Analytical Model to Create Proxy Server Sessions in Multimedia Networks

Mehdi Khazaei*
Department of Computer Engineering, Kermanshah University of Technology, Iran
m.khazaei@kut.ac.ir

Received: 29/Aug/2021 Revised: 13/Nov/2021 Accepted: 07/Dec/2021

Abstract
One of the most popular and widely applied protocols on multimedia networks is the Session Initiation Protocol (SIP) to create, modify, and terminate the sessions. SIP is the platform of Next Generation Networks (NGN). In this way, SIP should be able to respond to the needs of such a largely-used network. One of the major problems in SIP networks is overload. This challenge creates a sharp drop in quality of service for NGN users. In this regard, many studies have been conducted on the effectiveness of this protocol, especially under overload. A new analytical model is developed that prioritizes the SIP message processing. An analytical approach is proposed based on the Mean Value Analysis (MVA) algorithm in queue theory. Considering some appropriate assumptions customizing MVA as to implement this proposed model and to cope with the limitations of the MVA is highly essential. The output of the analytical model is compared with the standard SIP model obtained from the simulator and the results confirm that prioritizing original messages would enhance the SIP performance at different load conditions. Prioritization of original messages is advantageous, and outperforms the normal SIP. Nevertheless, prioritizing the repeated messages not only has no advantage, but also its performance is less than the normal SIP.

Keywords: Modeling; Prioritize; Queue theory; Proxy server; MVA algorithm.

1- Introduction

Today's world requires designing an efficient and scalable system, since the lack of these criteria may cause the system not to be implemented, so that the system is quickly eliminated from the competition. As the performance improvement after system designing may fail or associate with high costs, it is better to consider this issue during the design phase. Accordingly, it is necessary to develop a model of the system before construction that can describe a general view of the system to improve the performance of the system lifetime. Nevertheless, in practice, it is impossible to generate a model with all details and limitations of the actual system, and if possible, it is highly complex and expensive. Consequently, many critical details are mainly not considered in the modeling process that can affect the performance via either low effect or ineffectiveness. The details level of the model depends on the aim of the model that should not be more complex than the defined target.

The modeling process can be performed as simulation or analytic. In simulation method, a computer program mimics the various states of the system. However, the analytical method includes several mathematical and computational equations calculating performance parameters based on offered load. It should be noted the analytical model involves fewer details than the simulation method because it is implemented using some mathematical equations. The reasons to utilize the analytical modeling instead of the simulation method are: 1) simulators may depend on either time or event, so that non-convergence or very long convergence occurs by increasing the simulation time or load, and 2) if the parameters are not valued correctly, the results of the simulation are unrealistic. However, the analytical model calculates the performance parameters in a short time, and the results do not depend on the specific settings of the parameters. Ensuring the accuracy of the results of the analytical model, the output of the model is compared to the output of the simulation, which is highly useful.

In the queuing theory, the Markov model is utilized to describe the sequence of possible events, in which the probability of each event occurring depends only on the state of the previous event. In fact, the Markov model is a memory-less random process, which is the basis of most quantitative analysis methods. This model is formed based on the state diagram that is a powerful tool to describe systems. In addition, it can be visually understood by less-experienced people and is employed in many problems. It should be noted that as the Markov model often determines the interaction between the various components of the system, it could be utilized as a predictor of the system behavior. Nevertheless, the main challenge of the Markov model is its extreme sensitivity to

* Corresponding Author

the sudden increase in the state space. In some cases, as the number of requests increase, the state space becomes so large that solving its equations does not converge. Therefore, some alternative solution methods like Mean Value Analysis (MVA) have been proposed.

The MVA is an algorithm developed to solve various Markov models, which does not require both explicit and simultaneous solutions of a large number of equations. The MVA utilizes a recessive relation to obtain each state instead of solving simultaneous equations to find probabilities.

The Session Initiation Protocol (SIP) is applied in many multimedia networks, and is introduced as the kernel protocol for next-generation networks. SIP is a text-based protocol where both request and response are expressed as HTTP, standardized by IETF. In this protocol, the control and data transmission sections are separated. SIP contributes to the end points' coordination, finding session participants, and agreeing on the session features.

Many studies are done on the performance of the SIP servers under specific conditions like overload. Overload is one of the most unsolved issues in SIP networks. In general, when the offered load to the server exceed its processing capacity, overloading becomes inevitable and the retransmission leads to queues server overflow. Hence sessions creating is delayed and overload is increased. Overload problem solutions are presented within simulation or analytical models, which are often simulation dependent.

One approach to overcome the local overload is to prioritize processing of the incoming requests in the proxy servers, where retransmitted messages are reduced and the corresponding transaction is terminated faster [1-4]. The prioritization method can be combined with intelligent methods to reduce the unnecessary in retransmissions by giving high priority to retransmitted requests [5]. As to VLB-CAC [6], the Virtual load-balanced call admission controller (VLB-CAC) is introduced for the cloud-hosted SIP servers to control overload in SIP network. The VLB-CAC determines the optimal call admission rates and signaling paths based on a heuristic mathematical model. The proposed scheme is implemented in smart applications on virtual infrastructure as testbed. The new concepts of computer networks, OpenSIP^{Partial} and VLB-CAC are applied in overcoming the overload issue. As to the OpenSIP^{Partial}, the SIP network is upgraded based on SDN and NFV technologies [7, 8].

On the other, SIP proxy server is composed of the dynamic subsystems, each subsystem interacts with each other to manage session constantly. In addition, overload indicate the proxy server as dynamic and complex system. Specially, with the development of SIP networks and their role in communication, the number of SIP servers will increase, and they will be geographically distributed. Hence, simulation is slow and time-consuming method to model SIP server but an analytical method is the right tool.

A priority-based approach is modeled through M/D/1/R according to RFC 357 in [9, 10]. In this method, three normal (L), overload (H) and removal (R) areas are defined, the input load is controlled according to the number of requests in each area. Also, two types of loads are defined, the first has an exclusive priority over the second. In the overload area, only first type load is accepted, but no load is accepted in the rejection area. In this method, a random process is defined for each region, and the rapid probability distribution function of that region is obtained. At the end, the mean return time to the normal state is calculated by the distribution function [11].

Applying the prioritization and the Markov model, the proxy servers between each UAC and UAS are modeled through a series of M/M/1 open queues. Each proxy server consists of a number of parallel and series queues, each of which is associated with a specific request or response. In this method, each message enters one of these queues with a defined probability. Then, based on the Markov model, an analytical model is presented that assesses the performance of the SIP network by input load, service rate, and different delays. Finally, it has been shown to improve the performance in the use of memory and CPU [12, 13].

Appling the queue theory model, a hierarchical model is presented in [14] to assess the performance of the SIP server in normal and overload states, where, Petri Net networks are used to check the network performance. In this model, two types of load management are considered to reduce retransmissions. The present paper aims to prioritize resource planning and reservation, so that the performance of the degradation system is at a lower level.

As statistical analytical models, the SIP server is modeled by the queuing system and it is analyzed by numerical methods. In these method, two limited queues for invite and non-invite requests are considered, which the non-invite queue is a priority. For the non-invite queue, a threshold (L) is considered, and when the number of requests in the queue exceeds (L), the incoming requests are rejected in the invite queue. The switch between the two queues for processing is as exponential. The polling systems, along with two general and gradual policies, have also been utilized to process requests. In any gradual processing, a number of requests per queue are processed, and the rest of the requests are waited, until it is their turn to be processed. Evaluation indicates that the gradual method has better performance [15, 16].

Most of the non-statistical prioritization models are implemented through the recursive equations. The fluid-flow model of one server, and then tandem servers are obtained for infinite queue length [17]. In this research, the initial conditions of the queue are obtained, so that the server is not overloaded by prioritizing the messages. The fluid-flow model of tandem servers with a finite buffer size can be calculated [18].

The trapezoid topology to tandem servers and several user agents, providing a Liapanov function that can assess the probability of rejection of calls through the proxy servers to achieve stability [19]. For this purpose, the fluid-flow model of the trapezoid topology of the SIP server is obtained by giving higher priority to non-invite messages.
As to LB-CAC a call acceptance controller with load balancing is proposed in [20], where a linear programmed model is involved in estimating the acceptance rate and signaling pathways, to assure the CPU and memory allocations and acceptance rates optimization.

A. Innovations

Many solutions have been offered to overcome overload, one being the priority-based processing. There exists no article regarding an analytical model based on MVA with low cost computations. A new analytical model based on the MVA algorithm that models the SIP server performance under message prioritization, together with considerable details and accuracy is developed in this articles. For this purpose, the MVA algorithm must be customized to meet these proposed models requirements, something not done yet in this field. The output of this analytical model is compared with the same of the simulator to determine its reliability.
The article is organized as follows: The literature is reviewed in Sec. 2; the proxy server is modeled in Sec. 3; model is adapted to the MVA in Sec. 4; the results are analyzed in Sec. 5 and article is concluded is Sec. 6.

2- Literature Review

Since the purpose is modeled SIP server based on the MVA, needed the related concepts to be briefly explained.

2-1- MVA

The MVA has been developed for a broad class of useful Markov models that do not require the explicit solution to a large number of simultaneous equations. Instead of solving a set of simultaneous linear equations to find the steady state probability of being in each system state, from which performance metrics can be derived, MVA is a simple recursion. MVA calculates the performance metrics directly for a given number of requests, knowing only the performance metrics when the number of requests is reduced by one. The recursion is intuitive, efficient, and elegant. Its impact on the field of analytical performance evaluation has been huge. The basic MVA algorithm is quite powerful. It is applicable across a wide set of performance models. It has been the focus of much research and several extensions have been developed. These include: multi-class, load dependent servers and open and closed classes. The optimization parameters for each request are solved through MVA, with the following limitations [21]:

- MVA does not provide the steady-state probability on its own.
- MVA does not provide information on the transient and non-stationary state of the system.
- MVA does not model the state-dependent behavior.
- MVA is not applied in solving the non-multiplicative problems, like the Gaussian, uniform and stationary distributions, priority queues and multi-class FIFS queues. The approximate MVA techniques is applied to solve non-multiplicative cases.

2-2- SIP

The SIP protocol architecture contains two logical entities, namely the user agent and the server agent. The User Agents (UA) is divided into two groups of User Agent Client (UAC) is an applicant and User Agent Server (UAS) is a respondent. The proxy servers seek the user location and perform a routing operation to deliver messages to the desired user. SIP transaction is a request and all the relevant responses are exchanged between two adjacent components. Proxy servers are configured state-full or state-less depending on network conditions. In the state-full, the proxy server saves the data of each transaction to perform some operations like requests retransmission. In the state-less proxy, no operation log is recorded. Retransmission mechanism is utilized to prevent the loss of packets when the SIP is run on a non-reliable transmission layer protocol like UDP [22, 23]. The call establishment model between UACs and UASs as state-full proxy server is illustrated in Fig. (1).
The UACs start sending messages by delivering an invite messages (λ_{IO}), so that when this message reaches the server, the 100-Trying response ($\mu_{100\text{-}Trying}$) as confirming is sent to UACs, and the invite is passed to the other side of the session (μ_{IO}). When the invite message reaches by UAS, 180-Ringing message (λ_{180}) and 200-OK (λ_{200}) are sent by answering the call. Finally, a session is made when ACK (λ_{ACK}) is sent for every pair of 180-Ringing/200-OK ($\mu_{180\text{-}200}$). Afterward, the data are exchanged without passing through the proxy server. In terms of confirmation and retransmission, SIP transactions are classified as invite or non-invite invites. The different variables and parameters applied in this manuscript are tabulated in the Table 1.
When the invite request is transmitted, a timer is responsible to control the retransmission set to the initial value. Upon expiration of this timer, the request will be retransmitted again, thus, doubling the timer. This process continues until a response is received or the timer is expired with a default value, consequently, an invite request may be repeated for about six times. The λ_{IR} and λ_{IRS} symbolize the repeated invites retransmitted by the UACs or proxy servers. If no ACK is received the UAS will retransmitted a 200-OK message, that is, a 200-OK request can be retransmitted for about eleven times [22].

The μ_R and μ_{IR} symbolize the processing rate of responses and repeated invites, respectively.

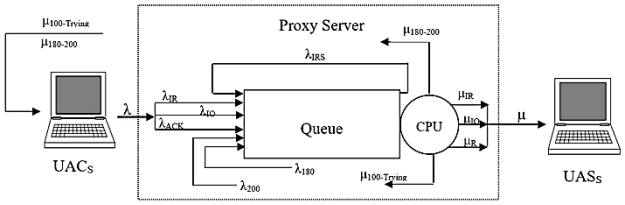

Fig. 1 The model of session create between the UAs

Table 1 The model parameters

Parameters	Description
λ	Offered load transmitted through UACs
λ_{IO}	Original invites rate transmitted through UACs
λ_{IR}	Repeated invites rate transmitted through UACs
λ_{IRS}	Repeated invites rate retransmitted through server
λ_{ACK}	ACK responses rate transmitted through UACs
λ_{200}	200-OK responses rate transmitted through UASs
λ_{180}	180 responses rate transmitted through UASs
μ_{IO}	Processing rate of original invites
μ_{IR}	Processing rate of repeated invites
μ_R	Processing rate of responses
μ	Processing rate of all requests
q_{IO}	Queue of original invites
T_j	Time to retransmit repeated messages
D_i	Sum of the times' means when a request receives service from the i^{th} source
V_i	Number of visits to the i^{th} source
S_i	Length of time that a request receives service from the i^{th} source
$D_{Cpu,k}^{Org}$	Demand service of original CPU applied through k^{th} category of invites (k= IO or IR)
$D_{Cpu,R}^{Org}$	Demand service of original CPU applied through responses
$D_{Cpu,R}$	Demand service of original CPU applied through responses
$D_{Cpu,R}^{shw\_y}$	Demand service of y^{th} Shadow CPU applied through responses
$D_{Cpu,k}^{shw\_y}$	Demand service of y^{th} Shadow CPU applied through k^{th} category of invites (k= IO or IR)
$D_{Cpu,k}$	Demand service of original CPU applied through k^{th} category of invites (k= IO or IR)
$U_{CPU,R}$	Original CPU utilization applied through responses
$U_{CPU,k}^{shw\_y}$	y^{th} Shadow CPU utilization applied through k^{th} category of invites (k= IO or IR)
$X_{0,R}$	Throughput of responses
$X_{0,k}$	Throughput of k^{th} category of invites (k= IO or IR)
$\alpha(j)$	Service rate coefficient
$\mu(j)$	Mean service rate of j requests
M	Coefficient determining the number of invites, repeated for the first time
H	Upper bound from which the number of processes remains constant
ω	Volume from which the service rate is constant
$P(j\|n)$	Probability of *j* occurrence when *n* requests are present throughout the network
N_{IO}	Number of original invites in the corresponding queue
N_{IR}	Number of repeated invites in the corresponding queue
N_R	Number of responses in the corresponding queue
$R_{IO}^{/}$	Average residence time for original invites
$R_{IR}^{/}$	Average residence time for repeated invites
$R_R^{/}$	Average residence time for responses
C	Maximum proxy server capacity

3- Proxy Server Modeling

The SIP proxy server model schemed in Fig. (1), is customized to implement prioritization, Fig. (2), where, the three separate queue of original invite requests, queue of repeated invite requests, and queue of responses are considered for transactions. In this modeling, the assumptions are as follows:

1- On proxy servers, UASs and UACs, the processing is compatible with the Poisson distribution [24, 25].
2- The transmitted original invites are compatible to the Poisson distribution through UACs [24, 25].
3- The UACs and UASs are high-powered systems to establish communication among subscribers by sending and receiving text-based SIP messages; message-processing delays are negligible and the responses are quickly returned.
4- In the proposed model, processing the responses queue is a priority, where, the relevant timers expire sooner, lead to a reduction of repeated requests and overload prevention. Once the queue is empty, it is time to process the queue with the next priority [5].
5- In this proposed model, the scheduling prioritization is exclusive. In request processing at low priority, if a request with a higher priority enters the relevant queue, the request processing with low priority stops, and the requests are processed with higher priority. The requests are processed in the lower priority queue when the higher priority queue is empty.
6- Because the three responses of 200-OK, 180-Rining, and ACK are similar in terms of CPU consumption and demand service, the messages within the queue of responses are considered as one class, consequently, the demand service is considered as D_R.
7- According to second assumption, the UASs quickly transmitted back responses for each invite. Timers are reset through processing returned responses, and no retransmission enters the queue through the server ($\lambda_{IRS}=0$). Therefore, original and repeated invites are only sent through the UACs.

4- Adapting the Model to the MVA Algorithm

The implementation of the proposed model through the MVA has some practical limitations, consequently, required compliance with this algorithm. The limitations consist of: 1) the adaptation of the input of the queues to the Poisson distribution, 2) prioritizations of queues and 3) processing the load dependence of the queues.

4-1- Adapting the Inputs to Poisson Distribution

As to the queue of responses, the third assumption states that, as soon as the original invite arrives at UAS, 200-OK

and 180-Ringing are returned immediately; consequently, the queue of responses is of exclusive priority, where the responses are transmitted to UACs and they return the ACK response without any delay, thus, no retransmission for the responses. The number of the 200-OK, 180-Ringing, and ACKs is equal to the number of original invites processed by the server obtained through Eq. (1).

$$\lambda_{ACK} = \lambda_{180} = \lambda_{200} = \mu_{IO} \quad (1)$$

The processing of the original invites in UAs is based on the Poisson distribution, accordingly, the entry rate of responses to the corresponding queue on the same distribution. As to second assumption, the input of the original invite queue must correspond to the Poisson distribution. In the repeated invite requests, it must be proved that the Poisson distribution is adapted. If success (p) is defined as not transmitting a repeated request to an original invite, then failure is defined as transmitting a repeated request to the same invite.

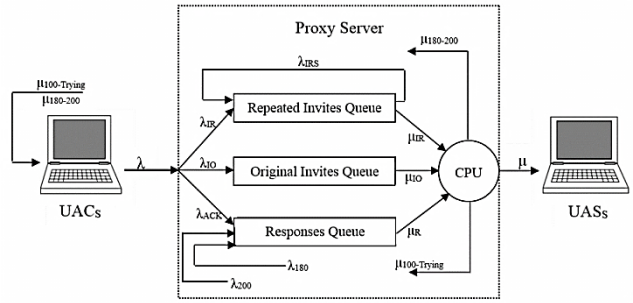

Fig. 2 Modeling the messages prioritization on the proxy server

P obtained through Eq. (2), where, p is percentage of invite requests, processed.

$$P = \frac{\mu_{IR} + \mu_{IO}}{\lambda_{IR} + \lambda_{IO}} \quad (2)$$

The number of repeated requests (x) from the total (n=λ_{IO}) original transmitted invites follow the binomial distribution obtained through Eq. (3).

$$\lim_{n\to\infty} \binom{n}{x} p^x (1-p)^{n-x} = \frac{e^{-\mu}\mu^x}{x!} \;\; ; \;\; \mu = np \quad (3)$$

In binomial distribution, if the number of repetitions is very high and the probability of success is low, it can be estimated through the Poisson distribution. When the number of the invite requests transmitted by UACs is very high, it is not possible that the server will process an original invite successfully with no need for repetitions. Therefore, the number of the repeated invite requests follows the Poisson distribution.

Time is split in to discrete proportions to calculate the parameters, which allows discrete time modeling and makes it easier to understand and simulate the model. Smaller time proportions to obtaining more accurate results, while simulation time increases and vice versa. In the equations, the variables t and n are the time and time slices, respectively. The number of the repeated invite requests (λ_{IR}) is obtained through Eq. (4) [18].

$$\lambda_{IR}(n) = \sum_{j=1}^{6} \left[\lambda_{IO}(n - T_j) + q_{IO}(n - T_j) - \mu_{IO}(n - T_j, n)\right] \quad (4)$$

The length of the nth original invite queue (q_{IO}) is calculated through Eq. (5).

$$q_{IO}(n+1) = [q_{IO}(n) + \lambda_{IO}(n) - \mu_{IO}(n)] \quad (5)$$

The time to transmit repeated invites (T_j) is obtained through Eq. (6).

$$T_j = (2^j - 1)T_1 \qquad ; 1 \le j \le 6 \quad (6)$$

The first retransmitted is generated at T_1 = 0.5 sec and the last at maximum 64 * T_1 Sec [18].

4-2- Implementing the Priority of Queues

Because prioritization of the queues violates the multiplicative property of MVA, the approximate methods should be applied for this proposed model. The Stepwise Inclusion of Classes (SWIC) algorithm is run to implement the priority of queues, where the queue of responses never stops by the queue of invites unless it is empty. The invite queues see only the part of CPU that remains after processing the queue of responses. In this process, first, an estimation of CPU consumption by the queue of responses is made, which can be applied by modeling the queue of responses as the only queue of the system. Then, one of the invite queues is added, and modeling is made with two queues and an additional processor named the first shadow processor. Finally, the third queue with the second shadow processor is added to the model. The original invite queue takes precedence over the repeated invite, Fig. (3), while the opposite holds in Fig. (4).

The demand service (D_i) is the sum of the times when a request receives service from the ith source, Eq. (7).

$$D_i = V_i * S_i \quad (7)$$

D_i does not include the waiting time in the queue. S_i is the time of a request received service from the ith source for each visit, while V_i is the number of visits to the ith source to be made by the request.

The demand service of invite queues from the main CPU, and the demand service of response queue from the shadow CPUs are zero, respectively, Eqs. (8 and 9) [21].

$$D_{Cpu,k}^{Org} = 0 \qquad ; \;\; k = IO \; or \; IR \quad (8)$$

$$D_{Cpu,R}^{Shw_y} = 0 \qquad ; \;\; y = 1 \; or \; 2 \quad (9)$$

The variable k is defined as the IO for original invites and IR for repeated invite in all equations based on the intended priorities. Moreover, the variable y indicates the shadow processor number that is either one or two.

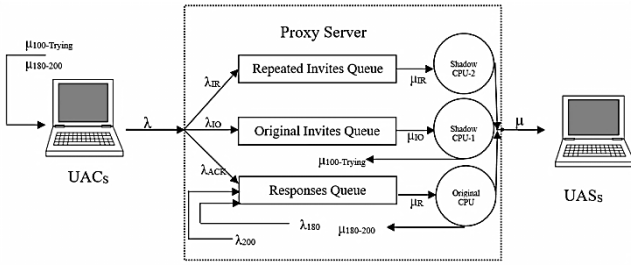

Fig. 3. Original invites queue has priority over the repeated invites queue

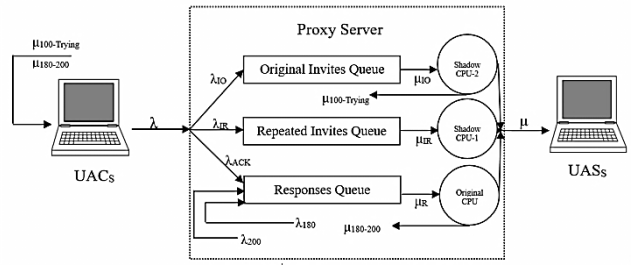

Fig. 4. Repeated invites queue has priority over the original invites queue

Because the queue of responses has the highest priority, the full CPU capacity is provided for responses, making the demand service for responses from the main CPU equal to the demand service expressed in Eq. (10).

$$D_{Cpu,R}^{org} = D_{Cpu,R} \tag{10}$$

The CPU utilization is obtained through Eq. (11).

$$U_{\mathrm{Cpu,R}} = \mathrm{D_{Cpu,R}} * \mathrm{X_{0,R}} \tag{11}$$

In the next step, the first shadow CPUs is added following the priority order. The demand service is then obtained through Eq. (12).

$$D_{Cpu,k}^{Shw\_y} = \frac{D_{Cpu,k}}{1 - \sum U_{high-priority}} \tag{12}$$

(e.g. $D_{Cpu,IO}^{Shw\_1}$ specifies the first shadow processor, allocated to the original invite queue, that is, the original invite requests take precedence over repeated invites, Fig. (3)). The CPU utilization of invite queues is obtained through Eq. (13) for the first and second shadow CPUs [21].

$$U_{Cpu,k}^{Shw\_y} = X_{0,k} * D_{Cpu,k}^{Shw\_y} \tag{13}$$

4-3- Request Processing

The request processing in each queue is performed in a load-independent or load-dependent manner. In the first, the mean service rate μ does not depend on the number of in-queue requests, while in the second, μ (n) is a function of load (n). In the model provided proxy server, the queue of responses and original invites is load independent, while in queues of repeated invite, processing is load-dependent. As to assumptions (3 and 4), the messages inside the response queue and the original invites queue are not repeated, consequently, they can be processed according to the capacity provided for the main and the corresponding shadow CPU. The number of original invites and responses in corresponding queue is obtained to modify Little's Law, Eqs. (14 and 15).

$$N_{IO} = \frac{U_{Cpu,IO}^{Shw\_y}}{1 - U_{Cpu,IO}^{Shw\_y}} \tag{14}$$

$$N_R = \frac{U_{Cpu,R}}{1 - U_{Cpu,R}} \tag{15}$$

The average residence time of the r[th] class is the total time spent by the same requests at the i[th] device in all visits. The average residence time of original invites and responses on corresponding CPUs is expressed as, Eqs. (16 and 17).

$$R_{IO}^{/} = \frac{D_{Cpu,IO}^{Shw\_y}}{1 - U_{Cpu,IO}^{Shw\_y}} \tag{16}$$

$$R_R^{/} = \frac{D_{Cpu,R}}{1 - U_{Cpu,R}} \tag{17}$$

In load-dependent processing, both the service rate and response time are a function of requests distribution, therefore, the MVA equations must be adjusted to fit this distribution. Therefore, instead of the queue length, a complete queue distribution is required. In the corresponding queue, the number of repeated invites is calculated through Eq. (18), provided that the stability condition ($\left[\frac{\mathrm{U}}{\alpha(\mathrm{w})}\right] < 1$) is met [21].

$$N_{IR} = p(0|\lambda_{IR}) * \left\{ \sum_{j=1}^{\omega} j \frac{U_{Cpu,IR}^{Shw\_y j}}{\beta(j)} + \frac{U_{Cpu,IR}^{Shw\_y \omega+1}}{\beta(\omega)*\alpha(\omega)} * \left[\frac{\frac{U_{Cpu,IR}^{Shw\_y}}{\alpha(\omega)} + (\omega+1)*\left(1 - \frac{U_{Cpu,IR}^{Shw\_y}}{\alpha(\omega)}\right)}{(1 - \frac{U_{Cpu,IR}^{Shw\_y}}{\alpha(\omega)})^2} \right] \right\} \tag{18}$$

$\mathrm{P(j|\lambda_{IR})}$ is the probability of j occurrence when λ_{IR} repeated invites enter to the queue, Eq. (19).

$$p(j|\lambda_{IR}) = \frac{D_{Cpu,IR}^{Shw\_y} * X_{0,IR}}{\alpha(j)} p(j-1|\lambda_{IR}), \;\; ; j = 1,2,\ldots.\lambda_{IR} \tag{19}$$

The probability $\mathrm{P}(0|\lambda_{\mathrm{IR}})$ is recursively obtained by resolving the $\mathrm{P}(j|\lambda_{\mathrm{IR}})$ and requiring all probabilities sum to one, Eq. (20).

$$p(0|\lambda_{IR}) = [\sum_{j=0}^{\omega} \frac{U_{Cpu,IR}^{Shw\_y j}}{\beta(j)} + \frac{[\alpha(\omega)]^{\omega}}{\beta(\omega)} * \frac{[\frac{U_{Cpu,IR}^{Shw\_y j}}{\alpha(\omega)}]^{\omega+1}}{1 - \frac{U_{Cpu,IR}^{Shw\_y j}}{\alpha(\omega)}}]^{-1} \tag{20}$$

α (j) is the coefficient of service rate, Eq. (21), directly related to the first repeated invites because they would terminate the state diagram of server transactions while processing the other repeated invites only waste the CPU.

$$\alpha(j) = \frac{\mu(j)}{\mu(1)} = \begin{cases} M * j & j \le H \\ H & j > H \end{cases} \tag{21}$$

M is a coefficient determining the number of the invites repeatedly delivered for the first time and is computed through Eq. (22).

$$M = \frac{\lambda_{IR}(j - T_1\} + q(j - T_1) - \mu_{IR}(j - T_1, j)}{\lambda_{IR}(j)} \quad (22)$$

However, service rate coefficient of repeated invites queue is limited to H, determining the upper bound where the throughput remains constant, Eq. (23).

$$H = X_{0,IR} = \frac{U_{Cpu,IR}^{Shw\_y}}{S_{IR}} = \frac{U_{Cpu,IR}^{Shw\_y}}{\frac{D_{Cpu,IR}^{Shw\_y}}{V_{IR}}} = \frac{V_{IR} * U_{Cpu,IR}^{Shw\_y}}{D_{Cpu,IR}^{Shw\_y}} \quad (23)$$

H is equal to shadow CPU throughput of repeated invites queue. W is the volume where the service rate is almost constant, that is, equal H. The β (j) is computed through Eq. (24) [21].

$$\beta(j) = \alpha(1) * \alpha(2) * \ldots * \alpha(j) \quad (24)$$

The average residence time for load-dependent processing of repeated invites is obtained through Eq. (25).

$$R_{IR}^{/} = \frac{N_{IR}}{\lambda_{IR}} \quad (25)$$

5- The Results Analysis

The C++ is applied to analyze the results. In this model, the server completes the C call per second (cps). The service time (S) for each session is equal to *1/C*. Each session consists of an original invite request and the three corresponding responses of 200-Ok, 180-Ringing and ACK, therefore, V is considered as 0.4 and 0.2 for the original invite and each one of the above responses. According to [26], the value of V for each repeated invite is more than the processing response and less than one original invite. Here, V is defined as 0.3 for each repeated invite. The volumes of Eqs. (7 and 10-13) parameters are tabulated in Table 2.

The priority column identifies the next priority after response queue processing. If the priority is in the original invites queue, the method is named IO and if the same is in the repeated invites queue, named IR.

The N original invites produced based on Poisson distribution by accept and reject method. Goodput, the delay of a session creation, the mean number of original and repeated invites, the server utilization and fairness are considered as evaluation parameters. A session is completed when the ACK message is processed by the server and transmitted to UAS; accordingly, the Goodput is equal to the rate at which ACK messages exit the server. A session is considered successful, when ACK of corresponding invite is received less than 10 seconds [24].

The results obtained through the analytical model are compared with the results of the standard model, Fig. 1. The purpose of this comparison is to explore the main effect of prioritizing the messages of a session on the evaluation parameters. The NS-2 is applied to obtain the standard model volume of parameters. In the simulator, the SIP is implemented in accordance with RFC 3261. In this model, the classes are considered without priority, the processing is FIFO load-independent, and the server is state-full with a capacity of 100 cps. The UDP is applied as the transmission layer protocol. Also, the equivalent of the analytical model is implemented in the simulator, and compared with the evaluation parameters to verify the model validity and reliability.

The following two phases are considered in evaluation of the parameters: 1) all queues are empty at the beginning of evaluation, because there will be no overload until the mean input requests reaches the C and 2) twenty original invites are placed in the corresponding queue, in a sense that the server can experience the overload before reaching the C and determine how to return to normal status. The average of these two phases are repeated to achieve the required 95% confidence interval. In the charts, the offered load is the number of the calls per second, normalized by the C.

Table 2 Values of demands service and CPU utilization

Input Parameters	Values	Priority
$D_{Cpu,R}$	$0.2C$	--
$D_{Cpu,IO}$	$0.4C$	--
$D_{Cpu,IR}$	$0.3C$	--
$U_{Cpu,IO}^{Shw\_1}$	$X_{0,IO} * D_{Cpu,IO}^{Shw\_1}$	IO
$U_{Cpu,IR}^{Shw\_1}$	$X_{0,IR} * D_{Cpu,IR}^{Shw\_1}$	IR
$U_{Cpu,IO}^{Shw\_2}$	$X_{0,IO} * D_{Cpu,IO}^{Shw\_2}$	IR
$U_{Cpu,IR}^{Shw\_2}$	$X_{0,IR} * D_{Cpu,IR}^{Shw\_2}$	IO
$D_{Cpu,IO}^{Shw\_1}$	$\frac{D_{Cpu,IO}}{1 - U_R}$	IO
$D_{Cpu,IR}^{Shw\_1}$	$\frac{D_{Cpu,IR}}{1 - U_R}$	IR
$D_{Cpu,IO}^{Shw\_2}$	$\frac{D_{Cpu,IO}}{1 - (U_R + U_{IR}^{Shw\_1})}$	IR
$D_{Cpu,IR}^{Shw\_2}$	$\frac{D_{Cpu,IR}}{1 - (U_R + U_{IO}^{Shw\_1})}$	IO

5-1- The Sessions Good put and Delay

The Goodput and delay of the mechanisms under study are shown in Figs. (5 and 6). The efficiency of all mechanisms is almost the same when the offered load is less than C, Fig. (5). In the IR method close to C, the number of repeated invites increase, that is, the original invites processing is delayed due to priority, causing more than 10 seconds delay and unsuccessful sessions. This would lead to starvation of original invites, because, before the processing of repeated invites load reaches C, the already low Goodput reaches zero. When the offered load exceeds C, the IO method Goodput remains almost constant due to the priority of original invites. The Goodput of the normal mechanism becomes zero after a while because of an increase in the number of repeated invites, thus, a delay in the process of original invites.

5-2- The Average Number of the Original and Repeated Invite Requests on the Server

In the corresponding queue, the average number of the original invite requests is shown in Fig. (7). In all three methods, the mean number of original invites is the same as it approaches *C*. In the IO mechanism, this count is increased with a gentle slope by exceeding the *C* due to the priority and an increase in the number of original invites. In normal mechanism, the slope of the requests increase is sharper than that of the IO method, because as the number of the original invite requests increase, the repeated invite requests enter the FIFO queue, and delay original invites processing. Among the three mechanisms, the IR method has the highest number of unprocessed original invites in the queue. Because the offered load exceeds the *C*, the number of repeated requests increases, that is, the original invites queue cannot be processed yet, and the chart growth is severe.

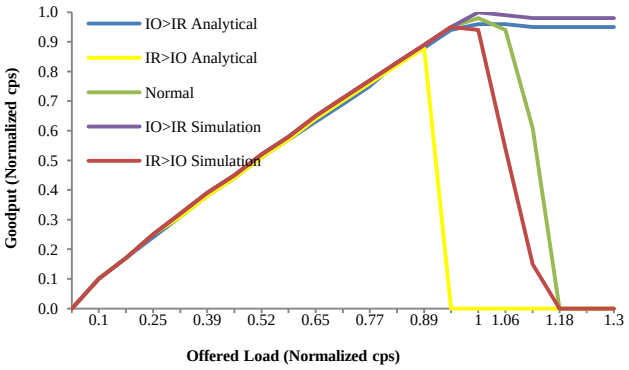

Fig. 5. Goodput of the mechanisms

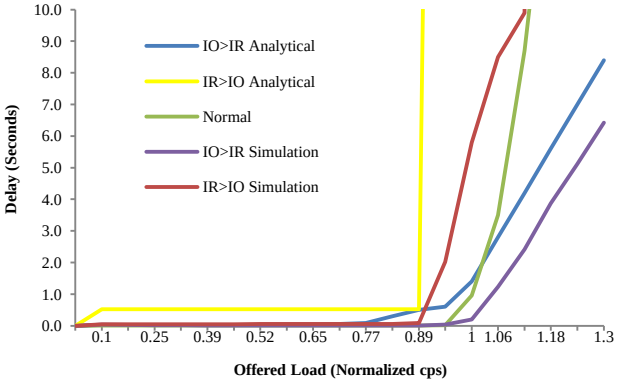

Fig. 6. Session delay of the mechanisms

In the corresponding queue, average number of repeated invites is shown in Fig. (8). The number of repeated requests in all three methods is almost the same before reaching the *C*. In IR, when the offered load exceeds *C*, the mean number of repeated requests increases. Because the repeated requests take precedence, and the original requests are processed with a long delay, the UACs constantly retransmit some repeated requests to the server, and this process continues. In the normal and IO methods, the mean number of repeated requests is lower, because the processing of the original requests took precedence, causing the UACs to transmit their corresponding repeated requests at a lower rate.

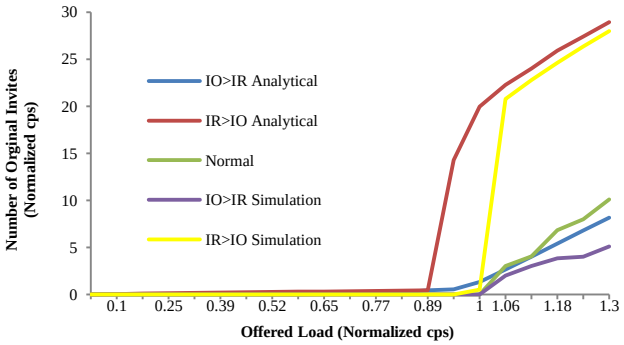

Fig. 7. Average number of original invites in the corresponding queue

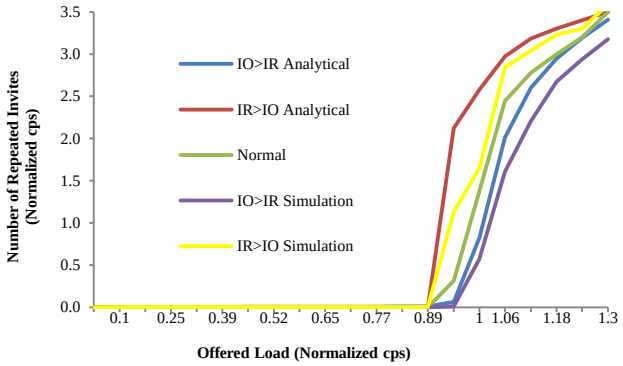

Fig. 8. Average number of repeated invites in the corresponding queue

5-3- Useful CPU Occupation

Processing of original invites is a useful task on the server CPU, while, the processing of repeated invites is a waste of capacity, respectively, Figs. (9 and 10). Among the subject mechanisms the one that makes better use of the capacity of server and reduce its waste outperforms others. The IO and IR mechanisms are such by increasing the offered load until it reaches the *C*, followed by the IR mechanism where the server capacity is wasted due to the processing more repeated requests than the IO mechanism. In normal mechanism, the CPU usage suddenly decreases because, likewise, the FIFO queue is filled by repeated invites after *C*.

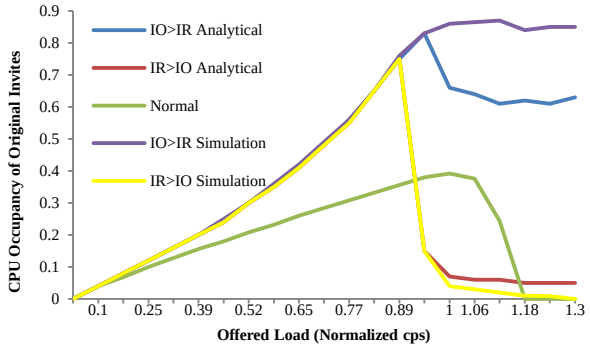

Fig. 9. Useful server occupation through the mechanisms

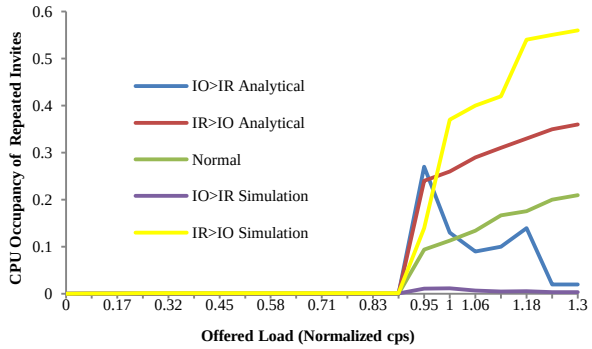

Fig. 10. Useless server occupation through the mechanisms

5-4- Fairness

The Jain's index, a statistical scattering measure is applied in measuring the average waiting time of original invites fairness, which is directly related to the number of calls made. This index with a volume within 0-1 is obtained through Eq. (26) [27].

$$J_w = \frac{(\sum_{i=1}^{m} w_i)^2}{k \sum_{i=1}^{m} w_i^2} \quad (26)$$

The closer the J_w rate to 1, the more the waiting time fairness and the opposite. W_i is the average waiting time of i^{th} offered load and K is the number of the applied offered load. The offered load range is within lower *C* to upper *C* range. The waiting time for M/M/1 queue is calculated through Eq. (27).

$$W_i = \frac{S * U_{CPU,k}^{Shw\_y}}{1 - U_{CPU,k}^{Shw\_y}} \quad (27)$$

The Jain's index of original invites waiting time with different priorities is shown in Fig. (11), when the priority is with IO queue, fairness is observed in the processing of original invites while priority to the IR queue acts contrariwise. Waiting time of original invites increases when IR queue is prioritized, and cause both the repeated invites retransmission and IO queue starvation. As observed in Fig. (5), there exists a direct relationship between the Goodput and the original invites waiting time fairness. In normal FIFO processing, waiting time of original invites increases for offered load up to C while less than C is acceptable.

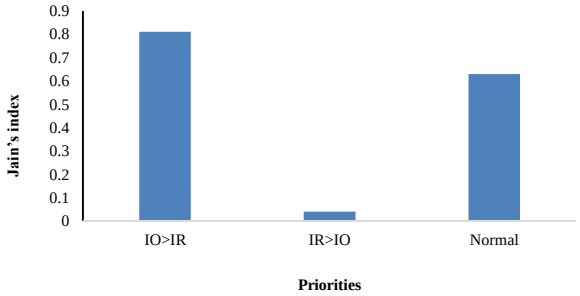

Fig. 11. Jain's index of waiting time of original invite

As to jitter, delay and package lost repeated processing of requests is not always in vain because this processing will not halt sessions. The fairness of repeated invites waiting time is shown in Fig. (12). In the process IR queue priority, is of high fairness and the same is observed in IO queue priority. In normal processing, fairness is almost the same for invites requests regardless of queue priority.

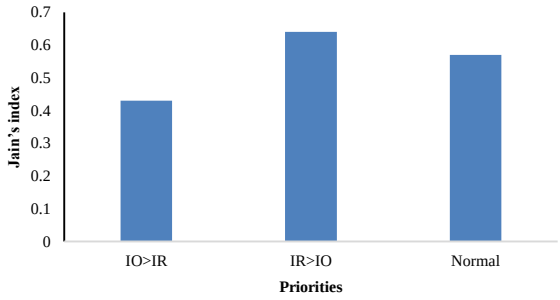

Fig. 12. Jain's index of waiting time of repeated invites

As can be deduced from the results, IO method is the most efficient, while, the IR method is more inefficient than the conventional SIP method and does not provide an appropriate response.

The results of the proposed analytical model are acceptable and close to the simulation results of Figs. (3 and 4). From comparison of the analytical and simulation results, it is deduced that the error percentage is acceptable and the results can be obtained with the time complexity of the O (n^2) where n is the number of original invites.

6- Conclusion and Future Works

Different messages are involved in creating the SIP session. If in the processing a message like invites is delayed, it would be retransmitted several times. In some cases, the number of these repeated messages are so big that no sessions are created or are delayed, where the result

is a considerable decline in the resource performance. The MVA algorithm of the queuing theory is customized to implement the proposed method. The question: at which priority should the messages involved in the session create be processed to cause less decline is assessed in this study. For this purpose, two scenarios are defined as follows: In the first scenario, priority is given to the original invites processing, because the rapid request processing and the prevention of retransmitting invites is evident. In the second scenario, priority is given to the repeated request processing, with the logic that this scenario would prevent retransmission. The results of these two scenarios are compared with the standard processing results of SIP messages, where priority is of no concern and are based on FIFS processing. As to the result, the mechanism for prioritizing original invite requests is more efficient than normal and IR mechanisms. The IR method performs poorly and does not provide acceptable answers. The simulation model confirms the analytical model reliability. As to the G/G/R queue and the corresponding algorithms can be replaced with M/M/1 discipline queue to model the proxy server. In multimedia networks, SIP tandem server is applied in establishing multimedia connection allowing the MVA algorithm to be run to model the consecutive proxy servers.

References

[1] D. Y. Yavas, I. Hokelek, and B. Gunsel, "Controlling SIP server overload with priority based request scheduling", International Conference on Computing, Networking and Communications (ICNC), 2015, pp. 510-514.

[2] I. Kuzminykh, "A combined LIFO-Priority algorithm for overload control of SIP server", International Conference on Modern Problems of Radio Engineering Telecommunications and Computer Science (TCSET), 2012, pp. 330-330.

[3] R. G. Garroppo, S. Giordano, S. Spagna, and S. Niccolini, "Queueing Strategies for Local Overload Control in SIP Server", In IEEE Global Telecommunications Conference, 2009, pp. 1-6.

[4] J. Lee and I. Joe, "An Overload Control Algorithm based on Priority Scheduling for SIP Proxy Server", Proceedings on the International Conference on Internet Computing, Athens 2012.

[5] K. K. Guduru and J. Usha, "Queuing strategies for self overload control in SIP servers", International Conference on Contemporary Computing and Informatics, 2014, pp. 1007-1011.

[6] A. Montazerolghaem, M. H. Yaghmaee, A. Leon-Garcia, M. Naghibzadeh, and F. Tashtarian, "A Load-Balanced Call Admission Controller for IMS Cloud Computing", IEEE Transactions on Network and Service Management, vol. 13, no. 4, pp. 806-822, 2016.

[7] A. Montazerolghaem, M. H. Y. Moghaddam, and A. Leon-Garcia, "OpenSIP: Toward Software-Defined SIP Networking", IEEE Transactions on Network and Service Management, vol. 15, no. 1, pp. 184-199, 2018.

[8] H. Kim and N. Feamster, "Improving network management with software defined networking", IEEE Communications Magazine, vol. 51, no. 2, pp. 114-119, 2013.

[9] P. Abaev, A. Pechinkin, and R. Razumchik, "Analysis of Queueing System with Constant Service Time for SIP Server Hop-by-Hop Overload Control", Berlin, Heidelberg, 2013, pp. 1-10: Springer Berlin Heidelberg.

[10] K. E. Samouylov, P. O. Abaev, Y. Gaidamaka, A. Pechinkin, and R. Razumchik, "Analytical Modelling And Simulation For Performance Evaluation Of SIP Server With Hysteretic Overload Control", In ECMS, 2014.

[11] Y. Gaidamaka, A. Pechinkin, R. Razumchik, K. E. Samouylov, and E. S. Sopin, "Analysis of an M|G|1|R queue with batch arrivals and two hysteretic overload control policies", International Journal of Applied Mathematics and Computer Science, vol. 24, pp. 519 - 534, 2014.

[12] V. K. Gurbani, L. J. Jagadeesan, and V. B. Mendiratta, "Characterizing session initiation protocol (SIP) network performance and reliability", presented at the Proceedings of the Second international conference on Service Availability, Berlin, Germany, 2005.

[13] S. V. Subramanian and R. Dutta, "A study of performance and scalability metrics of a SIP proxy server – a practical approach", Journal of Computer and System Sciences, vol. 77, no. 5, pp. 884-897, 2011/09/01/ 2011.

[14] G. Mishra, S. Dharmaraja, and S. Kar, "Performance analysis of SIP signaling network using hierarchical modeling", In Twentieth National Conference on Communications, 2014, pp. 1-5.

[15] S. Shorgin, K. Samouylov, Y. Gaidamaka, and S. Etezov, "Polling System with Threshold Control for Modeling of SIP Server under Overload", Cham, 2014, pp. 97-107: Springer International Publishing.

[16] Y. V. Gaidamaka, "Model with threshold control for analyzing a server with an SIP protocol in the overload mode", Automatic Control and Computer Sciences, vol. 47, no. 4, pp. 211-218, 2013/07/01 2013.

[17] Y. Hong, C. Huang, and J. Yan, "Modeling chaotic behavior of SIP retransmission mechanism", International Journal of Parallel, vol. Emergent and Distributed Systems, 02/01 2013.

[18] Y. Hong, C. Huang, and J. Yan, "Modeling and simulation of SIP tandem server with finite buffer", ACM Trans. Model. Comput. Simul., vol. 21, no. 2, p. Article 11, 2011.

[19] M. Jahanbakhsh, S. V. Azhari, and H. Nemati, "Lyapunov stability of SIP systems and its application to overload control", Computer Communications, vol. 103, pp. 1-17, 2017/05/01/ 2017.

[20] A. Montazerolghaem, M. H. Yaghmaee Moghaddam, and F. Tashtarian, "Overload Control in SIP Networks: A Heuristic Approach Based on Mathematical Optimization", 2015.

[21] V. A. F. A. Daniel A. Menascé, Lawrence W. Dowdy, Performance by Design: Computer Capacity Planning by Example. Prentice Hall PTR, 2004.

[22] M. Khazaei and N. Mozayani, "A dynamic distributed overload control mechanism in SIP networks with holonic multi-agent systems", Telecommunication Systems, vol. 63, 12/30 2015.

[23] M. Khazaei, "Occupancy Overload Control by Q-learning", Singapore, 2019, pp. 765-776: Springer Singapore.
[24] J. Wang, J. Liao, T. Li, J. Wang, J. Wang, and Q. Qi, "Probe-based end-to-end overload control for networks of SIP servers", Journal of Network and Computer Applications, vol. 41, pp. 114-125, 5// 2014.
[25] J. Liao, J. Wang, T. Li, J. Wang, J. Wang, and X. Zhu, "A distributed end-to-end overload control mechanism for networks of SIP servers", Comput. Netw., vol. 56, no. 12, pp. 2847-2868, 2012.
[26] C. Shen, H. Schulzrinne, and E. Nahum, "Session Initiation Protocol (SIP) Server Overload Control: Design and Evaluation", Berlin, Heidelberg, 2008, pp. 149-173: Springer Berlin Heidelberg.
[27] M. Khazaei and N. Mozayani, "Overload management with regard to fairness in session initiation protocol networks by holonic multiagent systems", International Journal of Network Management, vol. 27, no. 3, p. e1969, 2017.

Mehdi Khazaei received a B.Sc degree in computer Engineering (Computer Hardware) from Iran University of Science and Technology (Tehran, IRAN); M.Sc. and Ph.D degree in computer systems Architecture from Iran University of Science and Technology (Tehran, IRAN) in 2017. He is currently assistant professor in the School of Information Technology at Kermanshah University of Technology (Kermanshah, Iran).