are $\{\sum, S, S_0, \delta, F\}$. $\sum$ refers to a set of input symbols. S denotes a state or a group of predefined states in the FSM. S_0 refers to an initial state or a group of initial statuses which are sub-elements of the original state or states. δ refers to a change or movement function where it links a current phase to a next one and F has a final state or a group of final ones which is/are part of S [4, 5, 15, 16]. A job/task in any structure of a system can be categorized as accomplished successfully or aborted for any reason and failed. In Fig. 2, hierarchical generic FSM is displayed. The closed loop in the initial state in Fig 2 indicates that a device is incapable to operate accurately because there is a defect or a fault or a glitch [4]. In this paper, that device can be either a switch, a router, a server or a client machine.

Every feed into states refers to a job or a group of jobs which is combined to build an applicable task. Flowing from a recent phase S_i to a next one S_j is symbolized by bold arrows in fig. 2 and this movement is happened due to some conditions or events inside the current state. For the HGFSM, S contains all 6 states [4] as displayed in Fig. 2 whereas S_0 holds the Initial state only. F accommodates Completed phase, that phase is denoted by two circles as shown in Fig 2.

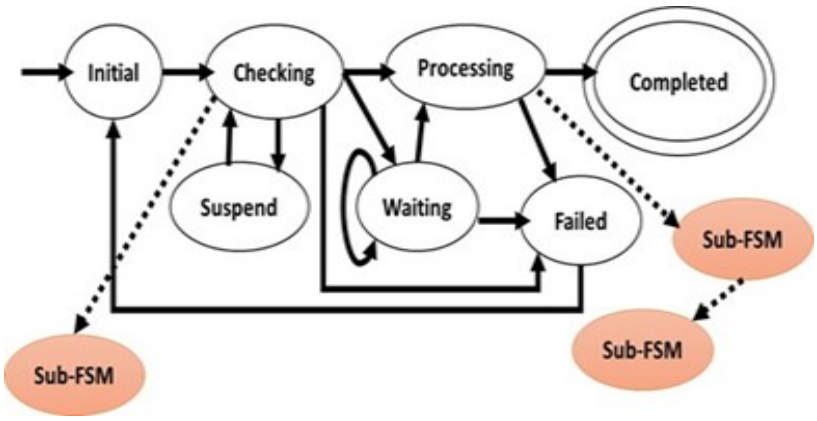

Fig. 2. Hierarchical Generic FSM

In fig. 2, the typical FSM is modified and adjusted to be the **H**ierarchical **G**eneric **F**inite **S**tate **M**achine which is abbreviated as HGFSM. The states are as follows:

The opening state, which is also known as initial one: all packets are delivered and acquired by the incoming terminal; the closed loop indicates that there is a malfunction or a defect or a bug which causes a network device to behavior improperly. If there is the malfunction, then the device goes into the closed loop which tells that the packets cannot be sent to the next phase. After the device or devices turned on, or connected to power sources, several initializations events take place such as generating the directory, constructing and arranging boot source files and deriving all needed Kernel registries and root files.

The Checking state: it determines a suitable path between a source and a destination in its directory. This path should be shortest one according to some predefined criteria in the device operating system. If the destination address is in the directory table, then, the packet is directed or forwarded either to the execution "processing" phase or the waiting one based on some circumstances or settings already defined in the devices such as the Processing Unit "P.U." is occupied or not and the precedency rank of the packet. The device decides the best path to the destination if the destination address is not found in the directory table. Readers can refer to [4] for more information.

The Execution state: denotes the residence where packets are sent to their destination addresses. If the operation is succeeded, then it directs the task/job into the done state which also refers to the completed one. Otherwise, it directs it to the unsuccessful state which is seen as failed one in Fig. 2. The state is broken down to form another two sub-FSMs as illustrated in Fig. 2.

In the Waiting phase, all packets wait their turn to be transferred into the next phase and this phase is seen as a medium for them. In the meantime, all packets are being examined regularly to determine which one should be moved into the next phase first.

The checking state is also decayed into another level of layer which holds 3 states while the execution state is decomposed to another two levels with 3 states in each one to form a hierarchical model. The implemented HGFSM in [4] contains 15 states which construct the desired stack layers of HPM as depicted in Fig. 3. Readers can get more information in [4, 5].

The HGFSM is converted to a Markov representation graph [4, 5]. This graph has its own 3-tuples which are {X, Y, T}, where X refers to a group of states already laid in the original HGFSM scheme. Y indicates a set of preliminary possibilities for all phases in the used approach whereas T denotes the transition probabilities matrix between all states. The conversion process is performed as follows:

1. Each phase in the developed model is linked to its peer phase in the Markovian model.

2. Every link or connection between states in the HGFSM is transformed to a change pointer q_{ij} which shows the direction of movement from the source (S_i) to the destination (S_j).

3. Each change pointer is allied with its variable k_{ij} which represents a number of tasks or instructions that propagate from the source to the destination phase. That variable is included in eq. (1) to find the possibility/probability value P_{ij} which denotes the chance to flow or shift from the S_i to S_j and it is computed as shows in the following equation, where N_i represents the total number of jobs in S_i.

$$P_{ij} = K_{ij} / N \quad (1)$$

4. Every FSM figure and its states are linked with their Computation Structure Model "CSM" [4, 5] to express data change and its flow in it. CSM supports in creating or building the objective functions.

The Hierarchical Performance Model (HPM) stack representation is depicted in Fig 3. Each layer is used to obtain its objective functions based on different causes.

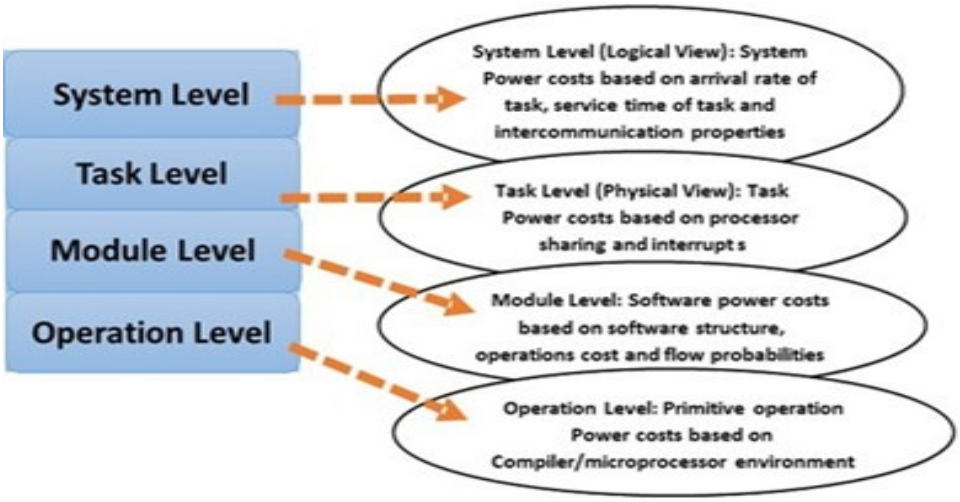

Fig. 3. Hierarchical performance model stack layers

The general overview of the system level is displayed in Fig 4; it provides a high-level image from top to illustrate the normal operation mode of OPENWRT. In Fig 4, bold lines specify the control flow direction and the dashed lines refer to a type and class of message being sent. Fig 5 displays the identifying basic elements of the system.

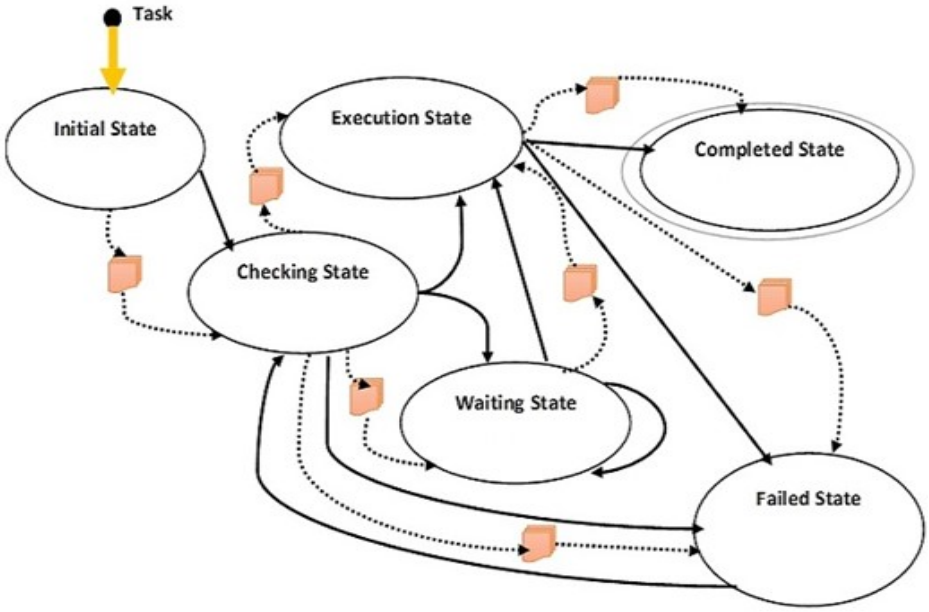

Fig. 4. System level overview for OPENWRT

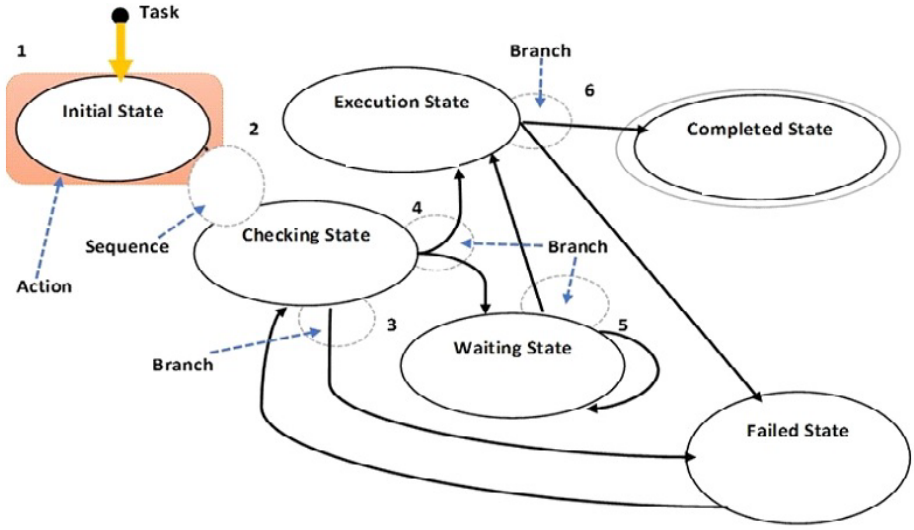

Fig. 5. System components

Fig 5 shows that the checking state is the place where main actions occur. The directory table which can be seen as the routing table is inspected to specify where the packet should be sent. In addition, every packet priority is examined to decide which packet can get its order first to be forwarded into the next phase. Table 1 displays the transition probabilities equations among all phases in the system.

Table 1. Probability transition equations

From-To	$Initial_1$	$Checking_2$	$Waiting_3$	$Execution_4$	$Failed_5$	$Completed_6$
$Initial_1$	0	$P_{12} = 1$	0	0	0	0
$Checking_2$	0	0	$P_{23} = k_{23}/N_2$	$P_{24} = k_{24}/N_2$	$P_{25} = k_{25}/N_2$	0
$Waiting_3$	0	0	$P_{33} = k_{33}/N_3$	$P_{34} = k_{34}/N_3$	0	0
$Execution_4$	0	0	0	0	$P_{45} = k_{45}/N_4$	$P_{46} = k_{46}/N_4$
$Failed_5$	0	$P_{52} = 1$	0	0	0	0
$Completed_6$	0	0	0	0	0	0

In Table 1, zeros mean there is no link and transition between states. The first subscript letter refers to the source states while the second subscript indicates the destination states. Computation structure model "***CSM***", which is referred as Data Flow Graph is used in this research to construct the objective functions. Readers can refer to [5] and [6] for more information.

The performance equation to determine the estimated average end-to-end latency is shown in eq. (2):

$$Delay = (1 * C_{Initial}) + [\ (1 + e_2) * (C_{check} + C_{test})\ ] + [e_7 * (C_{wait} + C_{test})\ ] + [\ (1 + e_2) * (C_{Exe} + C_{test})\ ] + (e_2 * C_{failed}) \quad (2)$$

In eq. (2), each variable, which is denoted by "C", is allied with its movement/flow parameter(s) "e"; this parameter represents a cost of going or passing over a certain route from the top to the bottom in the CFG. Each "e" can have a value, cost, ranging from {0,1, …., ∞} and largely relies on a category of the probability distribution being determined and taken. Those parameters are discrete random variables and can be represented by probability distribution categories [4, 5].

Bernoulli, Binomial, Geometric, Modified Geometric and Poisson are well known of probability distributions which are used these days in many applications. From the type of probability distribution being used, the concerned team can easily find the essential parameters which will be used in eq. (2) such as expected value E(e) [4,5,6]. Interested readers can refer to [4] for more information.

Now, finding the value of each parameter in eq. (2) can be easily obtained by knowing the operations occur in each phase by using its CSM. Keep in mind that the equations are derived from the control flow graphs "CFGs" which are constructed for each state. However, those CFGs are omitted in this paper.

1. Initial State: in this phase/state, the incoming message, also known as packet, is sent to its destination which is the next phase in the Checking state. Simulation experiments showed that sending packets to their destinations take a very small amount of time which can be ignored. So

$$C_{Initial} = 0 \text{ (in normal mode of operation).} \quad \text{Otherwise, } C_{Initial} \neq 0 \quad (3)$$

2. Check State: every message is examined from its header to know where it should be directed. Furthermore, the directory table is searched to determine the shortest path of that message to its destination. Then, the system performs a test to check if the processing unit is occupied or the message can be forwarded to it.

$$C_{check} = C_{receiving\ and\ checking} + C_{decision} + C_{test} \quad (4)$$

3. Wait State: this phase represents the medium of all waiting messages to get their turn once the P.U. is released from its occupied message. In addition, all messages are examined to find which one with a highest priority so it can be forwarded first to the P.U. The messages remain in this phase if the P.U. is busy with another job.

$$C_{wait}=([1+e_3]*(C_{check\ packet\ priority}+C_{test})) \quad (5)$$

4. Execution State: first, P.U. screens messages to find which one holds the highest priority to be forwarded first. Then, it sends it. In this phase, all messages are sent to their final addresses. In addition, keeping the track status of all sent messages whether successfully delivered or not is performed when a receiver sends an acknowledgment message.

$$C_{Execution}=[(1 + e_3 + e_7)*(C_{Handling} + C_{test})]+(e_3 * C_{aborted})+((1 + e_7)*C_{test}) \quad (6)$$

$$C_{Handling}= (1+e_3 +e_6)*(C_{ready}+C_{test})]+(e_3*C_{idle})+[(e_6 + 1)*C_{test}]+(1 * (C_{run})) \quad (7)$$

5. Failed State: First, an error message is shown to indicate that some messages were not delivered successfully for some reasons. Then, all messages in this phase are sent back to the next phase to restart their cycle again.

$$C_{failed} = C_{passing\ packet} + C_{display\ error\ message} \quad (8)$$

As stated earlier that sending packets/messages to their next phase is too small so it will be neglected. So

$$C_{passing\ packet} = 0 \quad (9)$$

Now, it is necessary to calculate a number of visits or traversals V for every state and it can be done as follows:

$$[V] = (I - P)^{-1} \quad (10)$$

In the above equation, [V] represents a matrix where its components specify the number of traversals to every phase and this matrix is a square matrix N*N. I is the identity matrix and P is the matrix of transition probabilities between all states. So the Average performance "delay" can be estimated as follows:

$$Delay\ (T_d) = \sum(V_i * C_i) \quad (11)$$

where subscript i denotes the number of states which goes from 1 to 6.

4. EXPERIMENTS AND RESULTS

Cisco Packet Tracer simulation tool is used to compute the actual delay inside a network. In addition, it helps by finding the cost of test operation which can be ignored as stated earlier due to its small value.

Two scenarios were developed and implemented in this research to compute the latency. In all implemented scenarios, several network topologies were used such as bus, star and mesh topologies. In this paper, point to point and hybrid topologies were implemented to compute the estimated average delay. Several considerations and hypothesizes were taken into the attention for the sake of profiling the delay time, also known as response time. Furthermore, numerous basic/essential parameters were counted and found to calculate the delay in eq. (2). Those considerations can be outlined as follows:

A) Once the message is received through incoming port/terminal, then its current time is the initial time and it is set to 0.

B) If the considered infrastructure has a few numbers of components (clients/workstations) then the time needed to examine the directory and finding the shortest path is negligible. However, if it is a big infrastructure then that time plays a significant role. In this paper, the network infrastructure was small so $C_{checking} = 0$.

Scenario 1:

Fig. 6 depicts this scenario as it has one router, 4 switches with 16 ports in each, where each 2 switches were connected directly, 2 servers with several services running on them and the speed link = 1GBS between all devices in the network.

In total, 100,000 messages/packets were emulated/imitated in the cisco Packet Tracer. Table 2 shows the consumed time in every phase/state.

Table 2. Delay time in scenario 1

DELAY TIME IN μs			
Initial Time	*Checking Time*	*Waiting Time*	*Forwarding Time*
0	*0*	*12*	*9.14*

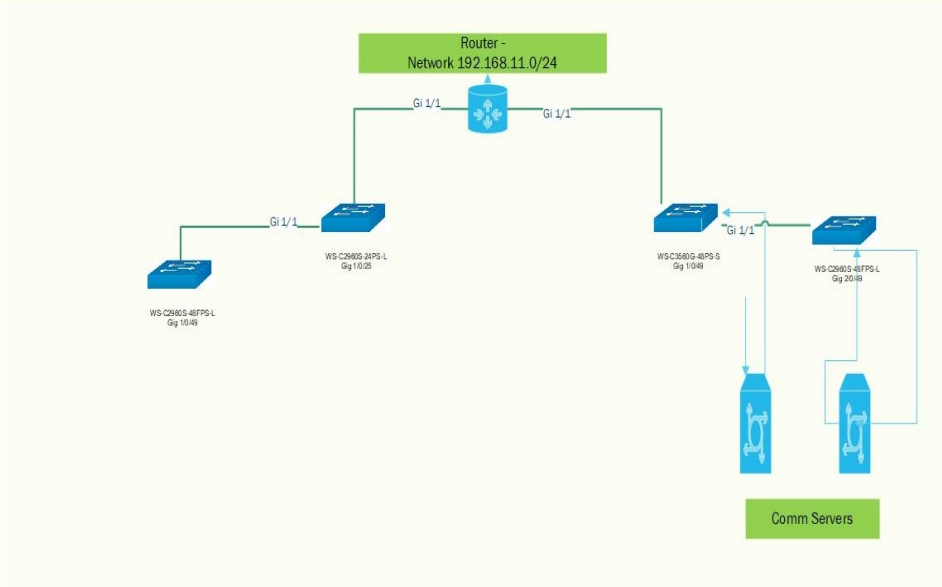

Fig. 6. Network topology and its devices

So the expected average execution time is

$$T_d = \sum (V_i * C_i),\ T_d = 12 + 9.14 = 21.14\ \mu s$$

The actual delay is found to be T = 19.7 μs.

Fig. 7 illustrates the values of the exact and calculated delay as obtained above. Fig. 7 clearly shows that the developed model gives a close estimate value since the error is less than 10%.

Fig. 7. Average actual and estimated end-to-end delay time in scenario 1

Scenario 2:

Fig. 8 shows the network being tested in this scenario with different link speed. X and Y refer to the routers models and their numbers.

Nearly 70,000 messages/packets were emulated. Table 3 illustrates the needed time for the message to be delivered to its destination.

Table 3. Delay time in scenario 2

DELAY TIME IN µs			
Initial Time	Checking Time	Waiting Time	Forwarding Time
0	0	7.02	45.33

So the expected average execution time is
$T_d = \sum (V_i * C_i)$
$T_d = 7.02 + 90.66 = 97.68$ µs.

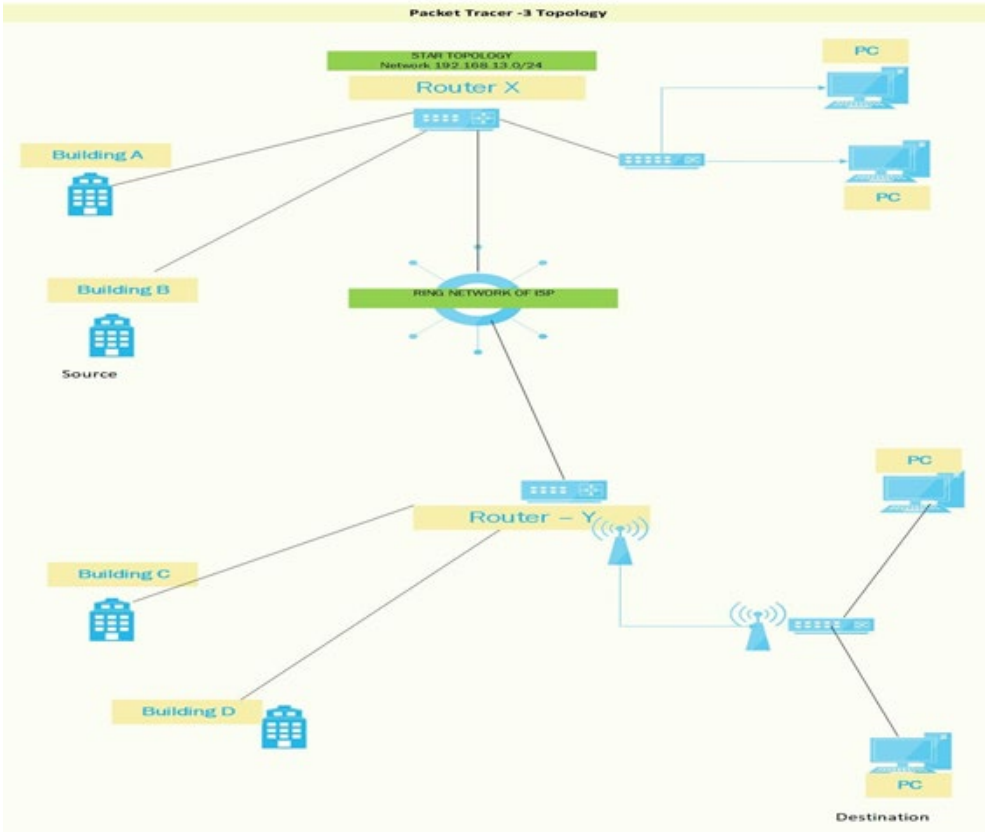

Fig. 8. Network topology and its infrastructure

Fig. 9 illustrates the values of the exact and calculated delay.

Overall error found to vary from 7% to 12%. From both scenarios, several factors affect the end-to-end delay which can be summarized as follows:

1. The size of network infrastructure plays a significant role as delay increases when packets traverse through a large network.
2. The number of hops affects the delay.
3. The speed of links also affects the delay as shown in scenario2.

Fig. 9. Average actual and estimated end-to-end delay time in scenario 2

5. CONCLUSIONS

This paper presented the developed framework to approximate the expected average end-to-end delay in OPENWRT. Two scenarios were presented to illustrate how the delay is computed using mathematical scheme and Cisco Packet Tracer simulator to represent the corresponding levels of the model. The obtained results from both conducted scenarios indicate that the estimated values are acceptable as they are close to the actual ones since the gap between them is narrow. In addition, using mathematical approach is faster than using dedicated tools since those tools require special setups and configurations while the mathematical method does not require those circumstances.

Lastly, the used approach is capable of spotting or determining the bottleneck inside a system under consideration. Currently, finding a bottleneck of a robotic system is under investigation. In addition, this approach gives a promising future in the performance evaluation field due to its capabilities and easiness for computing several metrics.

REFERENCES

[1] Lentaris, G., Maragos, K., Stratakos, I., Papadopoulos, L., Papanokolaou, O., Soudris, D. High-performance embedded computing in space: Evaluation of platforms for vision-based navigation. *Journal of Aerospace Information Systems*, Vol. 15, No. 4, 2018, pp. 178-192.

[2] Sreenath, M., Vijaya, P. A. Performance evaluation of embedded system using scheduling algorithms. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology IJSRCSEIT*, Vol. 3, No. 1, January 2018, pp. 85-89.

[3] Pilato, L., Fanucci, L., Saponara, S. Real-time and high-accuracy arctangent computation using CORDIC and fast magnitude estimation. *Journal of Electronics, MDPI, Real-Time Embedded Systems*, March 2017, pp.54-63.

[4] Alsheikhy, A. *High Performance Embedded System.* Doctoral Dissertation, University of Connecticut, CT, USA, 2016.

[5] Alsheikhy, A., Han, S., Ammar, R. Hierarchical performance modelling of embedded systems. *20th IEEE Symposium on Computers and Communications*, 2015, pp. 936-942.

[6] Alsheikhy, A., Han, S., Ammar, R. Delay and power consumption estimation in embedded systems using hierarchical performance modelling. *15th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, December 2015, Abu Dhabi, UAE, pp. 34-39.

[7] Sundaresan, S., Feamster, N., Teixeira, R. Measuring the performance of user traffic in home wireless networks. *Proceedings of the 16th International Conference in Passive and Active Measurement*, Book Chapter, New York, NY, USA, 2015, pp.305-317.

[8] Bajpai, V., Schonwalder, J. A survey on Internet performance measurement platforms and related standardization efforts. *IEEE Communications Surveys and Tutorials,* Vol. 17, No. 3, 2015, pp. 1313-1341.

[9] Kremer, G., Owezarki, P., Berthou, P., Capdehourat, G. Predictive estimation of wireless link performance from medium physical parameters using support vector regression and K-nearest neighbours. *Proceedings of the 6th International Workshop of Traffic Monitoring and Analysis*, Book Chapter, London, UK, April, 2014, pp. 78-90.

[10] Xue, L., Mok, R. K. P., Chang, R. K. C. OMware: An open measurement ware for stable residential broadband measurement. *Proceedings of the ACM SIGCOMM Computer Communication Review*, Vol. 43, No. 4, August 2013, pp. 497-498.

[11] Palazzi, C. E., Brunati, M., Roccetti, M. An OPENWRT solution for future wireless homes. *Proceedings of the 2010 IEEE International Conference on Multimedia and Expo "ICME"*, Suntec City, July, 2010, pp. 1701-1706.

[12] Botta, A., Donato, W. D., Pescape, A., Ventre, G. Networked embedded systems: A quantative performance comparison. *Proceedings of the 2008 IEEE Global Telecommunication Conference*, New Orleans, LO, USA, 2008, pp. 1-6.

[13] Smarkusky, D., Ammar, R., Antonios, I., Sholl, H. Hierarchical performance modelling for distributed system architectures. *Computer and Communications*, 2000. *Proceedings. ISCC 2000. 5th IEEE Symposium*, July 2000, pp. 659–664.

[14] Ammar, R. *Software Performance Analysis* (lecture notes), Univ. of Connecticut, 1991.

[15] Lee, B., Lee, E. A. Interaction of finite state machines and concurrency models, *Proceeding of Thirty-Second Annual Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, California, November 1998, pp. 1715-1719.

[16] Stan, A., Botezatu, N., Panduru, L., Lupu, R. G. A Finite state machine model used in embedded systems software development, *Universitatea Tehnica*, Gheorghe Asachi din Iasi, 2009, pp. 51-63.

[17] Kravets, O.Ja. Atlasov, V., Aksenov, I.A. et al. Increasing efficiency of routing in transient modes of computer network operation. *International Journal on Information Technologies and Security*, Vol. 13, No. 2, 2021, pp. 3-14.

Information about the author:

Ahmed Alsheikhy – Assistant Professor, Northern Border University, College of Engineering, Electrical Engineering Dept., Arar, Saudi Arabia. Received the B.S. and M.S. degrees in Electrical and Computer Engineering from King Abdulaziz University 2004 and 2010, respectively. In 2016, he received the Ph.D. degree in Computer Science and Engineering from University of Connecticut, USA. His research interests are on Performance Evaluation, Artificial Intelligence, Satellite and Radar communication systems and Internet of Things.

Manuscript received on 02 November 2020