Active Control with Backoff Algorithm for Reducing Broker Load in Smart Spaces

Andrey S. Vdovenko
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
vdovenko@cs.karelia.ru

***Abstract*—Performance degradation on broker side and the subsequent possible failure is an important issue in smart spaces. One of the reasons for this problem is the increase in the number of requests to the broker beyond what it can handle. The broker can have own management tools to solve such problematic situations. More often, it is more difficult to implement relevant approaches to broker itself. Our suggestion is to implement additional mechanism on agents sides to reduce fault events. The mechanism is to add a timeout to the persistent request, the value of which is selected based on the current system state. For timeout determining we suggest active control with adaptive strategy and improve it by adding backoff algorithm.**

I. Introduction

Smart spaces can be a perspective environments for deploying IoT applications due to the possible organization of dynamic detection and interaction between physical objects with use of the available context in their locations [1], [2]. We consider smart spaces deployed in localized resource-restricted IoT environments consumed by a group of mobile clients [3]. Such a smart space is typically associated with a physical spatial-restricted place equipped with a variety of devices, where everyday life objects, alongside traditional computers, become data processors and service constructors to their users, which use mobile devices for interactions.

A smart space requires a software platform, addressing interoperability across heterogeneous devices and providing multiple services. This paper considers the Smart-M3 platform, which provides open source middleware for implementation of the smart space concept in IoT environments [4]. The central component of the M3 architecture is the semantic information broker (SIB). Each SIB manages and shares a knowledge base with all the smart space participants. The knowledge base is semantic, in the form of the RDF triplestore, and operations with it through SSAP interaction protocol. In IoT, the broker can be hosted by heterogeneous devices as ordinary computers or embedded devices with low-capacity as single-board computers or wireless routers [5].

Large number of simultaneous requests can roughly reduce broker performance or even call its fault. The broker has own management tools to resolve such situation, but in some cases it is better to add additional mechanisms on the agent's side. To resolve this issue from mobile client, they can perform a persistent query with timeouts. But there stands problem of what is an optimal way to calculate such timeouts. We suggest the use of active control with adaptive strategy [6], which could be improved by backoff algorithm. Using of active control allows to reduce broker load for different requests and backoff algorithm is intended to avoid simultaneous of the same requests from different clients.

The rest of the work is organized as follows. Section II describes Smart-M3 platform management tools for request pools. Section III shows the implementation of the suggested approach with use of active control for many clients with backoff algorithm. Section IV summarizes our current results.

II. Requests pool problem in Smart Spaces

When working in smart spaces, there are problems associated with the overall performance of the broker and individual agents, most often mobile clients. Problems with the performance of the broker arise because of collisions of requests from many agents, that send them simultaneously or with small delays. To resolve such situations, the broker has operations to manage the processing of the incoming request flow [7]. Table I describes management tools that can be applied on the broker and client sides to solve overload situations. The broker can build a queue of requests in the order of their receiving [8]. As an optimization of this approach in the case of mobile clients, which rapidly send same requests, the broker could combine several queries into sets over the data area, and doesn't perform unnecessary operations with the database and sends the result immediately [9].

The broker's performance with high query intensity is strongly affected by the amount of information retrieved and the execution of additional operations (inserting and deleting triplets, SPARQL queries and subscription maintenance) because of which the queue begins to grow endlessly. In such cases, the broker can ignore the arrival of new requests, or do not perform queries that are more complicated in time and do not send out-of-date notifications [10]. But in this case, this strategy can adversely affect the work of the application system, so it is possible to set priorities performed requests for active agents, for example, when the highest priority is given to requests of service agents, and customer requests are performed in the best effort.

Problem of fault tolerance is vital issue of interests in related fields of distributed systems and client server communication networks [11], [12]. Increased loads leads to faults for them and a large bunch of problems in that cases cause by using Wireless LANs, because of higher losses percent due to collisions of radio signals and low-capacity routers. That is the same for the smart spaces issue. They use different approaches to achieve fault tolerance quality similar with the

TABLE I. MANAGEMENT TOOLS FOR OVERLOAD CONTROL

Host	Tool	Specification
SIB	FIFO processing	Requests pool processed in order of receiving
	Queries caching	Caching same queries in memory without interaction with database
	Request reject	Requests queue have length based on broker host capacity. Algorithm rejects requests after they number are bigger than maximum queue length
Clients	Active control strategy	Long-term requests performed with timeout. Timeout value calculation is based on current system situation

Smart-M3 broker and one of them is backoff mechanism that implemented by clients [13].

Management on the side of the broker does not always solve the problems related to mobile clients, for example in the field of delivery of notifications. Therefore, in this case, the client can perform its own management, for example, by addition to the subscription operation of timeout checks, or completely replacing the subscription operation for independent active control of data updates. Fig. 1 demonstrates request pool to the broker from m mobile clients in case of active control. For simplicity, we assume that clients are the same and they perform same persistent requests. Each client sends n requests, denoted by R_i. Each R_i request repeats after timeout value t_{ij}, where $j = 1, .., m$ is the index of repeat, this relation can be denoted by $t_{ij} = f(R_i, j)$. The broker receives requests and maintains queues of them and starts processing with intensity λ_{pr}. Processing intensity mean that if the client request pool will grow with intensity greater than λ_{pr}, broker would stop answering on requests even with the use of management tools. Rational choice of values for t_{ij} timeouts would allow to avoid such situation.

III. ACTIVE CONTROL IMPROVED BY BACKOFF ALGORITHM

The choice of the timeout value depends on the current situation in the system, i.e., the timeout is shorter when the data is updated more frequently and the load on the system small and bigger otherwise. To determine the value of the load on the broker, the query execution time can be used, since The average time for executing the query during a no load

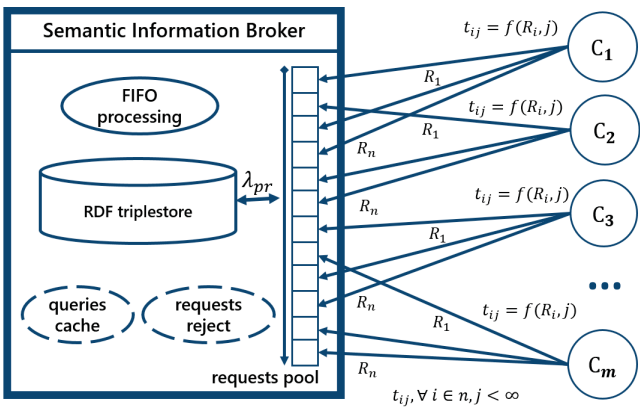

Fig. 1. Requests pool to the broker from m clients with same n requests of different types.

Algorithm 1 Active control with backoff algorithm for n persistent requests.

Require: initialization of active control on client side

```
for i = 0 to n do
    {start thread for R_i performing}
    loop
        doQuery(R_i);
        t^act = activeControl(R_i);
        t^bck = backoffTimeout(R_i);
        sleep(t^act + t^bck);
    end loop
end for
```

is known. Clients often fulfill their requests in parallel with similar parameters for determining the value of the timeout. In this case, it becomes necessary to determine the additional delay, aimed at reducing the number of simultaneously sent requests. The best mechanism for this purpose is the backoff algorithm. Its meaning is a successive increase in the timeout from which the size of the delay is randomly selected in the event of a collision of requests, for example, a significant increase in the execution time of the request. Due to that, total value of timeout for the request is equal to sum of active control timeout for i request and backoff algorithm timeout for client on current j repeat round, i.e., $t_{ij} = t_{ij}^{\text{act}} + t_j^{\text{bck}}$. Algorithm 1 gives pseudo code for use of timeouts with n persistent requests to reduce broker load by spreading requests on the timeline. Functions for calculation of active control and backoff timeouts are described in Algorithm 2 and Algorithm 3 correspondingly.

a) Active control timeout: Following our previous work [14], we consider the adaptive strategy of active control. It implements "adaptation to losses" when the client reduces its check timeout if updates losses are observed and increases the check timeout, otherwise. In fact, the adaptive strategy is a generalization of the TCP algorithm of additive–increase/multiplicative–decrease (AIMD).

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

$$t_j = \begin{cases} t_{j-1}/\alpha, & k_{j-1} > 0 \\ t_{j-1} + \delta & k_{j-1} = 0, \end{cases} \quad (1)$$

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 previous work [14].

In our previous work [6], we obtain analytical estimates for parameters that can be used to tune strategy such as T the expected length of check timeout before a multiplicative decrease, N the number of consecutive growths and K metrics for different types of a loss flow.

In this work we generalize adaptive strategy for m clients with n requests on each. Each client for each request have

Algorithm 2 Active control algorithm for n persistent requests.

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};

own calculations of timeout values denoted before by t_{ij}^{act}, where i is the index of request in the range of one client. Algorithm 2 show implementation of adaptive strategy correspondingly with (1) for n persistent requests with use of analytical estimates. For each request starts a separate thread to parallel execution of queries. After each query processing, we calculate estimates to reflect on the current system situation. The number of consecutive growths N is used for a fast increase of t_{ij} to its high bound. Check current t_{ij}^{act} values and if they are bigger than the expected length T, we use previous timeout value.

b) Backoff timeout: Adaptive strategy in active control has limitations in the case of many parallel clients. That is because of the same parameters for requests across all clients, which mean that requests will be performed in the same time and this will lead to performance degradation. To spread requests on timeline suggested use backoff algorithm for each request on detecting performance degradation.

The main idea in the implementation of the backoff algorithm for smart spaces is event detection when we need to use it. The most simplest way is a measure of average round trip time (RTT), i.e., we know the duration of the operation in most of cases and if the operation takes more time we need to use the backoff algorithm to improve this by reducing simultaneous request number. If first use doesn't give success, then we need to repeat backoff algorithm until RTT gets close to average.

Standard backoff algorithm can be described by following equation:

$$t_j^{\text{bck}} = min(t_{j-1}^{\text{bck}} * factor, maxTimeout), \qquad (2)$$

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

To add additional randomize for backoff value selection can be used variation described with following:

$$t_j^{\text{bck}} = t_j^{\text{bck}} + variation(t_j^{\text{bck}} * seed), \qquad (3)$$

where variation is random function that returning value, which follows normal distribution and seed is for random initialization.

Algorithm 3 Backoff algorithm for clients.

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} = minTimeout;
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^{bck} = min(t_{j-1}^{bck} * factor, maxTimeout);
14: t_j^{bck} = t_j^{bck} + variation(t_j^{bck} * seed);
15: **return** t_j^{bck};

Algorithm 3 shows how the backoff algorithm can be implemented for active control to decrease simultaneous requests with use of equations (2) and (3). Function store value of previously calculated timeout and reset it to zero on performance stabilization. The constant minTimeout is the start value (e.g., 100 ms). The constant maxTimeout is upper bound to timeout (e.g., 1 minute). The constant factor is coefficient of timeout grow (e.g., 2). The constant seed is used for random fluctuations and can be equal to system time. Duration of last query is less or equal to average value we stop backoff algorithm by setting its timeout to zero.

IV. Conclusion

We describe our approach to resolve problem of high request pool of mobile clients to broker. Our suggestion is to use active control for persistent queries with improving it by backoff algorithm to avoid simultaneous requests. Improvements are achieved by reducing the number of requests to the broker, 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 the broker 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

This research is 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 work is implemented within 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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.