Enhanced SDP-Dynamic Bloom Filters for a DDS Node Discovery in Real-time Distributed Systems

Williams Paul Nwadiugwu, Cha Joong-Hyuck, Dong-Seong Kim, Joong-Hyuck Cha

Enhanced SDP-Dynamic Bloom Filters for a DDS Node Discovery in Real-time Distributed Systems

Williams Paul Nwadiugwu, Joong-Hyuck Cha and Dong-Seong Kim, *Senior Member, IEEE*
School of Electronic Engineering, Department of IT Convergence Engineering
Kumoh National Institute of Technology, Gumi, South Korea.
{williams.nwa, ckwndgur, dskim}@kumoh.ac.kr

***Abstract*—In this paper, an enhanced SDP-Dynamic bloom filters for a DDS node discovery scheme in real-time distributed systems is proposed. Since the previous works of the DDS focuses more on the usage of a Simple Discovery Protocol (SDP) for endpoint to endpoint information communication of industrial-scale networks, attempts have now been made to enhance this approach into the Simple Discovery Protocol Dynamic Bloom Filters (SDP-Dynamic Bloom) focusing more on scalability in the amount of sent and stored message packets in the industrial network system. Simulation result show that the proposed scheme viciously reduce the overall computing and processing time of both stable and unstable industrial network environment which arises during the restructuring process of the existing SDP bloom filters approach.**

***Index Terms*—Data distribution service, Dynamic bloom filters, Large-scaled real-time systems, Middleware.**

I. Introduction

The importance of real-time systems in industrial applications can no-longer be overlooked. Recently, there have been quite a number of intensive studies in the real-time systems and application [1]. Data Distribution Service (DDS) [2] is a network middleware for real-time data distribution, defined by the Object Management Group to standardize the real-time publish-subscribe communication model. DDS have been used in real-world systems integration, such as combat management systems, tracking and management applications, air-traffic control systems, and in industrial controls [3].

Normally, a discovery protocol is required for communication in a DDS, allowing the publisher and subscriber to discover each other and eventually obtain information such as domain Identity(ID) and many more. Hence, a basic DDS implementation must provide at least the Simple Discovery Protocol (SDP) to enable interoperability.

Several methods have been proposed to improve SDP scalability. Previous study [4] has proposed the idea of an adaptive SDP for dynamic large-scale network systems and described the Quality of Service (QoS) supported by DDS. Their works considered the challenges of applying DDS in large-scale distributed 35 applications. A method to improve SDP is also presented by Kyoungho et al. [5], which proposed Content Filtering Discovery Protocol (CFDP). Their method aims to enhance SDP for large-scale systems by providing a content filtering mechanism based on topic names and endpoint types matching.

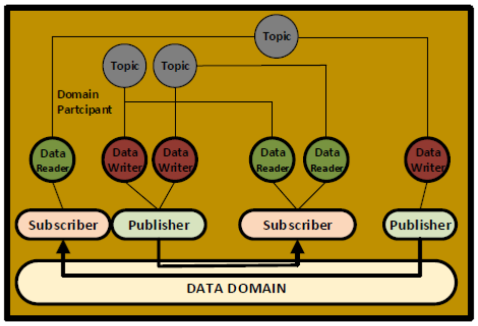

Fig. 1. Diagram showing the Data Distribution Service (DDS) entity representation.

To the best of our knowledge, a modified SDP-Dynamic bloom filters for an industrial-scale DDS node discovery scheme with major focuses on network latency has not been reported in this literature. Thus, our main contribution in this paper includes:

- Improving SDP by having each DDS participant send summary information with Dynamic Bloom Filters.
- Improving SDP scalability in unstable network environments by eliminating 55 unnecessary calculations to restructure bloom filters when endpoint addition or deletion operations occur.

Hence, an enhanced SDP-Dynamic Bloom filters is made simpler with more realistic prediction and implementation.

II. Problem Formulation

A. Data Distribution Service

DDS provides the communications service needed to distribute time-critical data between embedded devices or nodes. DDS uses the publish-subscribe communications model to make efficient and robust data distribution.

Fig. 1 above shows that in order for data published by the data writer to be received by the subscribing data reader, both the data reader, and the data writer must have the same topic.

B. Simple Discovery Protocol

Node discovery in DDS is implemented by using built-in data writers and data readers [6]. Javier et al. [7] proposed a combination of SDP with bloom filters for DDS discovery process (SDP-Bloom). However, their work did not consider

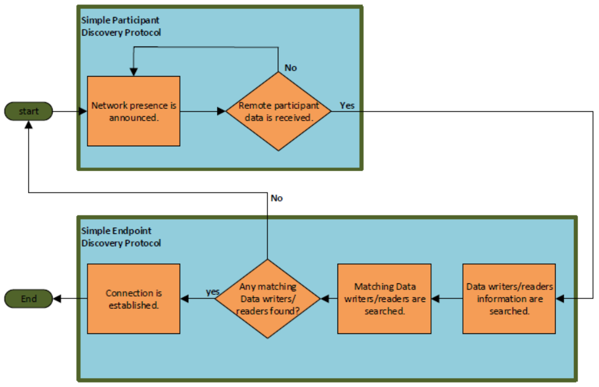
Simple Participant Discovery Protocol
start
Network presence is announced.
Remote participant data is received.
No
Yes
No
Simple Endpoint Discovery Protocol
End
Connection is established.
yes
Any matching Data writers/ readers found?
Matching Data writers/readers are searched.
Data writers/readers information are searched.

Fig. 2. Diagram showing the connection flow of a Simple Discovery Protocol from start to finish.

the endpoint deletion 45 scenario. It is not applicable for real DDS applications where the number of endpoints frequently changes. Fig. 2 shows an SDP flow diagram. Each domain participant has automatically created built-in data readers and data writers for discovery purposes.

C. Bloom Filters

In as much as bloom filters are widely used, they often have to be rebuilt from scratch each time an element removal operation is done in order to prevent a false negative error. When large-scale distributed systems are considered, these limitations becomes critical, since reconstruction of bloom filters becomes inefficient and this could in-turn cause potential disruption of the system. The main idea is to use k hash functions, $h_i(x), 1 \leq i \leq k$, to map item $x \in S$ to numbers in the range $1, ..., m$. An element $x \in S$ is inserted into the filters by changing the bit values of $h_i(x)$ to one. If the bit value of $h_i(y)$ is one, y is assumed a member of S. It becomes a member if any bit value of $h_i(y)$ is zero.

In the first initialization, all bit values in the filters are set to zero. Fig. 3 presents an overview of a 16 bit Bloom filter. Three elements $x, y,$ and z are inserted into the Bloom filter. Each of the elements is hashed using a $k = 3$ hash function to bit positions in the filters. Corresponding bits are set to one according to the hash function result. The membership test operation is performed to determine element existence in the filters. The false positive error probability of bloom filter $f^{BF}_{m,k,n}$ is calculated as follows [8],

$$f^{BF}_{m,k,n} \approx (1-(1-\frac{1}{m})^{kn}))^k \approx (1-e^{-\frac{kn}{m}})^k. \qquad (1)$$

III. SDP-Dynamic Bloom Filter

One distinction between SDP-Dynamic bloom and SDP-Bloom is when an endpoint deletion event occurs in SDP-Dynamic bloom, the filters immediately delete the endpoint information, while in SDP-Bloom the filters need to be reconstruct from scratch. SDP-Dynamic bloom can save significant computing time in the scenario where the endpoint to participant ratio is large, resulting in lower network latency time.

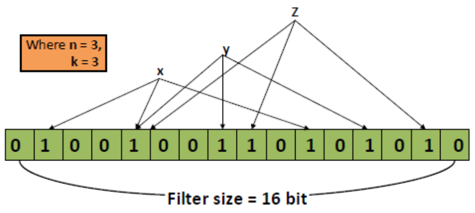

Fig. 3. An overview of a 16 bit Basic Bloom Filter Scheme with hash function k set at three.

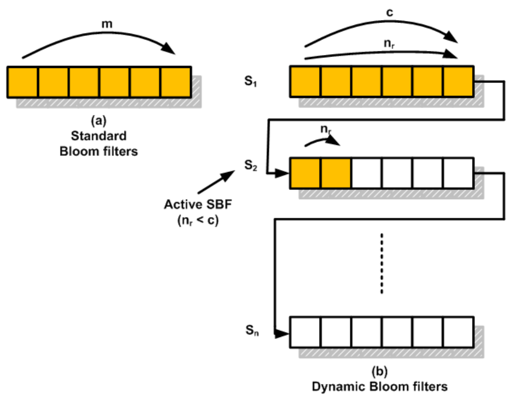

Fig. 4. Diagram showing the Data Structure of a Dynamic Bloom Filter size adjustment consisting of s standard Bloom Filter of m size.

Fig. 4 provides dynamic filter size adjustment [9]. DBF consists of s standard bloom filter of m size. DBF allows dynamic filter growth by adjusting the number of bloom filters. Only one bloom filter in the DBF can be active at a time, it is called the active bloom filter. The number of elements inserted into each bloom filter in a DBF, denoted by n_r, is tracked. Inspired by works on [10], the filter is implemented using a 4-bit counter instead of a single bit to prevents information loss from element deletion if one or more keys are deleted. A false positive error could happen on a DBF because there is a possibility of that all counters $h(x)$ in any standard bloom filter have been set to a non-zero value by elements of S [9]. False positive error probability of DBF can be calculated with the same method as standard bloom filters according to Equation (1), if the cardinality of S is not greater than the capacity threshold c.

The probability of a false positive error on a DBF when there is no element deletion operation performed is as follows,

$$f^{DBF}_{m,k,c,n} =$$

$$1\text{- } (1\text{-}(1\text{-}e^{-kc/m})^k)^{\lfloor n/c \rfloor} \times (1-(1-e^{-k(n-c\lfloor n/c \rfloor)/m})^k). (2)$$

After element deletion, let x represent one of the first $s-1$ during the element insertion process, the false positive error is calculated as follows,

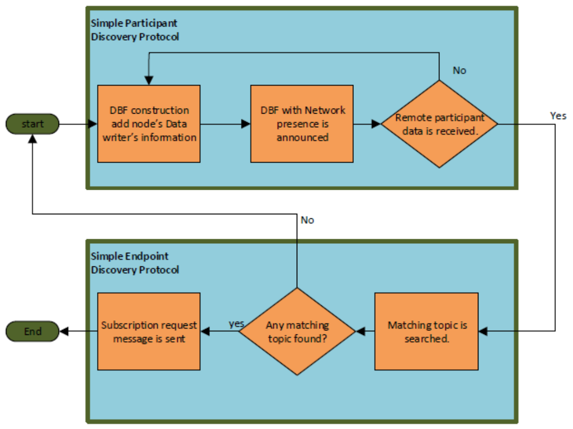
Simple Participant Discovery Protocol
start
DBF construction add node's Data writer's information
DBF with Network presence is announced
Remote participant data is received.
No
Yes
No
Simple Endpoint Discovery Protocol
End
Subscription request message is sent
yes
Any matching topic found?
Matching topic is searched.

Fig. 5. Flow Diagram of a Simple Discovery Protocol for a Dynamic Bloom Filter from the start of participant protocol to the endpoint protocol.

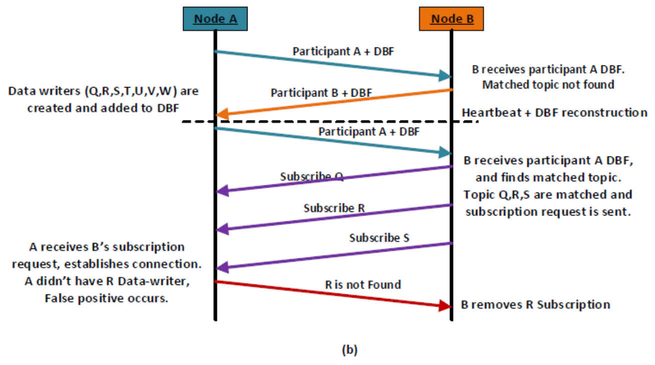
Node A
Node B
Participant A + DBF
B receives participant A DBF. Matched topic not found
Data writers (Q,R,S,T,U,V,W) are created and added to DBF
Participant B + DBF
Heartbeat + DBF reconstruction
Participant A + DBF
B receives participant A DBF, and finds matched topic. Topic Q,R,S are matched and subscription request is sent.
Subscribe Q
Subscribe R
Subscribe S
A receives B's subscription request, establishes connection. A didn't have R Data-writer, False positive occurs.
R is not Found
B removes R Subscription
(b)

Fig. 6. SDP for a Dynamic Bloom Timing Flow in which the remote endpoints are discovered, matched and then later published or subscribes information.

$$f^{DBF}_{m,k,c,n} =$$

$$1\text{-}(1\text{-}(1\text{-}e^{-kc/m})^k)^{s-2}(1-(1-e^{-k(n-c\lfloor n/c\rfloor)/m})^k).(3)$$

On the other hand, if element x is represented by the sth standard bloom filters during the element insertion process, which means that there is at least one of $s-1$ standard bloom filters, this will result in an incorrect membership report for x. The false positive error probability of this event is calculated using Equation (4).

$$f^{DBF}_{m,k,c,n} = 1-(1-(1-e^{-kc/m})^k)^{s-1}. \qquad (4)$$

Fig. 5 describes the proposed SDP-Dynamic bloom scheme flow diagram. In the first initialization of DDS each participant builds the DBF. The membership test operation is performed using Algorithm I. DBF iterates through its DBF until the bloom filters that contains the matched remote endpoints are discovered. If a matching key is found, then the next step is to check whether the received data is trying to publish or subscribe information. If the result is negative, then a false positive occurs. The connection is established when the receiver is successfully communicate with chosen participant. Fig. 6 visualizes the proposed SDP-Dynamic bloom scheme. If an endpoint wants to leave the network, it sends a delete/exit message to the participants in the same domain. Participants delete the endpoint from the DBF using Algorithm II and multi-cast the data to all other participants.

Algorithm I: MembershipTest (x)

```
Require: $x \neq$null
1:  for $j$ = 1 to $s$ do
2:      $counter \leftarrow 0$
3:      for $j = 1$ to $k$ do
4:          if $BF_i[hash_j(x)]$ = 0 then
5:              break
6:          else
7:              $counter \leftarrow counter$+ 1
8:          end if
9:      end for
10:     if $counter = k$ then
11:         return true
12:     end if
13:     return false
14: end for
15: if $ActiveBF$ = null then
16:     $ActiveBF \leftarrow CreateBF(m, k)$
17:     $DBF \leftarrow DBF \cup ActiveBF$
18:     $s \leftarrow s + 1$
19: end if
20: for $i$ = 1 to $k$ do
21:     $ActiveBF[hash_i(x)] \leftarrow ActiveBF[hash_i(x)] + 1$
22: end for
23: $ActiveBF.n_r \leftarrow ActiveBF.n_r + 1$
```

Removing an element requires finding the bloom filters claiming that the element is present. If only one element is found, the k counters are decremented by one. If more than one element is found, the element bits are not decremented to prevent false negative error.

Algorithm II: Deletion of (x)

Delete (x)

```
Require: $x \neq$ null
1:  $index \leftarrow$ null
2:  $counter \leftarrow 0$
3:  for $i$ = 1 to $s$ do
4:      if $DBF[i].MembershipTest(x)$ then
5:          $index \leftarrow i$
6:          $counter \leftarrow counter + 1$
7:      end if
8:      if $counter > 1$ then
9:          break
10:     end if
11: end for
12: if $counter = 1$ then
13:     for $i$ = 1 to $k$ do
14:         $DBF[index][hash_i(x)] \leftarrow DBF[index]\ [hash_i(x)]$
            - 1
```

TABLE I
SIMULATION PARAMETERS

Parameter	Value or Mode
Transport Mode	UDPv4
Endpoint/Participant Ratio	20 to 100
Hash Functions	MD5
Number of Hash (k)	5, 10, 15
False positives threshold ($f^{BF}_{m,k,n}$, $f^{DBF}_{m,k,c,n}$)	0.005%
Bloom filters Size (n_{BF})	500 bytes
Bloom filters Size (n_{DBF})	50 bytes
Bloom filters Vector Size (m_{DBF})	10 bytes
DBF capacity threshold (c_{DBF})	10

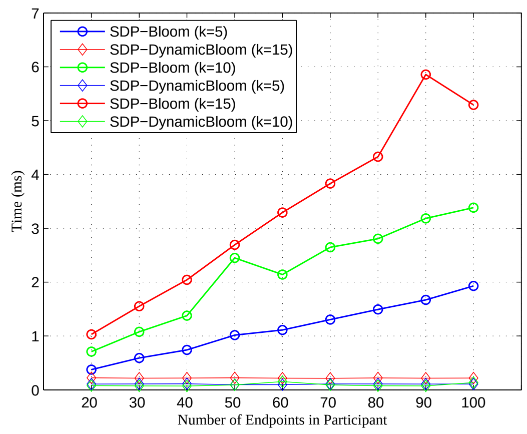

Fig. 7. Endpoint Deletion Delay Comparison Between SDP-Bloom and SDP-DynamicBloom.

```
15:         $DBF[index].n_r \leftarrow DBF[index].n_r - 1$
16:         for $k = j + 1$ to $s$ do
17:             if $BF_j.n_r + BF_k.n_r < c$ then
18:                 $BF_j \leftarrow BF_j \cup BF_k$
19:                 $BF_j.n_r \leftarrow BF_j.n_r + 1$
20:                 Clear $BF_k$ from the $DBF$
21:                 break
22:             end if
23:         end for
24:         return true
25:     end for
26: else
27:     return false
28: end if
```

IV. PERFORMANCE EVALUATION

The simulation result performs an endpoint deletion delay comparison between SDP-Bloom and SDP-Dynamic bloom experiments using OpenDDS software [11]. The parameters for the experiments are listed in Table 1 where performance of SDP-Dynamic bloom was compared with SDP and SDP-Bloom in unstable DDS environments. Instability of DDS environments is modeled as a combination of participant join and leave rates when the network suffers a particular churn rate. The simulator creates a DDS network of approximately n participants, it then produces a particular churn rate: 0.0/0.0 (stable situation without endpoints joins/leaves) and 1.0/1.0 (one endpoints join/leave per second). Average latency is calculated once the simulation is over. Fig. 7 shows that SDP-Dynamic bloom performs better than SDP-Bloom in terms of hash function iteration effect on filter construction.

V. CONCLUSION AND FUTURE WORKS

The proposed scheme was successfully combined with the existing SDP scheme to produce an efficient node discovery scheme. The critical difference between SDP-Bloom and SDP-Dynamic bloom is that the latter does not need to rebuild the DBF from scratch. The improvements were considerable in scenarios with large numbers of endpoints per participants. SDP-Dynamic bloom did not suffer from increased network delay and latency caused by filter reconstruction with an increased endpoints churn rate, while SDP-Bloom did.

Finally, potential practical integration issues shall be the hall-mark of focus in subsequent future researches.

VI. ACKNOWLEDGMENT

This research was financially supported by National Research Foundation of Korea (NRF) through the Human Resource Training Project for Regional Innovation 2015 (NO. NRF-2015H1C1A1035971) and the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the the ITRC(Information Technology Research Center) support program(IITP-2017-H8601-16-1011) supervised by the IITP(Institute for Information and communications Technology Promotion).

REFERENCES

[1] D.-S. Kim, J. Jeon, P. Mohapatra, "Scheduling of wireless control networks based on IEEE 802.15.4 networks: Mixed traffic environment," *Control Engineering Practice 19 (10) (2011) 1223-1230.*

[2] Object Management Group, "OMG the real-time publish-subscribe wire protocol dds interoperability wire protocol specifications," *version 1.2. (2009).*

[3] M. Bakir, D. Nugroho, Y. Wei, D.-S. Kim, "A Real-time data distribution scheme of MOMAT for the naval combat system," *ICT Convergence (ICTC), 2013 International Conference 2013, pp. 268 - 273.*

[4] N. Wang, D. Schmidt, H. van't Hag, A. Corsaro, "Toward an Adaptive Data Distribution Service for Dynamic Large-Scale Network-Centric Operation and Warfare (NCOW) Systems," *Military Communications Conference (MILCOM) IEEE 2008, pp. 1 - 7.*

[5] K. An, A. Gokhale, D. Schmidt, S. Tambe, P. Pazandak, G. Pardo-Castellote, "Content-based Filtering Discovery Protocol (CFDP): Scalable and Efficient OMG DDS Discovery Protocol," *Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, DEBS '14, ACM, New York, NY, USA, 2014, pp. 130-141.*

[6] Object Management Group, "RTI Data Distribution Service," *user's manual version 4.5. (2011).*

[7] J. Sanchez-Monedero, J. Povedano-Molina, J. M. Lopez-Vega, J. M. Lopez-Soler, "Bloom filter-based discovery protocol for DDS middleware," *in Journal of Parallel and Distributed Computing 71 (10) (2011) 1305-1317.*

[8] E. U. Michael Mitzenmacher, "Probability and Computing: Randomized Algorithms and Probabilistic Analysis," *Cambridge University Press, 2005.*

[9] D. Guo, J. Wu, H. Chen, Y. Yuan, X. Luo, "The Dynamic Bloom Filters," *IEEE Transactions on Knowledge and Data Engineering 22 (1) (2010) 120-133.*

[10] L. Fan, P. Cao, J. Almeida, A. Z. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *IEEE/ACM Trans. Netw. 8 (3) (2000).* 281-293.

[11] Object Computing Incorporated (OCI) "OpenDDS," *version 4.5 (2014).*