Random Backoff for Active Control of Information Updates in Smart Spaces

Olga Bogoiavlenskaia, Dmitry Korzun, Kirill Kulakov
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
{dkorzun, olbgvl, kulakov}@cs.karelia.ru

***Abstract*—Performance fluctuation on the semantic information broker side due to critical growth of the request queue length can cause its complete failure or substantial degradation of its services.Since semantic broker is one of the key element of the smart space architecture its performance is a vital issue for the whole environment. We consider two possible sources of the performance degradation.One of them is the growth of the request arrival rate beyond the limits of its capacity. Another source finds its origin in the request synchronization which happens due to the presence of the regular patterns in the activities of the smart space participants. Rather often implementation of the relevant approaches at SIB size poses substantial difficulties. We propose several algorithms that can be implemented on the KPs side. The algorithms form an additional mechanism that can reduce SIB fault rate and avoid performance degradation by regulating the KPs activity and hence balancing the load of SIB. The mechanism presumes additional timeout to the persistent request. The timeout value is selected according to the current system state and its history in the nearest past. The timeout consists of two components. The first one is determined be the active control with the adaptive strategy and the second one is defined by the random backoff algorithm.**

I. Introduction

Smart spaces are considered as service-oriented information sharing environments [1], [2]. Such an environment can be utilized for deploying Internet of Things (IoT) applications, which need dynamic location detection and context-available interaction of physical objects [3], [4]. We consider smart spaces deployed in localized resource-restricted IoT environments. They provide services to a group of mobile clients. Such an environment is typically associated with a physical spatial-restricted place equipped with a variety of devices. In the smart space everyday life objects, alongside traditional computers, become data processors and service constructors to their users, which use mobile devices for interactions.

Smart space deployment in a given IoT environment requires a software platform to address interoperability across heterogeneous devices and to support constructing multiple services. Our reference study is the M3 architecture with its implementation in the Smart-M3 platform [1], [5]. The latter provides open source middleware for implementation of the smart space concept in IoT environments. Semantic information broker (SIB) is the key element of the M3 architecture [6], [7], which manages a knowledge base shared with all the smart space participants. Such a participant can produce data for this information sharing, consume the shared information, reason new knowledge over the collected information, and share the new information in the knowledge base.

The knowledge base uses an RDF triplestore. The Smart Spaces Access Protocol (SSAP) [1], [8] implements interaction between a SIB and its smart space participants (called knowledge processors—KPs). SSAP provides read/write operations for inserting, removing, updating, querying, and publish/subscribe messaging. The operations can be extended with SPARQL queries. In an IoT environment, the SIB can be hosted by heterogeneous devices as ordinary computers or embedded devices with low-capacity as single-board computers or wireless routers [9].

The performance is subject to fluctuation on the SIB side due to the critical growth of its requests pool. The fluctuation can cause complete failure or substantial degradation of the SIB operation with KPs. In this paper, we consider the SIB performance problem and possible performance improvement mechanisms. Our development of the proposed performance improvement mechanisms uses technique of load balancing and congestion control.

The following two reasons of the SIB performance degradation are analyzed. The first reason is too high request arrival rate compared with the available capacity of SIB host device. The reason is crucial for the case of fog-based and edge-centric IoT environments [10]. The second reason is burst-like clustering of concurrent requests to the SIB from many KPs during some short time period. In smart space applications this clustering frequently occurs when many KPs subscribe on updates of the same shared information [11], e.g., reacting on the same events or advanced search queries from different KPs provide the same result.

There are mechanisms that a SIB can use for performance management [6]. When such a mechanism fully implemented on the SIB side, then some SIB capacity is consumed, and the performance is decreased. We expect that load balancing between the SIB and its KPs is a reasonable solution. Our proposed algorithms form an additional mechanism, which is implemented on the KP side, and the processing workload is delegated (partially) to the appropriate smart space participants themselves. The mechanism introduces timeout between subsequent requests of the same KP. The timeout value is selected according to the observable system state and the collected short-term history of faults and losses.

The timeout consists of the two components. The first one is determined by the active control with adaptive strategy, which we studied previously in [11], [12]. The timeout value tends to decreasing when much loss is observed. The second timeout component is determined by a random backoff algo-

ISSN 2305-7254

rithm [13]. The introduction of the backoff follows the idea of randomization to reduce clustering of concurrent requests from many KPs. Timeout values differ for different KPs, leading to more uniform distribution of the SIB workload in time. Note that the adaptive strategy in its pure from can lead to very short timeouts for many KPs, so making the SIB workload extremely high. The high load results in more losses, and the backoff algorithm prevents this decrease of the timeout.

The rest of the work is organized as follows. Section II describes SIB management mechanisms for request pools. Section III introduces our method of active control by many smart spaces participants using the adaptive strategy with random backoff. Section IV analyzes the required properties of the proposed active control. Section V discusses the step-wise character of the proposed active control, which aims at delegating some processing from the SIB to its KPs. Section VI summarizes our current results.

II. Requests Pool Problem

Let us consider the overall SIB performance and its relation to indiviual KPs behavior. Initially this performance problem was introduced in [13] in respect to mobile clients for services created in the smart space.

The SIB operation is RDF triple-based [1]. Each KPs access the information using the following operations, which can be resource-consuming for the SIB.

- *Request–Response* operation. KP sends a triple-based mask or even a complicated SPARQL request. The SIB responses with several found triplets. Such a KP request can be computationally time-consuming and the result can be large (many triples).
- *Subscription–Notification* operation. KP sends subscription request. The SIB sends a notification if the specified triples are updated, deleted or new triples are published. Subscription requests are sent from many KPs, and matching the information updates with the subscribed KP is a computationally difficult task.
- *Update* operation. KP sends a request to update triples. The SIB transfers this requests to the RDF triplestore. The operation itself is not resource-consuming. Nevertheless, an *Update* operation request frequently leads to *Subscription–Notification* operation.

The SIB performance can degrade due to the burst-like synchronization of requests coming from a large set of autonomous KPs. Each sends own requests sequence to SIB concurrently and possibly with high rate. The simultaneous occurrence of requests forms a burst (i.e., many requests during the same short time interval). Typically a requests burst is caused by detection of the same events (information update) that are detected by the KPs. The high individual rate is determined by too short delays between subsequent requests from the same KP. In this case, SIB can apply the following techniques for the processing of the incoming request flow [14].

1) The SIB implements one or more requests queues in the order of receiving the requests [15] and processes each queue using the FIFO discipline. Each queue is assigned for certain type of requests, e.g., for advanced search queries when a query requires much resources for resolving.
2) To optimize processing of repeating requests, the SIB combines several queries into a single query. The operation with information store is performed once while the result is delivered to several requesters [16]. This technique needs caching incoming requests at the SIB side.
3) Under the high rate of incoming requests, the SIB can drop some requests or deny some queries. For instance, a resource-demanding search query can be postponed or resolved with an incomplete result. Similarly, a query for out-of-date information can be rejected [17].

TABLE I. Basic techniques for workload control

Host	Method	Description
SIB	FIFO processing	Requests pool operates as several parallel FIFO queues.
	Queries caching	Caching the same queries without duplicative access to the RDF triplestore
	Dropping requests	Requests queue has the length in dependency on the SIB host device capacity. To preserve the queue size within the bounds some incoming requests are dropped
KPs	Active control	Persistence query includes requests activated individually by the KP, which selects timeout between subsequent requests. The timeout value calculation is based on the current system situation the KP observes.

Using the FIFO queue leads to increasing the SIB response time, caching capacity is limited. The cache size management and the requests denial policy reduce the service quality level for individual KPs. On the application level, some KPs and their requests can be prioritized. Processing is performed at the SIB side, so consuming its limited resources. Table I is inherited from [13] (see also [18]). The table summarizes basic techniques to solve the performance problem.

We expect that delegation of some part of processing from the SIB to its KP can essentially reduce the SIB workload. That is, KPs become more responsible for own information sharing activity control. The similar approach to the congestion control is widely applied in distributed systems [19], [20].Individual sources of the workload follow congestion control mechanisms to balance the load of the infrastructure elements. Proper balancing prevent and/or avoid the congestion. In particular, this approach showed its efficiency in the TCP protocol and some LAN MAC protocols. In our case, the individual KPs perform congestion control algorithms by using information update timeout.

Figure 1 shows the concept model of the SIB requests pool. The formal symbols are summarized in Table II. The SIB serves m concurrent KPs. Each KP can apply individual control of its activity determined by rate λ_k for $k = 1, 2, \ldots, m$. For simplicity we assume that KPs are identical and they requests similar persistent queries. Each KP k consequently sends requests R_i to the SIB, $i = 1, 2, \ldots$. The activity control is implemented by timeout value t_i elapsed between sending the subsequent requests R_{i-1} and R_i. The generic mathematical representation for the timeout calculation by an individual KP is as follows.

$$t_i = f\left(t_{i-1}, k_i, T^{\text{br}}, T^{\text{br}}_{i-1}, h\right). \quad (1)$$

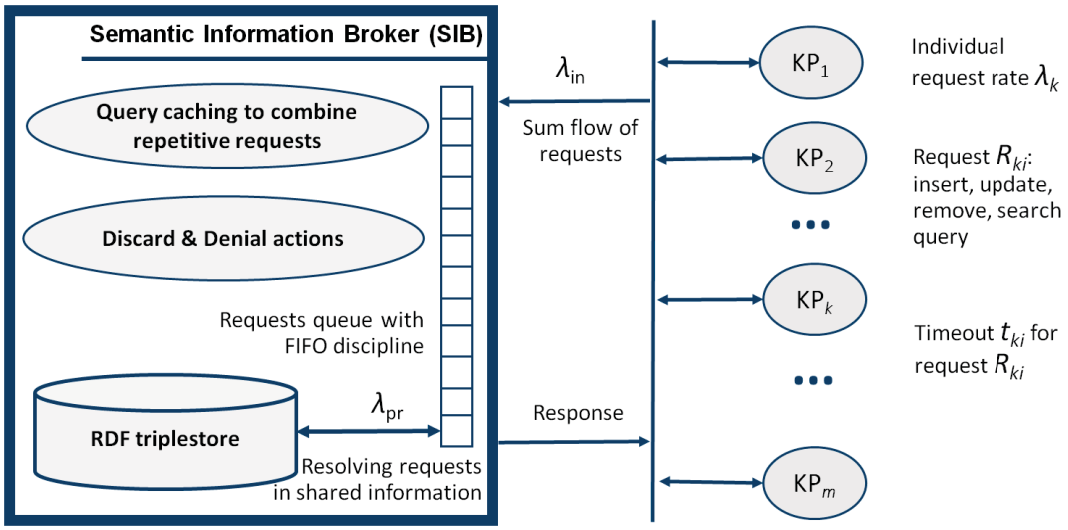

Fig. 1. SIB requests pool model

TABLE II. SYMBOL NOTATION

Symbol	Description
t_i	Active control timeout, i.e., the time elapsed between requests R_{i-1} and R_i
k_i	Number of update losses during t_i
α, δ	Parameters of the adaptive strategy
h	Parameters of the random backoff algorithm
t_i^{ast}, t_i^{bck}	Adaptive Strategy Timeout value (AST) and Random Backoff Timeout value (RBT)
T^{br}	The metric for burst-like requests synchronization detection
$\bar{T}^{\text{br}}$	Smoothed average of the SIB response time
R_{br}^n	The nth instance of the SIB response time visible by the individual KP
β	RBT multiplier

SIB accepts requests by forming the queues. Processing is with the sum rate λ_{pr}. This rate can be lower than the incoming requests rate λ_{in}. The difference is due to the discard/deny policy of the SIB. Each KP individually can influence λ_{in} by selecting own timeout value t_i for the next request R_i.

Based on the requests pool model, we study the following directions for improving the performance. The key idea is that each KP can take some workload while the KP activity is subject to adaptation and desynchronization with the activity from the other KPs.

1) The problem of SIB load balancing when each KP may individually make active control actions (active control for information updates). For the problem we formulate the timeout model for the active control and present the algorithm for applying the timeout model in the smart spaces. The timeout consists of two components: Adaptive Strategy Timeout (AST) and Random Backoff Timeout (RBT).
2) For AST we design AIMD-like evaluation scheme basing on its properties presented in our previous work we discuss its influence on SIB workload and its role in the high-quality information updates.
3) For RBT we formulate the model of its evaluation provide method for tuning its parameters in under numerous different conditions.

III. ACTIVE TIMEOUT CONTROL FOR INFORMATION UPDATES

Our approach is two-fold. First we use adaptive strategy from our previous work, see, e.g. [12]. In the strategy the choice of the timeout value depends on the current stage of the system. The timeout increases if the information updates are successfully delivered to the KP and decreases if updates losses were identified. Detection of losses means that the KP should increase the active request rate to avoid the losses of the important information.

Meanwhile in the case of multiple losses the strategy substantially decreases the timeout value which in turn increases the SIB workload and may cause its failure and/or performance degradation. At the same time KPs often fulfill their requests in parallel with similar parameters for determining the value of the timeout. In this case the requests may arrive in batches in SIB queue or in the tight groups similar to the batches. The synchronization appear due to the nature of some high-level applications. For instance if smart-room provides a presenter's slides to the listener's devices the slide updates happen synchronously and create a batch of requests (burst-like behavior). The batches may essentially increase SIB response time and hence decrease its performance.

We denote the inappropriate growth of the SIB response time as a collision despite of its nature. To avoid the collision the KPs use the random backoff algorithm. The idea of the algorithm is successive growth of the timeout starting from randomly selected value if a collision of requests was identified. Significant increase of SIB response time for a request is considered as an identifier of the collision.

The total value of timeout for the request to send is equal to the sum of the adaptive strategy timeout (AST) t_i^{act} for i's request and the random backoff timeout (RBT) t_i^{bck},

$$t_i = t_i^{\text{ast}} + t_i^{\text{bck}}. \quad (2)$$

Each timeout is a complicated function, where value for i is calculated based on the current system state and the previous history, see (1).

Algorithm 1 Active Timeout Control

Require: initialization of active control on client side
1: **for** $i = 0$ **to** End of the session **do**
2: {start thread for R_i}
3: **loop**
4: doQuery(R_i);
5: t_i^{ast} = activeControl(t_{i-1}, k_{i-1}, α);
6: t_i^{bck} = backoffTimeout(T_{br}, β, h);
7: sleep($t_i^{\text{act}} + t_i^{\text{bck}}$);
8: **end loop**
9: **end for**

Algorithm 2 Adaptive Strategy

function activeControlTimeout(R_i):
1: k_j = getQueryLosses(R_i); {receive losses number}
2: calculateEstimates(); {recalculate estimates}
3: **if** $k_j = 0$ **then**
4: $t_{ij}^{\text{act}} = t_{i(j-1)}^{\text{act}} + N * \delta$;
5: **if** $T - t_{ij}^{\text{act}} \leq 0$ **then**
6: {use previous timeout value to avoid losses}
7: $t_{ij}^{\text{act}} = t_{i,j-1}^{\text{act}}$;
8: **end if**
9: **else**
10: $t_{ij}^{\text{act}} = t_{i,j-1}^{\text{act}}/\alpha$;
11: **end if**
12: **return** t_{ij}^{act};

Algorithm 3 Random Backoff

Require: $j = 0$ {j is own for each thread}
function backoffTimeout(R_i):
1: j++;
2: {if backoff was reset on previous round}
3: **if** t_j^{bck} = 0 **then**
4: t_j^{bck} = $t_{\min}$;
5: **end if**
6: {determine performance level}
7: duration = getLastQueryDuration(R_i);
8: **if** duration $\leq$ getAverageQueryDuration(R_i) **then**
9: t_j^{bck} = 0;
10: **return** t_j^{bck};
11: **end if**
12: {calculate backoff timeout}
13: $t_j^{\text{bck}} = \min(t_{j-1}^{\text{bck}}\beta, t_{\max})$;
14: t_j^{bck} = t_j^{bck} + variation(t_j^{bck} * seed);
15: **return** t_j^{bck};

Algorithm 1 is an adapted version from [13]. The algorithm shows the use of the timeouts AST and RBT in (2) to reduce the SIB workload by spreading requests across the timeline. Algorithm 2 and Algorithm 3 are used for calculating the AST and the RBT, correspondingly. The explanation of the algorithms computational logic is as follows.

Adaptive Strategy Timeout: Following our previous work [11], we consider the adaptive strategy of active control. It implements "adaptation to losses" when a KP reduces its active request timeout if updates losses are observed and increases the timeout, otherwise. In fact, the adaptive strategy could be considered as a generalization of TCP congestion avoidance algorithm of additive–increase/multiplicative–decrease (AIMD).

Generalized AIMD-like adaptive strategy has the following form. Let $i = 1, 2 \ldots$ be a sequence of the checks done by the client, t_i^{ast} be the time period between consecutive checks $i-1$ and i, and k_i be the number of losses during t_i^{ast}. At the end of t_{i-1}^{ast} the client makes the decision about the next t_i^{ast} period using $t_i^{\text{ast}} = g(t_{i-1}^{\text{ast}}, k_{i-1})$. In the simplest case, we straightforwardly apply the AIMD algorithm as follows.

$$t_i^{\text{ast}} = \begin{cases} t_{j-1}/\alpha, & k_{j-1} > 0 \\ t_{i-1}^{\text{ast}} + \delta & k_{i-1} = 0, \end{cases} \quad (3)$$

where $\alpha > 1$ stands for decrease and $\delta > 0$ for increase values of check timeout length. More complete variant of this equation is described in our work [11].

In our previous work [12], we obtain analytical estimates for parameters in (3) that can be used to tune the strategy. In particular, T is the expected length of check timeout before a multiplicative decrease, N is the number of consecutive growths, and K is a metric for different loss types.

Any KP can implement the adaptive strategy. Let m identical KPs send requests to the SIB. Each KP computes its own value of the AST t_i^{act} using (3).

Random Backoff Timeout: Randomized backoff algorithms are widely used in distributed systems to coordinate access to a shared resource (e.g., to desynchronize the concurrent access). Different versions of the algorithm are implemented in the broad range of applications, e.g., Ethernet LAN, wireless networks, memory transactions, e-mail transmission, multiprocess locks, TCP timeout control and others. One could classify these methods onto two groups. Those are detected collision control and collision avoidance.

The first one assumes that collisions can be detected. If so the participants chose random delay and repeat an attempt to grab a resource. If collision repeats they increase the value using a multiplier or any other monotonously growing function. Most well-known implementation of the randomized exponential backoff is [21] where after each resolution failure the delay doubles. Collision avoidance in turn assumes that participants use random delay even before their first attempt to grab a resource. If collision happens anyway they chose new value of the delay from the wider range of values. The approach is used for the environments where collisions are hardly to detect, e.g. in the IEEE 802.11x standards family.

Since randomized backoff is widely used for the several decades it has many variants and it is studied extensively. One can found vast bibliography in [22]. This work presents important properties that randomized backoff is expected to provide. These are stable high throughput, few failed access attempts and efficient robustness. All these features are expected by the individual KPs from SIB performance.

As it was mentioned above the adaptive strategy has limitations when a large set of individual KPs operate in parallel. Supplementing the adaptive strategy by the randomized backoff is to adjust the drawback. The random backoff algorithm could spread the batch of the requests across the timeline and decrease total arrival rate visible by the SIB.

IV. Requirements Analysis

The proposed incorporation of the backoff algorithm into the smart spaces environment assumes using of the implicit collision detection which immediately invokes random backoff. The simplest way of the detection bases on a measurement of the average response time on the KPs side. Thus the KP calculates duration of the request processing. In the most of cases and if the processing takes substantially more time then queuing delay is significant and the KPs should use the backoff algorithm to improve the performance by reducing simultaneous request number. If first attempt doesn't bring success, then the KP follows random backoff algorithm until the response time returns to the value desired.

The random exponential backoff algorithm with collision detection could be described by following equation:

$$t_i^{\text{bck}} = \min(t_{i-1}^{\text{bck}}\beta, t_{\max}), \tag{4}$$

where $\beta \geq 2$ and $t_{\max} \leq 5 *$ average query duration, that stands for stopping increase backoff value after 5 rounds. This algorithm is currently in use in all 802.11 standards.

For additional randomization of the backoff delay selection one can use variation described as follows.

$$t_j^{\text{bck}} = t_j^{\text{bck}} + \mathsf{Var}(t_j^{\text{bck}} * \text{seed}), \tag{5}$$

where variation is the random function that returns a value, which follows normal distribution and its seed is for random initialization.

To indicate a congestion episode at SIB side the individual KP estimates SIB response time T^{br}. Several levels of the requirements to the value could be considered as follows:

1) Rigorous requirements. The client demands queuing delay to be as small as possible. Therefore

$$T^{\text{br}} = T^{\text{proc}} + T^{\text{net}} + \beta * T^{\text{proc}} \tag{6}$$

Here T^{proc} is the query processing time itself, T^{net} is network Round Trip Time (RTT) and $\beta * T_{proc}$ is queuing delay demand where $\beta < 1$.

2) Mediate requirements. The KP relays on SIB estimation of the queuing delay. Here SIB defines the limit of the queue size that allows stable processing. The SIB informs all KPs about the value

$$D^{\text{br}} = T^{\text{proc}} + T^{\text{queu}}.$$

The individual KP either assumes or if possible evaluates the network delay T_{net} and uses the value

$$T^{\text{br}} = D^{\text{br}} + T^{\text{net}}$$

as a collision indication.

3) Light requirements. Here the client applies technique which is widely used in many data communication environments. It maintains geometric moving average of the response time it observes and indicates congestion when next response time or response waiting period substantially exceeds the value of the average. The average value is calculated as

$$\tilde{T}_n^{\text{br}} = \gamma\tilde{T}_{n-1}^{\text{br}} + (1-\gamma)T_n^{\text{br}} \tag{7}$$

Here T_n^{br} is next value of the response time observed by the KP.

Rigorous requirement demand high performance from SIB or if SIB capacity is pure the KPs population should essentially reduce the SIB load to stay within the requirement. The mediate requirement need further communication between SIB and the clients since SIB implicitly informs clients about its capacity and the congestion level. The light requirement propose the simplest evaluation method but it is most error prone.

The minimum, average and maximum values of RBT are essential as well, since the initial value of the RBT plays key role in the congestion control for the considered environment. If the value is too small then the algorithm would not resolve a collision for fairly small number of attempts. If the value is too big then the overall performance of the system will degrade dramatically. We consider three possible approaches to the problem.

1) The client keeps short history of the check intervals and corresponding response times, pick from the history check interval before the last response time instance which was within the limits according to the chosen level of requirements t_{hist}^{ack}, computes $b = t_{\text{past}}^{\text{ack}} - t_{\text{cur}}^{\text{ack}}$ and uses $b > 0$ as an expectation of the distribution function which generates first t_i^{bck}. So in this approach the KP relies on the value from the past when congestion did not happen yet.
2) The client keeps history of the backoff values. When congestion is successfully eliminated it stores the values of the timeout which brought success. During next congestion event it uses this value as an expectation of the distribution which generates first t_i^{bck}. This method relies on the history of success in the congestion control.
3) The SIB informs the KP about desirable arrival rate. The KP starts the backoff from this value. This method proposes using the best information on the congestion level available but it will create overhead due to the additional signaling between SIB and the clients.

V. Discussion

Let us summarize and discuss the intuition behind the proposed algorithms in respect to the studied SIB performance problem. The basic idea is the following step-wise enhancing of the active control from the KPs side.

1) No active control. All information updates are processed by SIB, which sends appropriate notifications to all interested KPs. The case is subject to much loss when the SIB workload is high.
2) Basic active control. Any KP can occasionally request the SIB to check for updates. In a deterministic strategy, timeouts t_i are fixed. In a randomized strategy, t_i are random values according with some probability distribution, and leading to average timeout t_{avg}. The KP improves the SIB performance by taking a certain fixed share of processing, while the share is independent on the current system state.

3) Adaptive strategy. Any KP increases or decreases the timeout t_i in dependence on decreasing or decreasing the loss of information update notifications. When the SIB is under high workload then the SIB can reduce its control of information updates, and KPs become responsible for receiving notification on updates. Nevertheless, when many KPs make their timeouts small then the SIB receives more requests, increasing the SIB workload.
4) Active timeout control: Adaptive strategy + Random backoff. RBT makes the timeout larger when ATS is too small. Also, the random timeout component makes smoother a batch of incoming requests to the SIB (in the case of some information update interested to many KPs). As a result, the KP is prevented from generating high workload to the SIB.

The expected performance improvement provided by RBT implementation could be described as follows. Let us denote τ as average AST provided by the adaptive strategy. The value could be derived using our previous work [11]. Therefore a single client generates the flows of update request with the average rate $1/\tau$ and m clients generate total flow with the intensity m/τ. Applying the central limit theorem for flows we assume the the total flow of the requests is Poisson flow. Let N_t define the limit of SIB capacity, i.e., the maximum number of the requests arrived during the interval t which SIB can process without unacceptable delay and/or failure. Then the probability that the N_t limit is achieved in the system is

$$\Pr\{N_t\} = \frac{[\frac{m}{\tau}t]^{N_t}}{N_t!}e^{-[\frac{m}{\tau}t]}.$$

Hence, the RBT implementation supports avoiding completely or reducing significantly the probability of achieving or exceeding N_t.

In our further research we plan to experimentally evaluate the improvement degree that each step above provides for the system performance.

VI. Conclusion

The paper described a novel active control mechanism based on enhancing the known adaptive strategy with a random backoff. The mechanism reduces the problem of high request pool from many mobile clients to their SIB in a smart space. We use the active control for persistent queries (e.g., subscription on information updates). The adaptive strategy decreases timeouts when high loss is experienced, while the backoff algorithm avoids many simultaneous requests from many clients and frequent requests from an individual client. Improvements are achieved by reducing the number of requests to SIB, as well as in the event of a decrease in the intensity of broker processing, clients begin to be distributed on a time line to allow SIB to stabilize its work. The direction of our future work is evaluation of the proposed solution with the use of an experimental system consisted of services and clients.

Acknowledgment

The research was financially supported by the Ministry of Education and Science of Russia within project # 2.5124.2017/8.9 of the basic part of state research assignment for 2017–2019. The reported study was funded from Russian Fund for Basic Research according to research project # 19-07-01027. The results were implemented by the Government Program of Flagship University Development for Petrozavodsk State University in 2017–2021.

References

[1] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, "Smart-M3 information sharing platform," in *Proc. IEEE Symp. Computers and Communications (ISCC'10)*. IEEE Computer Society, Jun. 2010, pp. 1041–1046.

[2] S. Balandin and H. Waris, "Key properties in the development of smart spaces," in *Proc. 5th Int'l Conf. Universal Access in Human-Computer Interaction (UAHCI '09). Part II: Intelligent and Ubiquitous Interaction Environments, LNCS 5615*, C. Stephanidis, Ed. Springer-Verlag, Jul. 2009, pp. 3–12.

[3] D. Korzun, S. Balandin, and A. Gurtov, "Deployment of Smart Spaces in Internet of Things: Overview of the design challenges," in *Internet of Things, Smart Spaces, and Next Generation Networking*, ser. Lecture Notes in Computer Science, S. Balandin, S. Andreev, and Y. Koucheryavy, Eds., vol. 8121. Springer, Aug. 2013, pp. 48–59.

[4] L. Roffia, F. Morandi, J. Kiljander, A. D. Elia, F. Vergari, F. Viola, L. Bononi, and T. Cinotti, "A semantic publish-subscribe architecture for the Internet of Things," *IEEE Internet of Things Journal*, vol. PP, no. 99, 2016.

[5] D. G. Korzun, S. I. Balandin, A. M. Kashevnik, A. V. Smirnov, and A. V. Gurtov, "Smart spaces-based application development: M3 architecture, design principles, use cases, and evaluation," *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, vol. 8, no. 2, pp. 66–100, 2017.

[6] I. Galov, A. Lomov, and D. Korzun, "Design of semantic information broker for localized computing environments in the Internet of Things," in *Proc. 17th Conf. of Open Innovations Association FRUCT*. IEEE, Apr. 2015, pp. 36–43.

[7] F. Viola, A. D'Elia, D. Korzun, I. Galov, A. Kashevnik, and S. Balandin, "The M3 architecture for smart spaces: Overview of semantic information broker implementations," in *Proc. of the 19th Conference of Open Innovations Association FRUCT*, S. Balandin and T. Tyutina, Eds. IEEE, Nov. 2016, pp. 264–272.

[8] J. Kiljander, F. Morandi, and J.-P. Soininen, "Knowledge sharing protocol for smart spaces," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 3, pp. 100–110, 2012.

[9] S. Marchenkov, D. Korzun, A. Shabaev, and A. Voronin, "On applicability of wireless routers to deployment of smart spaces in Internet of Things environments," in *Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2. IEEE, Sep 2017, pp. 1000–1005.

[10] D. Korzun, A. Varfolomeyev, A. Shabaev, and V. Kuznetsov, "On dependability of smart applications within edge-centric and fog computing paradigms," in *2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT)*, May 2018, pp. 502–507.

[11] D. Korzun, M. Pagano, and A. Vdovenko, "Control strategies of subscription notification delivery in smart spaces," in *Distributed computer and communication networks*, ser. Communications in Computer and Information Science (CCIS), V. Vishnevsky and D. Kozyrev, Eds. Springer International Publishing, 2016, vol. 601, pp. 40–51.

[12] A. S. Vdovenko, O. I. Bogoiavlenskaia, and D. G. Korzun, "Study of active subscription control parameters in large-scale smart spaces," pp. 344–350, Nov 2017.

[13] A. Vdovenko, "Active control with backoff algorithm for reducing broker load in smart spaces," in *Proc. 22nd Conf. Open Innovations Association FRUCT*, May 2018, pp. 397–400.

[14] L. Ferdouse, A. Anpalagan, and S. Misra, "Congestion and overload control techniques in massive M2M systems: a survey," *Transactions on Emerging Telecommunications Technologies*, vol. 28, no. 2, 2017.

[15] L. De Cicco, G. Cofano, and S. Mascolo, "Local SIP overload control: Controller design and optimization by extremum seeking," *IEEE*

Transactions on Control of Network Systems, vol. 2, no. 3, pp. 267–277, 2015.

[16] M. Knuth, O. Hartig, and H. Sack, "Scheduling refresh queries for keeping results from a sparql endpoint up-to-date," in *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*. Springer, 2016, pp. 780–791.

[17] M. Ohta, "Overload control in a SIP signaling network," in *Proceeding of World Academy of Science, engineering and technology*, 2006, pp. 205–210.

[18] D. Kuptsov, B. Nechaev, A. Lukyanenko, and A. Gurtov, "How penalty leads to improvement: A measurement study of wireless backoff in ieee 802.11 networks," *Computer Networks*, vol. 75, pp. 37–57, 2014.

[19] A. S. Tanenbaum and M. Van Steen, *Distributed systems: principles and paradigms*. Prentice-Hall, 2007.

[20] A. Lukyanenko, A. Gurtov, and E. Morozov, "An adaptive backoff protocol with markovian contention window control," *Communications in Statistics-Simulation and Computation*, vol. 41, no. 7, pp. 1093–1106, 2012.

[21] R. M. Metcalfe and D. R. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Communications of the ACM*, vol. 19, no. 7, pp. 395–404, Jul. 1976.

[22] M. A. Bender, J. T. Fineman, S. Gilbert, and M. Young, "Scaling exponential backoff: Constant throughput, polylogarithmic channel-access attempts, and robustness," *Journal of ACM*, vol. 66, no. 1, pp. 6:1–6:33, Dec. 2018.