SRTTU

Journal of Electrical and Computer Engineering Innovations (JECEI)

Journal homepage: http://www.jecei.sru.ac.ir

JECEI

Research paper

Reinforcement Learning-based Load Controller in IP Multimedia Subsystems

M. Khazaei*

Computer Engineering Department, Kermanshah University of Technology, Kermanshah, Iran.

Article Info

Article History:
Received 29 January 2022
Reviewed 07 March 2022
Revised 22 April 2022
Accepted 01 May 2022

Keywords:
IMS
SIP
Overload
Multi-agent system
Reinforcement learning

*Corresponding Author's Email Address: *m.khazaei@kut.ac.ir*

Abstract

Background and Objectives: IP multimedia subsystems (IMS) have been introduced as the Next Generation Network (NGN) platform while considering Session Initiation Protocol (SIP) as the signaling protocol. SIP lacks a proper overload mechanism. Hence, this challenge causes decline in the multimedia QoS. The main propose of overload control mechanism is to keep the network throughput at the same network capacity with overload.
Methods: NGN distributed with IMS is a complex innovative network consisting of interacting subsystems. Hence, multi-agent systems (MAS) receiving further attention for solving complex problems can solve the problem of overload in these networks. To this end, each IMS server is considered as an intelligent agent that can learn and negotiate with other agents while maintaining autonomy, thus eliminating the overload by communication and knowledge transfer between the agents. In the present research, using MAS and their properties, the intelligent hop by hop method is provided based on Q-learning and negotiation capability for the first time.
Results: In the proposed method, parameters of overload controller are obtained by reinforcement learning. In order to check the validity of controller performance, a comparison is made with the similar method in which the optimal parameters are achieved based on trial and error. The result of the comparison confirms the validity of the proposed method. In order to evaluate the efficiency of the learner method, it is compared with similar and standard methods, for which the results are compared to show performance. The results show, the proposed method has approximately improved the throughput by 13%, the delay by 49% and the number of rejected sessions by 17% compare with methods, passing control messages through the network such as CPU occupancy methods. While compare with external controller methods like holonic, throughput is improved by 1% and the number of rejected requests is decreased by 10%, but delay is increased by 6% due to the convergence time of the learning and negotiation process.
Conclusion: To overcome overload, complex IMS servers are considered as learner and negotiator agents. This is a new method to achieve the required parameters without relying on expert knowledge or person as well as, heterogeneous IMS entities can be inserted into the problem to complete study in future.

Introduction

Packet switching allows service providers to provide multimedia applications to their subscribers. The future approach of the telecommunications industry is towards NGN, in which all types of fixed and mobile access networks are integrated based on the IP platform. NGN can provide multimedia services based on the standard IMS platform. IMS has been used by most mobile network operators since the third generation. In this regard, the SIP protocol has been adopted by 3GGP as the basic IMS architecture. Most cell phones and wireless devices support SIP as a multimedia session protocol [1]-[3].

SIP lacks a proper overload mechanism despite its positive features such as text-based, IP-based, data-independent, relocation support, and end-to-end properties. It uses the retransmitting mechanism to deal with packet loss when running on the UDP protocol. This is done by keeping different timers for each request to be sent. When the relevant response is not received within a specified time, the requests will be resent, which makes the situation worse in case of overload [4].

When the SIP encounters an overload, the overload control mechanism is activated rejecting the new session request messages by sending a 503 response. In this method, the messages must first be analyzed to generate a rejection response for new requests. The capacity devoted by the server to analyze messages and generate reject responses is wasted. All the server capacity is spent on rejecting repeated requests in addition to the negative effect on the overload. Therefore, it is inevitable to design an efficient overload controller for IMS. The overload controller must be able to gather the information needed to decide about overload. Based on this information, it must determine the appropriate response to the overload and apply it to the network. Depending on the type of overload and the location of the controller, different methods and policies have been proposed to deal with the overload [5].

Due to the complexity and heuristic nature of the overload problem in IMS, approximate mathematical or heuristic methods are used in designing the controller leading to occasionally a huge deal of errors and not acceptable answers. Therefore, in this paper, a new machine learning method is proposed to design an overload controller. In this regard, IMS servers are considered as intelligent learning agents able to negotiate with other network agents. These agents learn the amount of tolerable load by interacting with the dynamic environment and through unsupervised trial and error, and they control the overload in the network by negotiating with other agents.

In the following, the background and related works are given. Then, the proposed method is presented and the performance evaluation and analysis of the results are provided. Finally, deals with conclusions are presented.

Related Backgrounds

Since the papers' objective is to design an overload controller based on intelligent agents in IMS, it is essential to explain the related concepts and works briefly.

A. Multi-Agent Systems (MAS)

An agent is defined as a software or hardware located in the environment and acts autonomously to achieve its goals. The agent perceives the environment through its sensors and affects the environment through its stimulants. Everything around the agent except itself is called the agent environment. Today, the use of MAS to solve complex problems has received a huge deal of attention. MAS is made up of several agents trying to solve problems that are sometimes difficult and sometimes impossible to solve for a centralized and integrated system. On the other hand, communication is an important concept in MAS. Without communication, agents should rely only on decisions based on their observations, while communication enables agents to make more coordinated decisions. Negotiation is the dominant way to reach an agreement without the involvement of others to gain mutual benefit in common areas of interest to agents. A negotiation protocol is a set of rules that all agents know. In negotiation, an agreement is obtained when there is a common ground between the proposals. Agents have a personal preference for the outcome of the negotiation and may also have limitations on the use of the proposals [6], [7].

In learning the agents, there are problems for which scarce or incomplete resources exist for solving, thus, unsupervised learning has been considered. Reinforcement learning is unsupervised learning in which the system tries to optimize interaction with the dynamic environment through trial and error without specifying the action for the agent. Reinforcement learning is indeed to map different situations into actions to get the best results with the most reward. The two characteristics of "trial and error" and "reward with delay" are the most important characteristics of reinforcement learning [8].

Standard reinforcement learning, or Q-learning, is a model-independent method, in which the agent has no access to the transfer model. Q-learning based on Markov's decision-making process, with a delayed reward function, can learn the optimal policy. In this method, the agent estimates the pair (action, state) by continuous interaction with the environment as trial and error. The Q-learning steps are based on Algorithm 1.

The algorithm starts from an initial state and reaches the goal state by performing a series of actions and receiving a reward. In this situation, the agent state is not changed with each action while not receiving any reward from the environment. The action can be selected by exploration and exploitation. Selection of action by exploration means selecting the action randomly regardless of the values in Q-Table. This may discover optimized actions not selected yet and add them to the table. In the exploitation form, the best action is selected based on Q-Table [9].

Algorithm 1: The Q-learning method algorithm

```
Begin
Q[states][actions]=0;
S=Get_Current_State ();

While (S! = absorption state)
    a= Select_Action ();
    r= Calculate_Reward ();
    s/= Get_New_state ();
    max=For_All_Possible_Action_Find_Max_Q (s/, a) ();
    Q (s, a) =α*(r+max) +(1-α) *Q (s, a);
    Update_Q_Table ();
    S= s/;
End While
End
```

In the Q-learning algorithm, $\alpha \in [0, 1]$ is the agent learning rate. A value of one causes the agent to consider only the most recent information, while zero leads the agent to have no learning. Parameter $\gamma \in [0, 1]$ is called the discount factor. Zero means that the agent only considers the current reward, however, the values close to one cause the agent to wait a long time to reach a higher reward. When all pairs (action, state) are experienced repeatedly while reducing the learning rate over time, Q-learning converges with a probability of one to the optimal value of $Q^*(s,a)$. Some applications of reinforcement learning include continuity of services in IMS [10], [11], unsupervised learning in next-generation networks [12], [13], intelligent transportation systems and urban traffic control [9], [14], [15], management of wireless distributed sensor networks [16]-[18], and complex software systems [19].

B. IP Multimedia Subsystems (IMS)

Unlike traditional applications, the purpose of IMS is to integrate different types of multimedia services and applications and converge between wired, wireless, and mobile networks. Other features of IMS are control of sessions, development of services, Quality of services (QoS), and the possibility of calculating costs under a single standard. IMS is a packet switching network extending over the IP platform and has a three-tier architecture. The users connect to IMS using IP-based networks. The simplified IMS core architecture is shown in Fig. 1 [20], [21].

There are two databases in the IMS architecture including Home Subscriber Server (HSS) and Subscription Locator Function (SLF). HSS is the place for storing subscribers' information and related services, and SLF is used to find subscribers' HSS addresses. ASs provide multimedia value-added services. Call Session Control Functions (CSCF) are SIP proxy servers each with a specific function. Their common role is during the registration, session creation, and routing process.

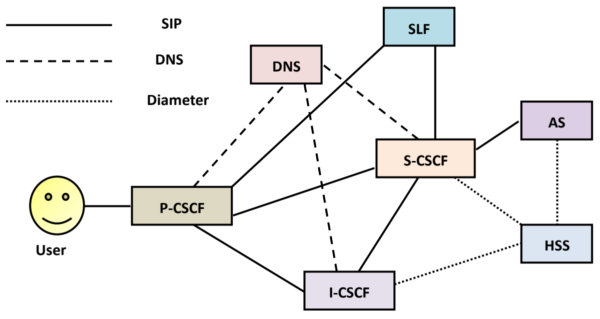

Fig. 1: The IMS core architecture.

The servers can be configured stateful or stateless, depending on the situation and needs. The stateful server stores transaction information, however, in the stateless server, no transaction is created on the server, and the server is solely responsible for receiving and routing messages. P-CSCF is a stateful SIP server. This server is the first point of users' contact with IMS. All user traffic is transferred to this server and also network traffic is transferred to users through this server. S-CSCF is also a stateful SIP server always located on a home network domain. This server is the central point of the IMS responsible for managing the registration process, routing, maintaining session status, and storing service information. All SIP signaling packets pass through the S-CSCF to determine the next action by processing. I-CSCF is a stateless SIP server, contacting point of an operator to connect with subscribers within that operator. I-CSCF allocates an S-CSCF server based on the defined policies when registering by receiving information from the HSS. Another task of this server is to get the next step in routing through HSS as well as directing requests to the assigned S-CSCF or AS.

While receiving the IP, the user receives the P-CSCF address and has to register in the IMS. After registration, the relevant P-CSCF knows the S-CSCF assigned to each user according to the response package. The S-CSCF also knows the P-CSCF to contact to reach the user. This information is used to establish the sessions.

As shown in Fig. 2, when user A wants to make a session with user B through SIP, it sends an Invite to P-CSCF. P-CSCS sends the packet to the S-CSCF assigned to A in the registration process. Based on ID B, S-CSCF finds the corresponding I-CSCF in domain B and delivers the package to it.

By contacting the HSS, the I-CSCF finds the S-CSCF assigned to B and delivers the package to it. The S-CSCF, with the information obtained at the time of registration, delivers the Invite package to the P-CSCF and then to the B. After receiving Invite by B, the response is generated and sent to A of the same route as Invite reached. After exchanging messages, a session is formed between A and B [1].

A SIP transaction is a request and all related responses exchanged between two adjacent SIP entities. Since in most cases the 503 mechanisms embedded in the SIP cannot cope with overload, an overload controller is inevitably required in IMS. The overload controller consists of three main components. The monitoring unit collects information from the specified parameters and provides them to the control function. The control function determines the policy and the amount of load received based on a defined algorithm and provides it to the stimulator unit, which rejects the overload based on the policies received [22], [23].

If all the controller components are on one server, this is called the internal method otherwise, it is termed external control. External controls are divided into two end to end and hop by hop categories.

In the end to end method, the edge server is responsible for regulating the load sent to the overloaded server. The challenge in this method is how to inform the edge server about the server with overload and how the edge server detects the request passing through the overloaded server. In the hop by hop method, two tandem servers determine the amount of load sent from the server upstream to downstream by different policies [24], [25].

In the window policy, the downstream server allows the upstream server to send the request in the specified window size without receiving confirmation. The size of the window on the upstream server can be determined using incoming messages, acknowledgments, 503 messages, timers expire, or calls delay. The overloaded server can also dynamically and continuously estimate its response capacity and notify upstream servers as the number of windows available [24], [26]-[29]. A window-based holonic mechanism (WHOC) is used holonic multi-agent system to control and manage overload in SIP networks.

Based on past observations, the normalized least mean square algorithm is used to estimate each agent window size. The size of the windows is adjusted in the way that no overload will occur in network paths, which could be fulfilled through using holonification properties, negotiation process and communications. WHOC offers an appropriate window size for edge servers to control the load from the beginning of the network and prevent network overload [30].

In the loss-based policy, the downstream server specifies the percentage of reduced load sent by the upstream servers. In this method, the rate of retransmissions can be reduced by modeling the interactions between the downstream and upstream servers as a controller [31]. Also, sometimes the upstream server predicts overload on the downstream server by methods such as 503 received message rates or uses mechanisms such as the leaky-buckets technique [32], [33]. Intelligent methods are provided to monitor the overloaded server and then prevent overload by classifying packets, intelligently deleting repeated packets, controlling active sessions and obtaining thresholds [34], [35].

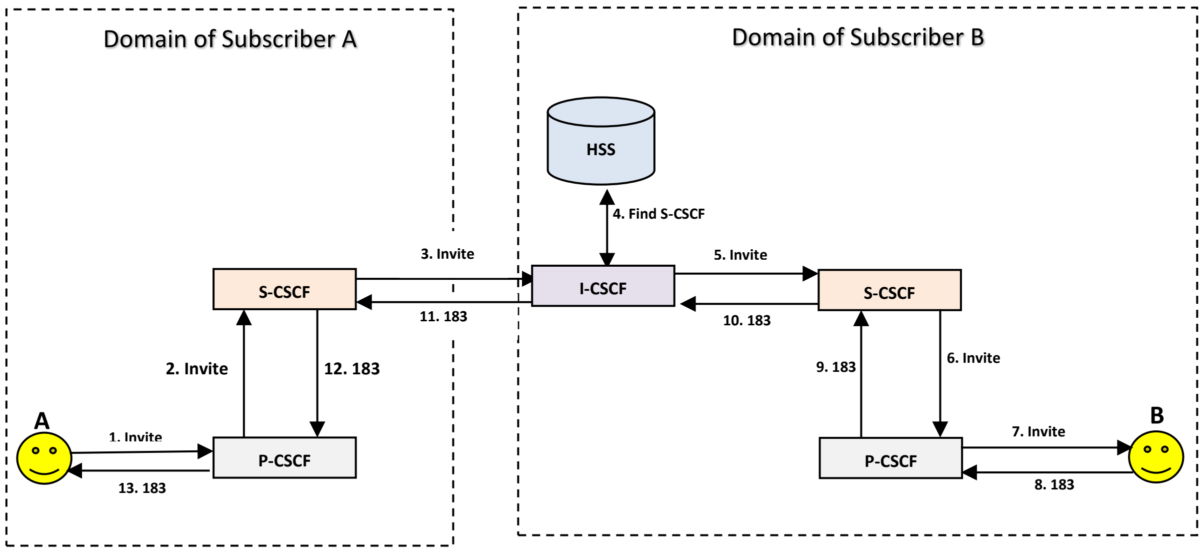

Fig. 2: Establishing a session between two users in two different IMS domains.

In HOC, a loss-based mechanism is implemented to control overload, using multi-agent with holonic organization to implement a user perspective of fairness if possible. HOC uses a greedy method on the network graph to obtain constant holarchy. When a proxy server is overloaded, the sending load is adjusted from the source servers, causing fewer network resources to be involved in the overload. Load fitting is done based on received requests and used as predictor. Therefore, each holon uses the server capacity amount of its offered load. In HOC, when a holon involved in overload is recognized with a request from the due holon, the overload is most likely solved because servers with high dependency are placed in a holon in terms of load exchange with each other. Complexity decreased by the use of holonic organization and cooperation between holons through the communication, agreement and knowledge that is exchanged among them [36].

In the rate-based policy, the rate of sending the upstream server to the downstream server is limited. Setting a threshold for CPU consumption and sending this value at regular intervals is one of these techniques controlling the rate to enter the requests into the downstream server [5], [24], [37]. In the on-off policy, the downstream server can stop or connect the received load for a while. Determining the time required to process messages within the downstream server queue and announcing the downtime to the upstream server is among the methods of this policy [24].

Finally, a hop by hop method can be developed based on new policies, especially the concept of software networks, Network functions virtualization (NFV), and cloud environments [38]-[40].

Designing Load Controller

In many IMS load control algorithms, there are parameters determining the efficiency of the algorithm. For example, in the proposed algorithms [5] a threshold value is considered for the amount of CPU occupation. Whenever the percentage of CPU occupation exceeds this threshold, the overload control algorithm will operate. Method [28] also determines a delay threshold in the upstream server. It determines the window size in the upstream server by comparing the delay of the received responses with this threshold. The best and most accurate way to calculate the values of strategic parameters is to use mathematical relations. Nonetheless, mostly due to the complexity or heuristic of the method, it is not possible to calculate the desired values using mathematical equations [41]. Trial and error is another way to determine the values of the parameters and check the value producing the most optimal answer by simulating and placing different values. The problem with this method is that the obtained values are only suitable for that network situation, and by changing the network conditions, the answers are no longer acceptable and the calculations must be performed again. Learning-based methods help in such situations. According to Fig. 2, IMS has three tandem SIP servers (S-CSCF, I-CSCF, S-CSCF) in which the I-CSCS server is stateless not contributing to overload. However, the two tandem servers are stateful and play a role in organizing and managing the sessions. Since the servers connected through one hop, the proposed controller used the hop by hop overload control [42].

As seen in Fig. 3, for each proxy server queue, two warning (T_w) and constraint (T_c) thresholds and thus three regions are defined determining the decision and controller response. If the number of session requests exceeds T_c (constraint region), requests will be rejected locally by messages 503. If the number is between two thresholds (warning region), the negotiation process with the upstream server begins. Nevertheless, if the number of requests is less than T_w, no reaction occurs.

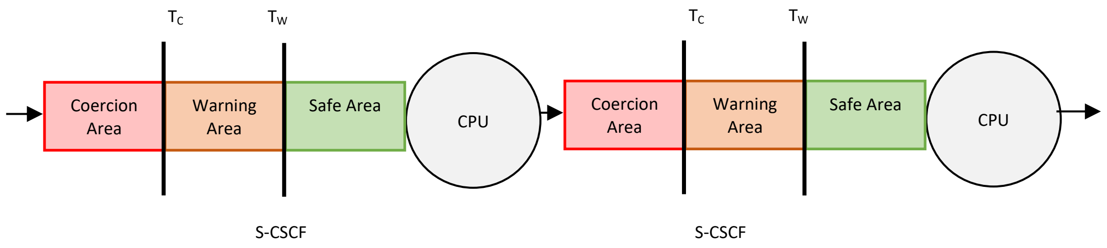

Fig. 3: Two tandem servers in IMS.

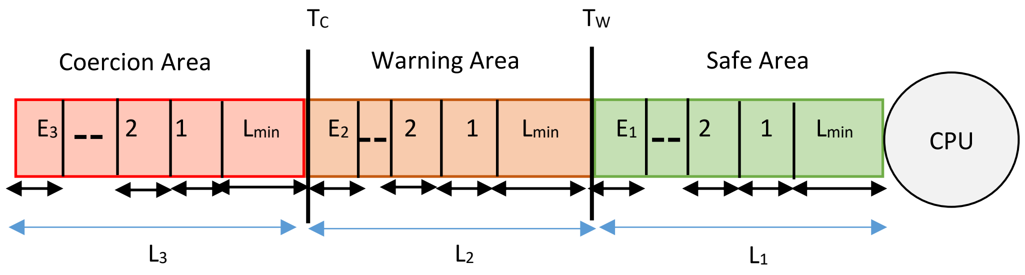

Fig. 4: Assigning states to the agent queue.

In this regard, each proxy server is defined as an intelligent agent with the ability to learn and negotiate. Agents learn the values of T_w and Tc by Q-learning through monitoring their resources and the knowledge from the environment. Q-learning allows the controller to be independent of expert knowledge and prior environmental information. Among the requisites for designing a Q-learning algorithm is the definition of states space, actions space, and appropriate reward function. In the designed controller, the control function implements the learning algorithm, the stimulator unit negotiates with the upstream agent, and the monitoring unit collects information about the queue length and the rate of incoming and outgoing requests. In the proposed controller, the monitoring unit, the control function, and the stimulator unit are located in the agent.

A. Defining Agent Q-Learning Requirements

The actions defined for each agent determine how the constant queue length is divided into each region. To define each action by the agent, a minimum constant length is assigned to each region. Moreover, a certain number of length increases are considered as an extension. The action space is defined by <N_a, L_{min}, N_{ex}, L_{ex}>. N_a indicates the number of regions, L_{min} represents the minimum length assigned to each region, N_{ex} denotes the number of extensions, and L_{ex} indicates the length of each extension. Server queue length (L) is obtained from (1).

$$L = N_a * L_{min} + N_{ex} * L_{ex} \qquad (1)$$

The length assigned to each region is obtained from (2). In Fig. 4 the length of each region is equal to the sum of the minimum initial lengths and the number of allocated extensions

$$L_i = L_{min} + e_i * L_{ex} \qquad (2)$$

where, e_i is the number of extensions of the ith region. Since the sum of the extensions considered for all 3 regions is constant, the method of allocating the extensions to each region is calculated by (3) [14].

$$\sum_{i=1}^{N_a} e_i = N_{ex} \quad ; \quad e_i \in N \qquad (3)$$

The answer of (3) specifies the number of actions of each agent and depends on the value of N_{ex}. As the N_{ex} increases, the number of responses, and consequently the number of actions of each agent increase. With (4), the number of extensions of each region can be controlled by the parameter θ to reduce the action space to accelerate convergence [14].

$$\sum_{i=1}^{N_a} e_i = N_{ex}\,; \quad e_i \in N; \quad e_i \le \theta \quad ; \quad 1 \le \theta \le N_{ex} \qquad (4)$$

The values of the parameters used to determine the actions space are given in Table 1. According to these values, the set of actions is 19 actions, which are given in Table 2. The actions are selected through an exploration.

To determine the states space, the number of transactions in each region is used. In this regard, the number of transactions in each region is arranged according to their order of entry and each arranged list corresponds to a state. The total number of states Table 3 is equal to the sum (ten states) of the inequality permutations of the number of areas transactions (six states) plus the possibility of equalizing the number of areas transactions (four states). For example, if T_i indicates the number of transactions in the ith area, T_1> T_2> T_3 will represent the highest number of transactions in the safe region and the lowest number of transactions in the constraint region.

Table 1: The parameters used in determining the actions space

Parameters	Volumes
L	110
L_{min}	20
L_{ex}	10
N_{ex}	5
N_a	3
Ɵ	5
|Actions|	21
|States|	13

Table 2: The values of the length assigned to each region for possible actions

Actions					
Normal area	Warning area	Coercion area	Normal area	Warning area	Coercion area
20	70	20	30	60	20
20	60	30	30	50	30
20	50	40	30	40	40
20	40	50	30	30	50
20	30	60	30	20	60
20	20	70	50	40	20
40	50	20	50	30	30
40	40	30	50	20	40
40	30	40	60	30	20
40	20	50	60	20	30
70	20	20			

Table 3: The different states defined for each agent

States	
Inequality States	Equality States
T_1>T_2>T_3	T_1=T_2>T_3
T_1>T_3>T_2	T_1=T_3>T_2
T_2>T_1>T_3	T_2=T_3>T_1
T_2>T_3>T_1	T_1=T_2=T_3
T_3>T_1>T_2	
T_3>T_2>T_1	

The directing of the agent in the states space and actions to achieve an optimal policy is done by the reward function. The reward function determines how much closer an agent is to a goal through a state. In the proposed learning method, the concept of Goodput is used as a reward function.

The amount of reward received by an agent for acting is proportional to the amount of increase given by Goodput compare with the previous states. Hence, the values in Q-Table are updated with a delay step [35], the reward function for the ith agent is predicted by applying the Normalized Least Mean Square method (NLMS), Table 4.

Table 4: The agents' reward value prediction by NLMS

Number	Equations
1	$R_{n-2} = G_{n-1} - \bar{G}_{n-1}$
2	$\underline{W_n} = \underline{W}_{n-1} + \omega * \frac{R_{n-2} * \underline{G}_{n-2}}{\lVert \underline{E}_{n-2} \rVert 2}$
3	$\underline{G}_{n-1} = combine G_{n-1} and\ \underline{G}_{n-2}$
4	$\overline{G}_n = \underline{W}_n^T * \underline{G}_{n-1}$

In Table 4, the underlined variables are vectors and the over-lined ones are to hold the predicted values. The **W** is the vector of prediction coefficient filter at size *q* and **G** is a vector to hold the *q* value of process reward. The initial value of **W** is zero which can be updated per each new data. $\overline{G}_n$ is predicted reward of G_n.

Q-Table is used during the learning process to store and update Q-function values. This table is considered as a two-dimensional matrix in which rows and columns specify states and actions, respectively. The initial value of Q-Table is considered zero. After the learning stage, according to the values of Q-Table, the thresholds T_w and T_c of each agent are extracted based on the length of the regions [30], [36].

B. Negotiation Protocol for Implementing the Policy

In negotiation, a set of agents is involved along with a set of variables dependent on agents. Agents negotiate a set of possibilities (values). To reach an agreement, the possibilities are assigned to the variables through negotiation. In the controller, the set of agents participating in the negotiation are the agents in Fig. 3. The variables define the amount of load sent to the downstream agent. The possibilities are also the values suggested by the agents to obtain the amount of load sent or received.

The agents will participate in the negotiation based on a defined strategy presented in Table 5. A suggestion cycle includes the initiator suggestion and the response of other agents to it. The initiator is the downstream agent initiating the negotiation process by passing the load through T_w, while the respondent is the upstream agent.

Table 5: The negotiation protocol to reduce the load

1	*K=1*
2	The downstream agent (j) asks the upstream agent (i) to reduce the number of requests sent in the period Δt based on R_{ij} according to (5). $R_{ij} = \frac{(load_j - Tw_j)}{\Delta t} * \left(1 - CPU^j_{Occupancy}\right) * CPU^j_{Capacity}$ (5) Load is the number of sessions within the queue.
3	If agent i is in the safe region, it calculates the number of requests that it can process during Δt while not leaving the safe region according to (6) and notifies agent j. Otherwise, agent, i asks the upstream agent (P-CSCS) to reject the request as $R_{ij} + R_{pi}$ randomly. $D_{ij} = \frac{(TW_i - load_i)}{\Delta t} * \left(-CPU^i_{Occupancy}\right) * CPU^i_{Capacity}$ (6)
4	Agent j receiving the answer of agent i, rejects the request locally and randomly as R_{ij}-D_{ij}. If it enters the safe region, it sends the value $R_{ij} = 0$ to agent i and the negotiation ends. Otherwise, agent i recalculates the load reduction rate according to (7) and sends it to j. $R_{ij} = \left(\left(\sum_{k=1}^{I} R^k_{ij} - D^k_{ij}\right)\right) * \beta * I$ (7) Where, *β* is the reduction coefficient and *K* is the number of negotiations.
5	*K=K+1*
6	The above process is repeated until the end of the negotiation.

Results and Discussion

The proposed method is implemented based on RFCs 3261 and 6026 in NS-2 (2.34). NS-2 is run on the same software and hardware platform to compare the studied mechanisms (Fedora Linux 20, Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz 4.00 GHz, Cache Size 8.0 MB, Install RAM 16.0 GB). UDP is considered as the transmission layer protocol. The servers have a processing capacity of 300 sessions per second (SPS). The users have unlimited capacity and can send or receive multiple session requests at the same time. The priority of processing agents is to negotiate messages because failure to process these messages on time causes an overload on the network. If the queue is full when receiving a request, the request will be deleted.

Goodput, sessions delay, number of rejected sessions, stability, and rapid response are the main criteria for evaluating the performance of Reinforcement Learning Overload Controller (RLC).

Goodput is the number of successful sessions that the agent handles per unit of time. A session is successful, which is created in less than 10 secs. A session delay is a time for creating a session. Session response time is exponential with average of 30 secs. Stability means that the overload controller should not cause throughput fluctuations on the proxy servers and prevent the Goodput from being zero. On the other hand, by the sudden reduction in the traffic imposed on the server, all the applied controls should be removed quickly and the situation should return to the normal state [25], [43].

The offered load was entered into the network as a Poisson distribution by the acceptance and rejection method. The performance evaluation criteria were extracted with a confidence interval of 95%. In the diagrams and tables, the offered loads were divided by the network capacity and normalized. The parameters used in the Q-learning algorithm were shown in Table 6. To obtain the values of T_w and T_c, different input modes with consecutive performances are considered and then the values of these two thresholds are used in RLC.

Table 6: The values of parameters in Q-learning

Parameters	Values
α	0.7
Learning rate	0.9 Decreasingly
Discount factor (γ)	0.4
Exploration rate	0.7

A. *Evaluating the Accuracy of RLC Performance*

Table 7 shows the average and variance of Goodput, sessions delay, and the number of rejected sessions of RLC, and Overload Controller (OC) when the average number of different input sessions is more than the network capacity; otherwise T_w and T_c have no role in the normal operation of the network. The optimal value of T_c is 37 and the optimal values of T_w are 13, 17 and 25, obtaining as trial and error for OC. In Table 7, the columns of improvement show the percentage which RLC improve performance relative to selected thresholds. In OC, thresholds have better performance, providing better average and less variance. Goodput at T_w= 17 has better performance, T_w = 17 performs better at delaying sessions, and T_w = 25 rejects sessions more efficiently. Since the rejects sessions are approximately equal for T_w=17 and T_w=25, the values T_w = 17 and T_c = 37 are chosen to compare the performance of RLC with OC. Therefore, Goodput is improved through 1.25%, sessions delay is decreased through 3% and number of rejected sessions is reduced through 1.07% by RLC compare with selected OC.

B. *Performance Evaluation of RLC*

RLC performance is compare with the known overload control methods of CPU occupancy end to end (EOCC), CPU occupancy hop by hop (HOCC) and Holonic overload control (HOC) [5], [36]. The reason for choosing these methods for comparison is 1) They are well-known and include standard codes 2) They have been used in many studies to compare performance, and 3) The end to end methods such as EOCC and HOC have better performance than hop by hop methods. Therefore, comparison with these methods can be a good benchmark to test RLC. Since there is no overload, when the offered load is less than the network capacity, comparisons are only made for offered load more than network capacity. Table 8 shows the improvement of the RLC over the compared methods.

Goodput is shown in Figs. 5. In HOCC, when the downstream S-CSCF is overloaded, it notifies its upstream S-CSCF. Upstream S-CSCF receives this message to reduce the load on the downstream S-CSCF, however, it continues to send since P-CSCF has no knowledge causing overload in the S-CSCF upstream and eventually the entire network.

Table 7: Checking validity and accuracy operation of the RLC

		RLC	T_w=13	Improvement	T_w=17	Improvement	T_w=25	Improvement
Goodput	Average	0.963	0.942	2.2%	0.951	1.25%	0.944	1.97%
	variance	1.571	1.611	2.48%	1.600	0.68%	1.710	8.13%
Sessions delay	Average	0.291	0.301	3.32%	0.300	3.00%	0.311	6.43%
	variance	0.022	0.043	48.8%	0.032	31.3%	0.033	33.3%
Number of rejections	Average	1.022	1.051	2.76%	1.033	1.07%	1.031	0.87%
	variance	298.4	317.1	5.89%	301.7	1.09%	300.6	0.73%

Table 8: RLC performance compare with studied mechanisms

RLC Compare with	HOC	EOCC	HOCC
Goodput	0.6%	13.5%	40.4%
Session delay	-6.7%	49.17%	62.35%
Number of rejections	9.4%	17.3%	31.04%

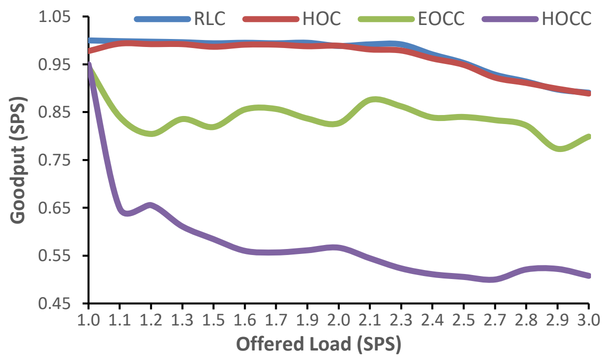

Fig. 5: Goodput for the studied mechanisms.

In EOCC, the probability of acceptance of sessions is adjusted so that the amount of CPU consumption is less than 90% so under heavy load, 10% of CPU capacity is wasted. Hence, the servers do not process at full capacity and Goodput is never exactly equal to *C*. In HOC, CPU capacity is fully utilized. In addition, overload occurrence is prevented in the proxy servers by holons, and the sending load is adjusted from the edge servers. Thereafter, the proxy servers are not overloaded. In RLC, whenever it detects that one of the agents leaves the safe region, it reacts quickly and tries to prevent overload from entering the network by preventing the additional load from entering the source. In RLC, the average of Goodput is almost 0.6% more than HOC because of the hierarchical structure of the HOC, it reacts more slowly than RLC. Goodput of RLC is 13.5% more than EOCC also 40.4% more than HOCC.

The results of sessions delay are shown in Fig. 6. Due to the local view of the overload, the HOCC spends some of the server capacity on processing requests that are eventually deleted, causing delays the processing of other requests. In EOCC, the probability of accepting load from the destination server to the source servers is reported. Upon receiving the restrictions, each server changes its information and sends the updated values to the source servers. As a result, updating the parameters and sending the appropriate amount of load to the destination take time, inadvertently causing the additional load to enter the network and delay the establishment of sessions. HOC has smaller sessions delay (average is 0.13 secs). In HOC, the retransmission mechanism is rarely activated due to keeping the servers in the safe section and not permitting the extra load to enter the network. The due delay exists because of corresponding holon calculation. RLC starts negotiations with the upstream server as soon as the load passes through the secure region. However, in RLC, negotiation process delay is added to RLC delay therefore, sessions delay is 6.7% more than HOC. But RLC sessions delay is 49.17% less than EOCC and 62.35% less than HOCC. Fig. 7 shows the total number of the rejected sessions. By increasing the number of rejected sessions, the network resources are spent to process the requests with no results.

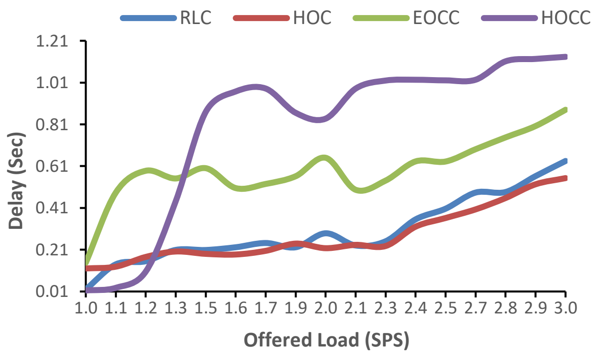

Fig. 6: The sessions delay for the studied mechanisms.

When the number of input sessions is equal to the network capacity, OCC methods reject the sessions due to the CPU consumption threshold of 90%. HOCC reject 31.04% and EOCC reject 17.3% of the session more than RLC. However, in RLC and HOC there are no restrictions. Thus, no session is rejected until the load passes through the network capacity, after which RLC has fewer rejected sessions. Because the rejection of the sessions is based on an intelligent process in accordance with the existing conditions, the negotiation of the agents makes the rejection of the sessions more logical, rather than selfish behavior in HOC. Therefore, RLC reject 9.4% of session less than HOC.

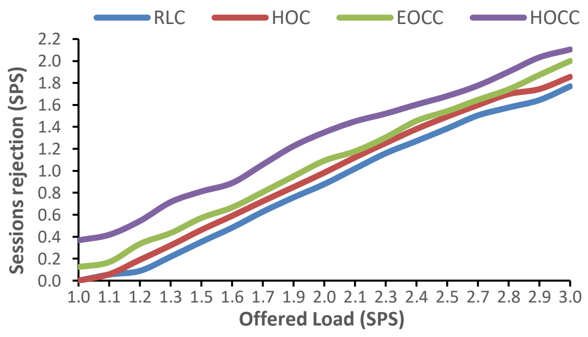

Fig. 7: The number of rejected sessions for the studied mechanisms.

To test the stability and rapid response of the mechanisms, the number of input sessions is initially considered equal to 200 SPS below the network capacity, which lasts up to 100 secs. During this period, Goodput corresponds to the offered load. At 100 secs, the number of offered load suddenly increases to three times the network capacity (900 SPS). With this technique, it can be seen how quickly the mechanisms under study react in the face of sudden load changes and maintain their stability or not. The offered load is returned to 200 SPS at 200 secs. The Goodput and sessions delay of tests are shown in Figs. 8 and 9. In Fig. 8, all mechanisms respond quickly to the offered load sharp changes. Moreover, HOC achieves better Goodput because the offered load is predicted. In HOC, the holons react quickly to a sudden increase in the offered load and control it from the sources. However, its Goodput is more than RLC due to consecutive switching between the holons. RLC responds steadily to sudden load changes due to the negotiation. This is because the agents act quickly by increasing the offered load and entering the warning area. Moreover, they prevent overload from occurring through negotiation. In HOCC, the Goodput is initially decreased and then increased. Because the control parameters are updated based on 200 SPS and until the next update, sessions are accepted. EOCC methods require time to propagate overload information from destination to the sources, causing temporary instability. At 200 secs, Goodput of RLC returns to its previous value without any fluctuation. Therefore, RLC completely satisfies stability.

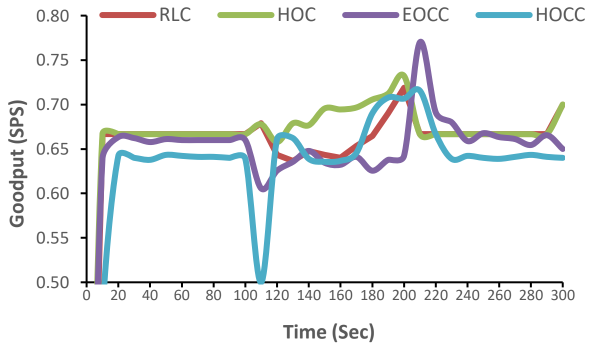

Fig. 8: Goodput of the studied mechanisms when the offered load changes suddenly.

In Fig. 9, RLC has less delay than other methods. When the load returns to its previous value, the RLC delay will return to the value before the change. As the offered load increases due to the lack of up to date parameters in OCC methods, a large amount of load enters the network, and the delay increases. By removing the overload, the delay is slightly reduced. Parameter values in EOCC are updated with more delay due to being end to end and passing of control values over the entire network. The HOC makes a temporary error because it predicts load based on previous observations and the current load is very different from the previous.

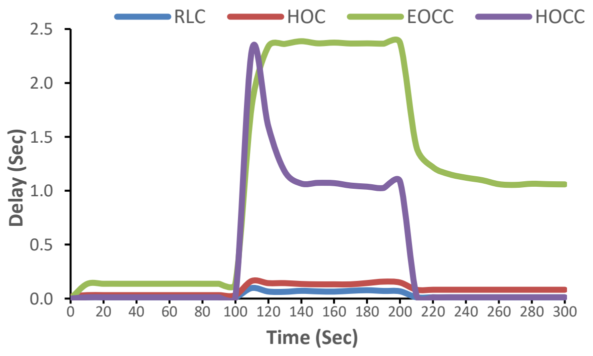

Fig. 9: Sessions delay of the studied mechanisms when the offered load changes suddenly.

To evaluate the performance of RLC in a real VOIP environment, the Goodput and sessions delay for variable offered load with Poisson distribution are shown in Figs. 10 and 11. According to Fig. 10, RLC follows the changes in the number of incoming sessions well and adapts to it without network failure. When the offered load and Goodput are not distinguishable, they have been overlapped because Goodput is equal to offered load. In addition, the delay is controlled and fluctuated with the load changes and the system becomes stable.

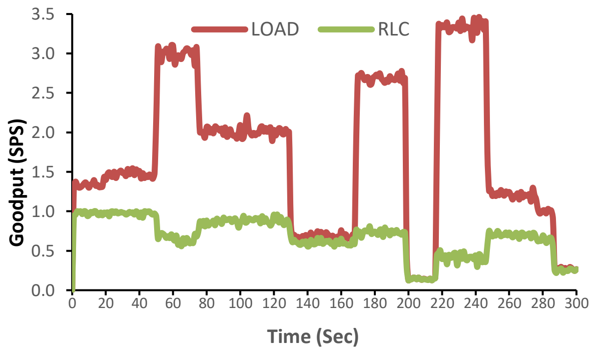

Fig. 10: Goodput of RLC under real offered load.

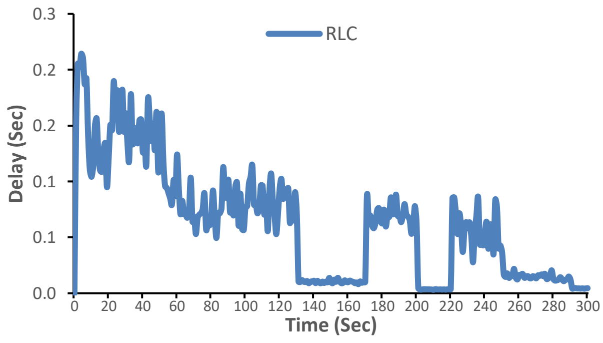

Fig. 11: The sessions delay of RLC under real offered load.

Conclusion

IMS will become the most important platform for multimedia applications. By increasing the number of users, the throughput of the IMS servers is decreased. On the other hand, the issue of overload control in IMS is a complex system, for which multi-agent systems are good alternatives to classical solving methods. In multi-agent systems, a large task can be divided into a set of smaller tasks so that each agent performs a task partially. In this study, IMS servers are considered as a learner and negotiator agents. Agents learn the values of thresholds by Q-learning and they implement a hop by hop control method through negotiates strategy with the upstream agent. To prove the performance of the proposed method, it was compared to similar methods. In Table 8, we reported the efficiency, the mean sessions delay, the average of Goodput and number of rejected sessions for different methods. As shown, RLC has better performance on all three measured parameters; while only, its delay is more than HOC. Because holonic communication of HOC is faster than negotiation process of RLC. In the proposed method, the learning process is done independently by each agent. Although this type of learning is suitable for obtaining the parameters related to each agent, to show the optimal reactions by all agents, it is better to perform learning in the whole network to implement end to end methods through clustering. On the other hand, in IMS, there are HSS and DNS that are not based on SIP and contribute to the overload. These heterogeneous creatures can be inserted into the problem. Nowadays, by moving the process to cloud environments with NFV, the cost of IMS structure and platform has decreased. By quickly developing this method in scalable form, the proposed method can be implemented in cloud environments using NFV.

Author Contributions

M. Khazaei wrote the manuscript, designed the experiments, analysis the data, interpreted the results and revised the manuscript.

Acknowledgment

The author would like to thank the editor and reviewers for their helpful comments.

Conflict of Interest

The authors declare no potential conflict of interest regarding the publication of this work. In addition, the ethical issues including plagiarism, informed consent, misconduct, data fabrication and, or falsification, double publication and, or submission, and redundancy have been completely witnessed by the authors.

Abbreviations

IMS	IP Multimedia Subsystems
NGN	Next-Generation Network
SIP	Session Initiation Protocol
MAS	Multi Agent Systems
QOS	Quality of Services
HSS	Home Subscriber Server
SLF	Subscription Locator Function
CSCSF	Call Session Control Functions
S-CSCF	Serving CSCF
P-CSCF	Proxy CSCF
I-CSCF	Interrogating CSCF
NFV	Network Functions Virtualization
NLMS	Normalized Least Mean Square
SPS	Sessions per Second
C	CPU Capacity
RLC	Reinforcement Learning Overload Controller
OC	Overload Controller
EOCC	End to end Occupancy
HOCC	Hop by hop Occupancy

References

[1] P. Agrawal, Y. Jui-Hung, C. Jyh-Cheng, Z. Tao, "IP multimedia subsystems in 3GPP and 3GPP2: overview and scalability issues," IEEE Commun. Mag., 46: 138-145, 2008.

[2] K. K. Guduru, U. Jayadevappa, "Overload control in SIP signalling networks with redirect servers," Int. J. Wireless Mobile Comput., 19: 124-132, 2020.

[3] V. S. Vaishnavi, Y. M. Roopa, P. L. Srinivasa Murthy, "A survey on next generation networks," in Proc. ICCNCT 2019: 162-171, 2020.

[4] C. Shen, H. Schulzrinne, E. Nahum, "Session Initiation Protocol (SIP) server overload control: Design and evaluation," in Principles, Systems and Applications of IP Telecommunications. Services and Security for Next Generation Networks, S. Henning, S. Radu, and N. Saverio, Eds., ed: Springer-Verlag: 149-173, 2008.

[5] V. Hilt, I. Widjaja, "Controlling overload in networks of SIP servers," in Proc. IEEE International Conference in Network Protocols: 83-93, 2008.

[6] J. Davin, P. Riley, M. Veloso, "CommLang: Communication for coachable agents," in Proc. RoboCup 2004: Robot Soccer World Cup VIII. vol. 3276, D. Nardi, M. Riedmiller, C. Sammut, and J. Santos-Victor, Eds., ed: Springer Berlin Heidelberg: 46-59, 2005.

[7] M. Wooldridge, An Introduction to MultiAgent Systems, Newyork: Wiley, 2009.

[8] B. Jang, M. Kim, G. Harerimana, J. W. Kim, "Q-Learning algorithms: A comprehensive classification and applications," IEEE Access, 7: 133653-133667, 2019.

[9] M. Abdoos, N. Mozayani, A. C. Bazzan, "Hierarchical control of traffic signals using Q-learning with tile coding," Appl. Intell., 40: 201-213, 2014.

[10] H. C. Hsieh, J. L. Chen, "Distributed multi-agent scheme support for service continuity in IMS-4G-Cloud networks," Comput. Electr. Eng., 42: 49-59, 2015.

[11] D. Pereira, R. Oliveira, H. S. Kim, "A machine learning approach for prediction of signaling sip dialogs," IEEE Access, 9: 44094-44106, 2021.

[12] F. B. Mismar, J. Hoydis, "Unsupervised learning in next-generation networks: Real-time performance self-diagnosis," IEEE Commun. Lett., 25: 3330-3334, 2021.

[13] S. Chen, Z. Yao, X. Jiang, J. Yang, L. Hanzo, "Multi-agent deep reinforcement learning-based cooperative edge caching for ultra-dense next-generation networks," IEEE Trans. Commun., 69: 2441-2456, 2021.

[14] M. Abdoos, N. Mozayani, A. L. C. Bazzan, "Holonic multi-agent system for traffic signals control," Eng. Appl. Artif. Intell., 26: 1575-1587, 2013.

[15] M. Abdoos, N. Mozayani, A. L. C. Bazzan, "Traffic light control in non-stationary environments based on multi agent Q-learning," in Proc. 14th International IEEE Conference in Intelligent Transportation Systems (ITSC): 1580-1585, 2011.

[16] Y. Yao, V. Hilaire, A. Koukam, W. Cai, "A holonic model in wireless sensor networks," in Proc. International Conference in Intelligent Information Hiding and Multimedia Signal Processing: 491-495, 2008.

[17] S. M. Hosseini, N. Mozayeni, "An intelligent method for resource management in wireless networks," in Proc. 5th Conference in Information and Knowledge Technology (IKT): 371-376, 2013.

[18] E. Pei, L. Zhou, B. Deng, X. Lu, Y. Li, Z. Zhang, "A Q-Learning based energy threshold optimization algorithm in LAA networks," IEEE Trans. Veh. Technol., 70: 7037-7049, 2021.

[19] M. Cossentino, N. Gaud, V. Hilaire, S. Galland, A. Koukam, "ASPECS: an agent-oriented software process for engineering complex systems," Auton. Agents Multi-Agent Syst., 20: 260-304, 2010.

[20] M. Poikselkä, The IMS: IP multimedia concepts and services: Newyoek, J. Wiley & Sons, 2006.

[21] N. M. Ahmed, N. E. Rikli, "QoS-Based data aggregation and resource allocation algorithm for machine type communication devices in next-generation networks," IEEE Access, 9: 119735-119754, 2021.

[22] V. K. Gurbani, R. Jain, "Transport protocol considerations for session initiation protocol networks," Bell Labs Tech. J., 9: 83-97, 2004.

[23] M. Ohta, "Overload control in a SIP signaling network," Int. J. Electr. Electron. Eng., 3: 87-92, 2009.

[24] E. N. V. Hilt, C. Shen, A. Abdelal, "Design considerations for session initiation protocol (SIP) overload control," Internet Engineering Task Force (IETF), Request for Comments: RFC6357, 2011.

[25] J. Liao, J. Wang, T. Li, J. Wang, J. Wang, X. Zhu, "A distributed end-to-end overload control mechanism for networks of SIP servers," Comput. Networks, 56: 2847-2868, 2012.

[26] H. Dong-Yeop, P. Ji Hong, Y. Seung-wha, K. Ki-Hyung, "A window-based overload control considering the number of confirmation massages for SIP server," in Proc. Fourth International Conference in Ubiquitous and Future Networks (ICUFN): 180-185, 2012.

[27] M. Homayouni, H. Nemati, V. Azhari, A. Akbari, "Controlling Overload in SIP Proxies: An Adaptive Window Based Approach Using No Explicit Feedback," in Proc. IEEE Global Telecommunications Conference: 1-5, 2010.

[28] S. V. Azhari, M. Homayouni, H. Nemati, J. Enayatizadeh, A. Akbari, "Overload control in SIP networks using no explicit feedback: A window based approach," Comput. Commun., 35: 1472-1483, 2012.

[29] A. Montazerolghaem, M. H. Yaghmaee Moghadam, "Improving efficiency of SIP protocol using window-based overload conditions," Soft Comput. J., 2: 16-25, 2021.

[30] M. Khazaei, N. Mozayani, "A dynamic distributed overload control mechanism in SIP networks with holonic multi-agent systems," Telecommun. Syst., 63: 437-455, 2016.

[31] Y. Hong, C. Huang, J. Yan, "Applying control theoretic approach to mitigate SIP overload," Telecommun. Systems, 54: 387-404, 2013.

[32] A. Abdelal, W. Matragi, "Signal-Based Overload Control for SIP Servers," in Proc. 7th IEEE Consumer Communications and Networking Conference: 1-7, 2010.

[33] R. G. Garroppo, S. Giordano, S. Niccolini, S. Spagna, "A Prediction-Based Overload Control Algorithm for SIP Servers," Network and Service Management, IEEE Transactions, 8: 39-51, 2011.

[34] S. Jing, T. Ruixiong, H. Jinfeng, Y. Bo, "Rate-based SIP flow management for SLA satisfaction," in IFIP/IEEE International Symposium on Integrated Network Management: 125-128, 2009.

[35] M. Khazaei, "Occupancy overload control by Q-learning," in Fundamental Research in Electrical Engineering, Singapore: 765-776, 2019.

[36] M. Khazaei, N. Mozayani, "Overload management with regard to fairness in session initiation protocol networks by holonic multiagent systems," Int. J. Network Manage., 27: e1969, 2017.

[37] A. Akbar, S. M. Basha, S. A. Sattar, "A cooperative overload control method for SIP servers," International Conference in Proc. Communications and Signal Processing (ICCSP): 1296-1300, 2015.

[38] A. Montazerolghaem, S. K. Shekofteh, M. H. Yaghmaee, M. Naghibzadeh, "A load scheduler for SIP proxy servers: design, implementation and evaluation of a history weighted window approach," Int. J. Commun. Sys., 2015.

[39] A. Montazerolghaem, M. H. Y. Moghaddam, A. Leon-Garcia, "OpenSIP: Toward software-defined SIP networking," IEEE Trans. Netw. Serv. Manage., 15: 184-199, 2018.

[40] R. Gandotra, L. Perigo, "SDVoIP—A software-defined VoIP framework for SIP and dynamic QoS," Comput. J., 64: 254-263, 2019.

[41] L. D. Cicco, G. Cofano, S. Mascolo, "Local SIP overload control: controller design and optimization by extremum seeking," IEEE Trans. Control Network Syst., 2: 267-277, 2015.

[42] Y. Hong, C. Huang, J. Yan, "Modelling chaotic behaviour of SIP retransmission mechanism," Int. J. Parallel Emerg. Distrib. Syst., 28: 95-122, 2013.

[43] J. Wang, J. Liao, T. Li, J. Wang, J. Wang, Q. Qi, "Probe-based end-to-end overload control for networks of SIP servers," J. Network Comput. Appl., 41: 114-125, 2014.

Biographies

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).

- Email: m.khazaei@kut.ac.ir
- ORCID: 0000-0002-4780-065X
- Web of Science Researcher ID: NA
- Scopus Author ID: 1027784
- Homepage: https://kut.ac.ir/en/profile/6-mehdi-khazaei

How to cite this paper:
M. Khazaei, “Reinforcement learning-based load controller in IP multimedia subsystems,” J. Electr. Comput. Eng. Innovations, 11(1): 21-32, 2023.

DOI: 10.22061/JECEI.2022.8723.546

URL: https://jecei.sru.ac.ir/article_1713.html