An Efficient Discovery Protocol of Large-Scale CPS Middleware for Real-Time Control System

Jeman PARK[1], Inwhee JOE[1], Won-Tae KIM[2]

[1] Department of Electronics Computer Engineering, Hanyang University,
17 Haengdang-dong, Seongdong-gu, Seoul, Korea,
mirrsam@hanyang.ac.kr, iwjoe@hanyang.ac.kr

[2] CPS Research Team, ETRI,
138 Gajeongno, Yuseong-gu, Daejeon, Korea,
wtkim@etri.re.kr

Abstract: A Cyber Physical System (CPS) is an autonomous embedded system based on high reliability with real-time control of distributed physical systems through wired/wireless networks. There is usually large volume of data which needs to be delivered to right places at the right time. In addition, large number of controllers in the automation and control systems are usually distributed which increases the complexity that there needs to be more point-to-point Ethernet-connections in the network. Because the controllers in the network may share control data and interact with each other from different communication protocols, including higher level operator systems. The interdependencies between these nodes may potentially create a complex architecture of the network in the distributed system especially if the point-to-point connection needs to be established. Publish-subscribe model shows some appealing properties, such as connectionless and multicast, that can be used to reduce some of the visible complexity in the software systems. Data distribution middleware for CPS should be based on a data-centric approach and guarantee real-time performance. In this regard, OMG's DDS is the best proximity middleware. RTPS (Real-Time Publish/Subscribe) is proposed for real-time service discovery in DDS. However, legacy discovery protocols cannot completely support the CPS system with a large-scale network (approx. 100,000 entities) like a warship, because service discovery messages are proportional to the square of the number of participants in RTPS. This paper proposes a scalable and fast service discovery protocol with improved discovery time for large-scale cyber physical systems based on the boot-strap algorithm and adaptive PDP message period. As a result, the proposed protocol improves reliability and real-time for service discovery in cyber physical systems. In this paper, mathematical analysis and test-bed experiments are conducted to evaluate the performance of the proposed protocol. Consequently, mathematical analysis and test-bed experiments provide almost identical results. The performance results prove that our protocol works to scale for large-scale CPS networks by minimizing the discovery time as well as traffic simultaneously

Keywords: Service Discovery, CPS, Optimal Discovery Time, Boot-strap algorithm, Adaptive period.

1. Introduction

The integration of physical systems and processes with networked computing has led to the emergence of a new generation of engineered systems [1]. A cyber physical system (CPS) is a computing system that interacts with physical processes [2, 3, 4]. Recent years have seen a growing development of embedded systems. An embedded system is a special-purpose computer system designed to perform dedicated functions for specific physical devices. They are generally assumed to be standalone. The last decade has been an explosive development period of the Internet all over the world. The advent of the Internet has raised numerous research challenges and has brought arguably the greatest technological impact on society, allowing a new communication paradigm. A CPS will bring other research challenges and social impacts, suggesting yet another new paradigm of controlling information and physical devices. There are various other research issues to consider, including real-time, privacy, reliability etc. To effectively manage such data deliver. CPS may need data-centric middleware, such as the Data Distribution Service (DDS) specification [5, 6], which makes it easy to deal with complicated data distribution. A data-centric middleware model makes it easier to address such requirements of CPS applications. The OMG Data Distribution Service (DDS) specification defines publish-subscribe middleware, which enables CPS nodes to satisfy user's QoS requirements. At the core of DDS is the Data-Centric Publish-Subscribe (DCPS) model, which defines standard interfaces that allow applications running on heterogeneous systems to read/write data to/from a global data space in a networked system. Also, Real-Time Publish Subscribe (RTPS) support interoperability and real-time transmission among different types of DDS in a networked system.

The RTPS specification [7] defines a discovery service for domain participants called the Participant Discovery Protocol (PDP) and another for matching data readers and data writers called the Endpoint Discovery Protocol (EDP). Two specific interoperable protocols (SPDP and SEDP) are layered atop RTPS using special built-in topics and data readers/writers. Once participants find each other in the domain, the SEDP should be executed. If publication/subscribe topics match, then participants initiate communication between each entity. After initiating communication, SPDP messages of (N*(N-1)) are sent periodically (N is the number of participants). This is a critical problem to support reliability and real-time. The scalability issue of such a decentralized peer-to-peer system is well-documented [8, 9, 10] in group communication. In large-scale networks, congestion occurs when many participants start the RTPS discovery protocol simultaneously. Therefore, a new participant must wait to receive the message from the built-in writers of participants in the domain. Consequentially, legacy RTPS discovery cannot support real-time in a large-scale network. In this paper, a RTPS's SPDP fast auto discovery algorithm is proposed for an improved large-scale CPS environment. In Section 2, related research about DDS and RTPS is presented. In Section 3, a fast and scalable service discovery algorithm is proposed for a CPS-based warship. In Section 4, mathematical analysis and simulation through the test bed results evaluate the proposed algorithm.

2. The DDS for Distribution Control System

The DDS is designed to address the mission-critical systems and has been widely used in both commercial and administrative field. Some properties of DDS suggest the possibility to be used in the communication of the industrial distributed control systems.

DDS-based middleware is implemented for real-time control systems [11]. It provided a new scheduling strategy as a solution for data distribution. This strategy consists of an algorithm for determining scheduling parameters and a real-time scheduler based on DDS QoS policies such as deadline, transport priorities. This implementation used real-time network as transport. This implementation could use CAN bus to send a number of sensor data samples in one message and the connectionless DDS property is well suited for complex distributed system [12].

An IEC-compliant field device model is proposed for distributed control applications [13]. The architecture of this device model is consisted with three layers. Application Execution (AE) layer is used for deployment and execution of the application's implementation model; Industrial Process-Control Protocol (IPCP) layer and Mechanical process Interface (MPI) layer which provides the communication infrastructure for the IEC-compliant device and for application to interfacing with controlled mechanical process respectively. Besides, they also presented a reference implementation on top of Real-Time CORBA Object Request Broker. However, they did modifications to the service interface function blocks. Further, Real-Time CORBA and DDS are both OMG middleware specifications, they are designed with different mechanisms for different purposes. In addition, DDS provides some features that CORBA does not, for example, the rich set of QoS provided by DDS middleware. This work will mainly focus on the DDS.

Calvo et al. presented guidelines to build communication SIFBs based on the OMG DDS middleware. They suggested first using IEC 61499 tool to define SIFB interfaces finally compiling and deploying the SIFB to an application [14]. However, they did not present any performance evaluation and applicability of the technology.

3. A Scalable Service Discovery Protocol with Optimal Discovery Time

The RTPS discovery protocol present potential scalability concerns due to the amount of data exchanges periodically and the number of multicast groups required. They also require the need to preconfigure nodes to bootstrap the discovery service. The scalability issue of such a decentralized peer-to-peer system is well-documented [15, 16] in group communication

We improved The RTPS discovery protocol by the boot-strap algorithm and adaptive PDP message period for large-scale CPS environments. Discovery messages are

dispersed by the boot-strap algorithm for collision avoidance in the bootstrap of a system. Also, discovery messages are reduced by the adaptive PDP message period after bootstrapping a system.

3.1 Participant priority

Participant priority can be defined by a user or a system. When QoS values are provided by the system, participant priority is decided by the following steps, which are applied to the Analytic Hierarchy Process (AHP) method [17].

3.1.1 Hierarchy of QoS values

QoS values are classified as reliability and real-time in DDS. QoS_Reliability and QoS_History relate to reliability. Also, QoS_Transport_Priority, QoS_Latency_Budget and QoS_Deadline refer to real-time. Figure 1 shows the hierarchy of QoS values.

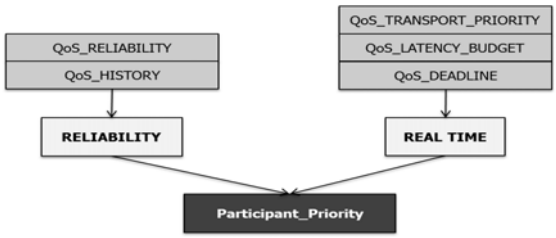

Figure 1. The hierarchical structure of QoS values

3.1.2 Pairwise comparison matrixes of QoS values

The relative magnitudes of QoS values are decided through a pairwise comparison based on human knowledge and experience. Figure 2 shows pairwise comparison matrixes of hierarchical values.

RELIABILITY VS. REAL TIME

	RELIABILITY	REAL TIME	Eigen Vector (Weight)
RELIABILITY	1	a_{12}	w_{A1}
REAL TIME	a_{21}	1	w_{A2}

RELIABILITY

	QoS RELIABILITY	QoS_ HISTORY	Eigen Vector (Weight)
QoS RELIABILITY	1	b_{12}	w_{B1}
QoS HISTORY	b_{21}	1	w_{B2}

REAL TIME

	QoS TRANSPOR TPRIORITY	QoS LATENCY BUDGET	QoS DEADLINE	Eigen Vector (Weight)
QoS TRANSPOR PRIORITY	1	c_{12}	c_{13}	w_{C1}
QoS LATENCY BUDGET	c_{21}	1	c_{23}	w_{C2}
QoS DEADLINE	c_{31}	c_{32}	1	w_{C3}

Figure 2. Pairwise comparison matrixes of hierarchical QoS values

3.1.3 Weight of participant priority

Eigen vectors of each pairwise comparison matrixes are weight values of hierarchical QoS factors. A weight of participant priority is a multiplication of each column of vectors. Table 1 shows the process of finding a weight of participant priority.

Table 1. Weight of participant priority

QoS factor	Participant Priority Weight
QoS RELIABILITY	$w_1 = w_{A1} * w_{B1}$
QoS HISTORY	$w_2 = w_{A1} * w_{B2}$
QoS TRANSPORT PRIORITY	$w_3 = w_{A2} * w_{C1}$
QoS LATENCY BUDGE	$w_4 = w_{A2} * w_{C1}$
QoS DEADLINE	$w_5 = w_{A2} * w_{C1}$

3.1.4 Classification of QoS values

In DDS, QoS value is different for each QoS factor. QoS_Reliability consists of best_effort and reliability. Additionally, QoS_History consists of *Keep_Last[k]* and *Keep_All*. Finally, QoS_Transport_Priority, QoS_Latency_Budget and QoS_Deadline have a value in the range of $0 \sim \infty$. Accordingly, the normalization of QoS value is necessary. Table 2 shows normalized QoS values.

Table 2. Applications in each class

QoS factor	variable	Qos value
QoS RELIABILITY	q_1	*BEST_EFFORT =1*
QoS HISTORY	q_2	*KEEP_LAST [K] = 1, KEEP_ALL = 0.5*
QoS TRANSPORT PRIORITY	q_3	$0 \sim \infty$ *(0.2, 0.4, 0.6, 0.8, 1)*
QoS LATENCY BUDGET	q_4	$0 \sim \infty$ *(0.2, 0.4, 0.6, 0.8, 1)*
QoS DEADLINE	q_5	$0 \sim \infty$ *(0.2, 0.4, 0.6, 0.8, 1)*

3.1.5 Decision of participant priority

Finally, participant priority is calculated by Equation 1.

$$Participant\_Priorty = \sum_{i=1}^{n} w_i q_i \quad (1)$$

3.2 Boot-strap algorithm for collision avoidance

In the legacy RTPS discovery protocol of DDS, each entity generates participant discovery messages. On boot-strap, if every entity simultaneously generates PDP messages, then critical collision occurs in a large-scale network. This collision leads to more delay at the initial participant discovery. This paper proposes our boot-strap algorithm for collision avoidance in a large-scale and time-critical network. Each participant sets a Random Back-off Time (*RBT*). *RBT* is a uniform distribution over the range between minimum back-off time and maximum back-off time. After every participant waits for Random Back-off Time, it then sends PDP messages. Equation 2 shows the *RBT*.

$$RBT_n = U(BT_{\min}, BT_{\max}) \quad (2)$$

Some participants require a time-critical service. Therefore, the *RBT* of each participant is applied to participant priority. In other words, a participant requiring real-time QoS has a short Back-off Time. Equation 3 shows the Back-off Time considered QoS.

$$QoS\_RBT = RBT_n \times Participant\_Priority \quad (3)$$

3.3 Adaptive PDP message Period for Scalable Service

In conventional DDS of OMG, SPDP messages of *(N*(N-1))* are sent periodically. As entities are increased, PDP messages are increased exponentially. Consequentially, system performance should be degraded due to collisions. Therefore, the period of PDP messages needs to adapt dynamically. In this paper, we propose our adaptive period mechanism for PDP messages. As a result, system performance is improved due to decreased total PDP messages. Figure 3 shows the period of PDP messages according to the condition of the network.

At first, the stability ration of a network can be obtained by Equation 4. There are four states for a *PDP_period*: exponential increase; numerical increase; stable state and ebbing state.

$$Stability\_Ratio = \left(1 - \frac{Leave\_Participant}{Total\_Participant}\right) \quad (4)$$

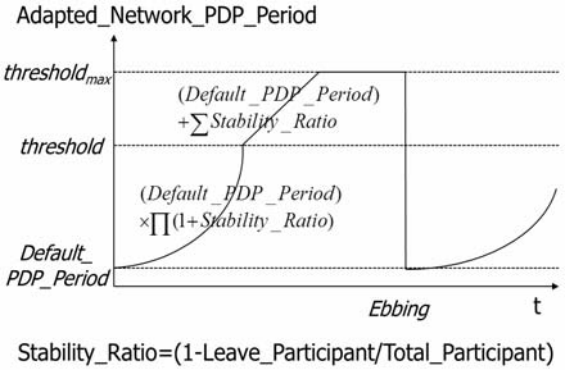

Figure 3. Period of PDP messages according to condition of the network

3.3.1 Exponential increase when joining the network

Until the threshold is reached, the period of PDP message increases by the product of *Default_PDP_period* and *Stability_Ratio*. Equation 5 shows the *Adapted_network_PDP_period* in exponential increase state. It reduces message overheads due to the increased period.

$$Adapted\_Network\_PDP\_period = Default\_PDP\_Period \times \prod(1 + Stability\_Ratio) \quad (5)$$

3.3.2 Numerical increase after threshold

In order to prevent a rapid increase of period, The *Adapted_Network_PDP_period* is increased linearly after threshold. It is defined as Equation 6

$$Adapted\_Network\_PDP\_period = Default\_PDP\_Period + \sum(1 + Stability\_Ratio) \quad (6)$$

3.3.3 Stable state on $threshold_{max}$

If a period of PDP messages increases excessively, then the performance of reliability and real-time should be reduced. Therefore, *Adpated_network_PDP_period* is static to $threshold_{max}$ such as in Equation 7.

$$Adapted\_Network\_PDP\_period = threshold_{\max} \quad (7)$$

3.3.4 Roll back to default in ebbing state

When a lot of participants leave simultaneously from the network, each participant needs to know the state of other participants immediately. Accordingly, the

Adpated_network_PDP_ period is returned to the default period such as in Equation 8

$$Adapted\_Network\_PDP\_period = Default\_PDP\_Period \quad (8)$$

Some participants require time-critical service. Therefore, a Random Back-off Time of each participant is applied to participant priority. In other words, a participant requiring real-time QoS has a short Back-off Time. Equation 9 shows the Back-off Time considered QoS

$$PDP\_period = Adapted\_network\_PDP\_period \times Participant\_Priority \quad (9)$$

Figure 4 shows overall flow for decision of the *PDP_period*.

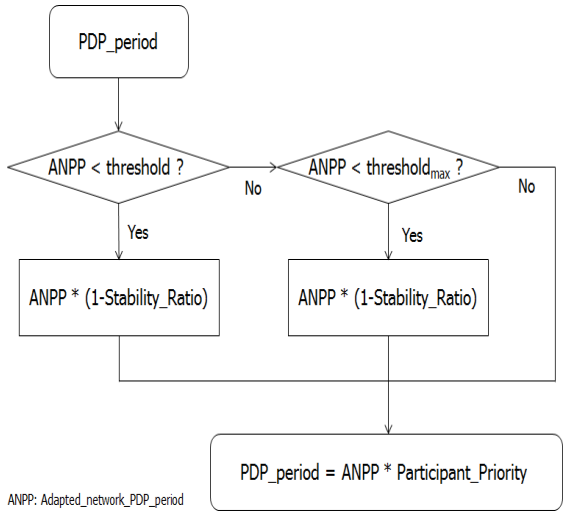

Figure 4. Flow chart for the *PDP_period*

4. Performance Evaluation

4.1 Mathematical analysis

The M/M/1/N queuing model [18] is applied for evaluation during boot-strap. It is defined as blocking the probability of participants. Finally, an optimal PDP period is obtained for full connection to each participant in the whole network. Equation 10 shows a steady state probability for an M/M/1/N queuing model.

- Single server, Poisson arrivals (rate λ)
- exponential service times (rate μ)
- λ: average arrival rate
- μ: average service rate
- ρ: utilization factor (λ/μ),
- N: total number of buffer slots (including server)
- K: number of node

State probabilities:

$$P(0) = \frac{1-\left(\frac{\lambda}{\mu}\right)}{1-\left(\frac{\lambda}{\mu}\right)^{N+1}}, \qquad P(n) = P(0)\left(\frac{\lambda}{\mu}\right)^{n} = \frac{\left(1-\left(\frac{\lambda}{\mu}\right)\right)\left(\frac{\lambda}{\mu}\right)^{n}}{1-\left(\frac{\lambda}{\mu}\right)^{N+1}} \quad (10)$$

If a system queue is full, then remaining PDP massages are blocked, therefore Participant Blocking probability is represented like Equation 11.

Blocking probability:

$$P(N) = \frac{\left(1-\left(\frac{\lambda}{\mu}\right)\right)\left(\frac{\lambda}{\mu}\right)^{N}}{1-\left(\frac{\lambda}{\mu}\right)^{N+1}} \quad (11)$$

Equation 12 shows the number of PDP rounds for full connection to each participant. *K*P(N)*, that is blocked participants among whole participants, is reduced by increasing the PDP round. When *K*P(N)* is less than 1, every participant is connected.

R: Round for full connection.

$$K \times P(N)^{R} \cong 1, \quad R \cong \log_{P(N)}\left(\frac{1}{K}\right) \quad (12)$$

An initial registration is completed after receiving the first PDP message. Nevertheless, if a participant does not receive the next *PDP_messages* during *Lease_duration*, then they will be recognized as leaving. Equation 13 shows the probability of this situation.

$$Participant\_Loss\_Probability = \left(1 - P(N)\right)^{\frac{Lease\_duration}{PDP\_Period}} \quad (13)$$

This is important for maintaining registration. This paper solves this problem by reducing Participant Blocking Probability.

Figure 5 shows the number of PDP rounds according to the Utilization Factor (a measure of how busy the system is). The Utilization Factor (ρ) is the ratio of average arrival rate of PDP message (λ) and average service rate (μ). A considerable change of the PDP round appears as a round at $\rho=1$. In general, μ is constant in the network interface (e.g. 10Mbps, 100Mbps, 1Gbps and etc.). Therefore, ρ is determined by the λ. Consequently, the arrival rate of a PDP message is an important factor for a fast and scalable discovery protocol.

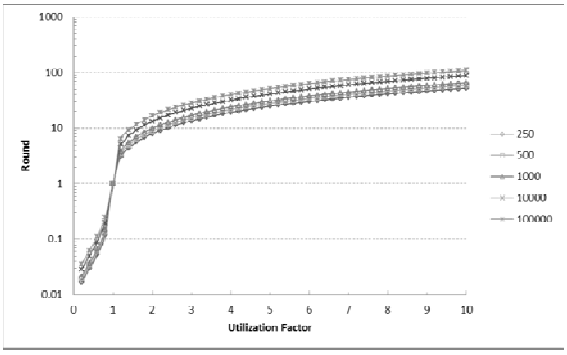

Figure 5. The number of PDP Rounds depending on the Utilization Factor

Figure 6 shows the number of PDP Rounds according to PDP period when μ is 10Mbps. The number of PDP rounds is reduced by increasing the PDP period, since λ is in inverse proportion to the PDP period. Nevertheless, total completed time is not reduced when PDP periods are decreasing. Figure 7 shows the total completed time according to PDP period. Because of the Utilization factor (ρ) is less than 1, total completed time is the PDP period in case of 250, 500 and 1000. In case of 10000 and 100000, the total completed time is decreasing to begin with, but is increasing from 3sec, 29sec. It is the time of 3sec to 29sec when the utilization factor is 1.

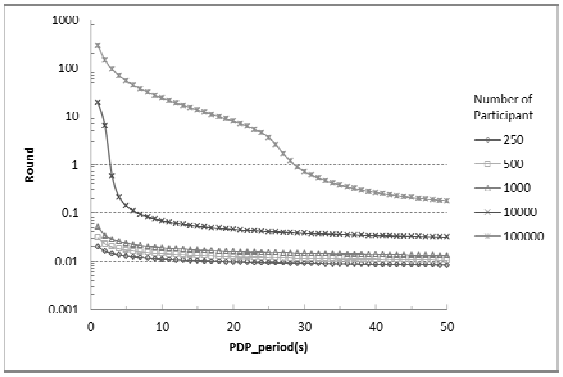

Figure 6. The number of the PDP Rounds depending on the PDP period

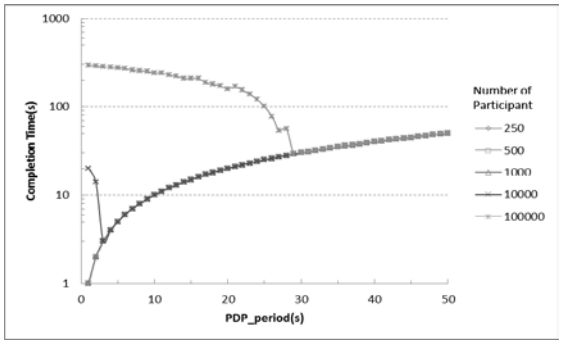

Figure 7. The Total completion time for registering whole participants depending on the PDP period

4.2 Experimental Test-bed

For analyzing the performance of the proposed Fast Auto Discovery algorithm, we configured the network which contains 5~100 participants with 10kbps link (0.1~2 *kbps/participant*). This configuration is actually equivalent to the network which contains 5,000~100,000 participants with a 10Mbps link (0.1~2 *kbps/participant)*. The number of rounds for registering all the participants depends on the number of participants. The difference between the test-bed and the actual situation is calculated by Equation 14.

$$R_1 = \log_{P(N)}\left(\frac{1}{K_1}\right), \quad R_2 = \log_{P(N)}\left(\frac{1}{K_2}\right)$$
$$R_1 = R_2\left(1+\log_{K_2}\left(\frac{K_1}{K_2}\right)\right) \tag{14}$$

Figure 8 is the result of the test-bed and Figure 9 is the simulated result using the Equation 14. Both graphs demonstrate the number of rounds depending on the number of the participants. Figure 10 is drawn theoretically by using Equation 12. As shown, when the number of participants is small, the number of the rounds varies, largely because of the errors caused by abnormal time-distribution. When the number of the participants is large, Figure 9 and Figure 10 show similar shapes.

Figure 11 demonstrate the latency for registering whole participants and the message generation rate respectively. We calculated the optimal completion time by using the variation of the number of the participants and the distribution of the period of PDP messages. As shown in Figure 12, if the distribution of the period of the PDP messages increases, the message generation rate decreases. Therefore, the overall message overhead can be reduced by discovering the optimal period and applying our proposed algorithm.

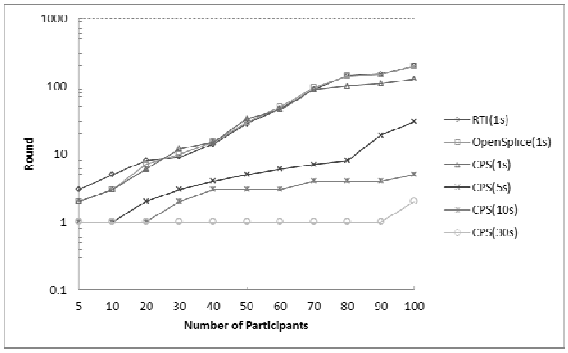

Figure 8. The number of PDP Rounds depending on the number of the participants (Testbed)

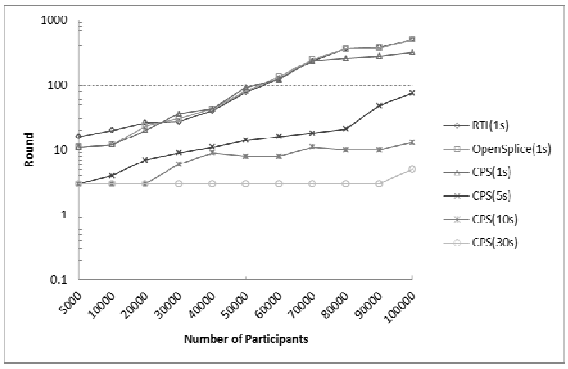

Figure 9. The number of PDP Rounds depending on the number of the participants (Test-bed with Equation (14))

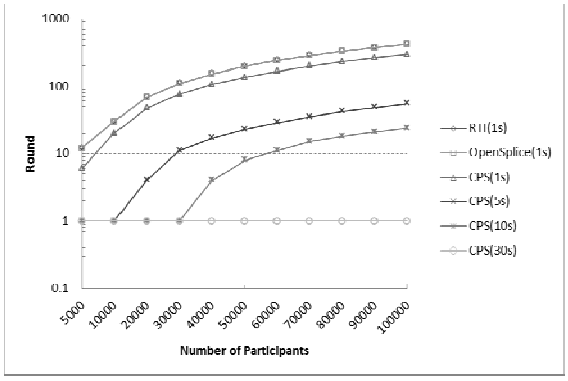

Figure 10. The number of the PDP Rounds depending on the number of the participants (Theoretical Values)

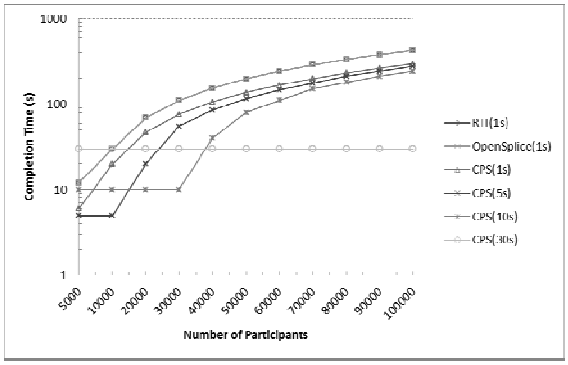

Figure 11. The completion time for registering whole participants depending on the number of the participants

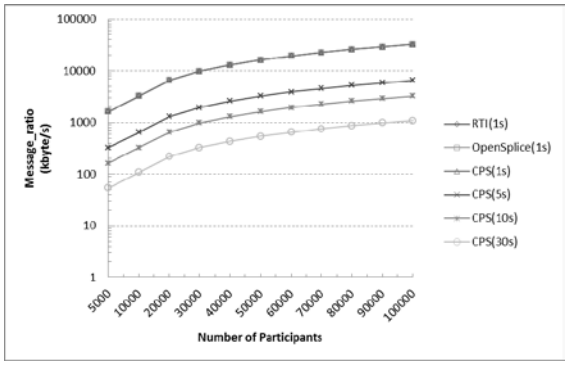

Figure 12. The PDP message generation rate depending on the number of participants

5. Conclusion

This paper proposes an efficient discovery protocol of Large-Scale CPS middleware for real-time control system. The boot-strap algorithm and adaptive PDP message period with participant priority are applied to the RTPS protocol. The number of discovery messages and congestion of a network were reduced. Also, the stability of the network improved. The proposed scalable service discovery protocol is stable when the first discovery is performed, since it can distribute discovery messages effectively. As a result, the proposed protocol improves scalability and discovery time for service discovery in the cyber physical system. Mathematical analysis and testbed experiments were applied to prove performance. According to the performance results, our protocol works to scale for large-scale CPS networks by minimizing the discovery time and traffic simultaneously.

Acknowledgements

This work was supported by the IT R&D Program of MSIP/KEIT [10035708, "The Development of CPS (Cyber-Physical Systems) Core Technologies for High Confidential Autonomic Control Software"]

REFERENCES

1. KROGH, B.H., E. LEE, I. LEE, A. MOK, R. RAJKUMAR, L.R. SHA, A.S. VINCENTELLI, K. SHIN, J. STANKOVIC, J. SZTIPANOVITS, W. WOLF, W. ZHAO, **Cyber-Physical Systems, Executive Summary,** CPS Steering Group, Washington D. C., 2008.

2. LEE, E., **Cyber Physical Systems: Design Challenges,** University of California, Berkeley Technical Report No. UCB/EECS, Aug. 2008.

3. LEE, E., **Computing Foundations and Practice for Cyber-Physical Systems: A Preliminary Report,** University of California, Berkeley Technical Report, UCB/EECS-2007-72, May. 2007.

4. ZHANG, L., J. HE, W. YU, **Challenges, Promising Solutions and Open Problems of Cyber-Physical Systems,** IJHIT Vol. 6, No.2, March 2013.

5. PARDO-CASTELLOTE, G., **OMG Data-Distribution Service: Architectural Overview,** Proceeding of the 23rd International Conference on Distributed Computing Systems Workshops, 2003.

6. MOKADEM, R., A. HAMEURLAIN, **An Efficient Resource Discovery while Minimizing Maintenance Overhead in SDDS Based Hierarchical DHT Systems,** IJGDC Vol. 4, No. 3, September 2011.

7. **OMG DDS Interoperability Protocol (DDS-RTPS) Specification,** http://www.omg.org/spec/DDS-RTPS/2.1/PDF/.

8. SCHMIDT, D. C., A. CORSARO, H. V. HAG, **Addressing the Challenges of Tactical Information Management in Net-Centric Systems with DDS,** The Journal of Defense Software Engineering, Mar. 2008.

9. YIM, H., Y. SON, C. SONG, C. KIM, **A Meta-Model Transformation between DDS and DBMS Representation of Data for DDS-DB Integration,** IJSEIA Vol. 7, No.3, May 2013.

10. GU, J., G. CHEN, **Design of Physical and Logical Context Aware Middleware,** IJSIP Vol. 5, No. 1, March 2012.

11. GUESMI, T., R. REKIK, S. HASNAOUI, H. REZIG, **Design and Performance of DDS-based Middleware for Real-Time Control Systems,** International Journal of Computer Science and Network Security, Vol.7, No.12, December 2007, pp. 188-200.

12. REKIK, R., S. HASNAOUI, **Application of a CAN BUS transport for DDS Middleware,** 2nd International Conference on the Application of Digital Information and Web Technologies (ICADIWT), August 4-6, 2009, pp. 766-771.

13. THRAMBOULIDISET, K. C., G. S. DOUKAS, T. G. PSEGIANNAKIS, **An IEC-Compliant Field Device Model for Distributed Control Applications**, 2nd IEEE International Conference on Industrial Informatics (INDIN), June 24-26, 2004, pp. 277-282.

14. Calvo, I., F. PÉREZ, I. ETXEBERRIA, G. MORÁN, **Control Communications with DDS using IEC61499 Service Interface Function Blocks**, IEEE Conference on Emerging Technologies and Factory Automation (ETFA), September 13-16, 2010, pp. 1-4.

15. AMIR, Y., C. DANILOV, J. R. STANTON. **A Low Latency, Loss Tolerant Architecture and Protocol for Wide Area Group Communication,** In Proceedings of International Conference on Dependable Systems and Networks, June 2000, pages 327-336.

16. LEE, S., Y. KO, D. LEE, **Realization of a Scalable and Reliable Multicast Transport Protocol for Many-to-Many Sessions**, ETRI Journal, vol. 29, no. 6, Dec. 2007, pp. 745-754.

17. SAATY, T. L., **Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process,** RWS Publications, 2000.

18. TIJMS, H. C., **A First Course in Stochastic Models,** Wiley, 2003.