IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, May 2011
ISSN (Online): 1694-0814
www.IJCSI.org

Efficient Application-Level Causal Multicasting for Pre-Planned Sensor Networks Using Resynchronization Method

Chayoung Kim[1] and Jinho Ahn[2*]

**[1] Contents Convergence Software Research Center, Kyonggi University
Suwon, Gyeonggi-do 443-760, Republic of Korea**

**[2] Dept. of Computer Science, College of Natural Science, Kyonggi University
Suwon, Gyeonggi-do 443-760, Republic of Korea**

Abstract

In this paper, we present a causal order guaranteeing multicast protocol based on sensor brokers as publishers that aggregate the information of results in sensor networks, periodically gossiping about the result messages to the query nodes according to their interesting topics. Although subscribers' join and leave are highly perturbed in such sensor networks, some sensors can form overlapping multicast groups and query nodes as subscribers receive the results of the queries based on gossip-style dissemination. Each sensor broker manages a vector for each overlapping multicast group that represents its knowledge for every member of the group, and uses a vector whose dimension is the number of groups to time-stamp message. Each sensor broker needs piggybacking only one value per group for each message, which carries only one vector. Also, the broker makes information of the causally ordered delivery list and attaches the list to all messages. It gossips about them to their subscribers in overlapping multicast groups Therefore, this protocol guarantees causally ordered delivery of messages in a highly reliable manner using gossip-style dissemination technique.

Keywords: *Sensor Network, Reliable Group Communication, Overlapping Groups, Scalability, Resynchronization.*

1. Introduction

Wireless sensor networks(WSNs) are gaining high attention from academia and industry with its potentially infinite applicability to a lot of areas, being actively researched for energy-efficiency and optimization in various aspects. Their architectural styles are categorized in two approaches, structured and unstructured. An unstructured WSN is one that constrains a dense collection of sensor nodes, deployed in an ad hoc manner into the field. A structured WSN consists of all or some of the sensor nodes which are deployed in a pre-planned manner [18]. Especially, in terrestrial pre-planned deployment, there is grid placement, optimal placement, 2-d and 3-d placement model [18]. The challenge is how to gain the most meaningful information from the data collected by the distributed sensor nodes, to make decisions in a reliable and efficient manner. Therefore, the implementation of the communication protocols such as data aggregation, data propagation and collaborative interaction processing, explored as sensor network services, can significantly affect energy consumption and end-to-end delay in WSNs [13, 18]. Data aggregation and compression reduce communication cost and increase reliability of data transfer and aid in reducing the amount of data to be transferred [13, 18]. And these above applications in WSN need a variety of collaboration features, such as chat windows, white boards, p2p video and other media streams, and coordination mechanisms [13, 14, 15, 18]. Therefore a new data dissemination paradigm for such sensor networks is required to handle data propagation and aggregation generated by a very large number of sensor nodes in an efficient manner [10, 16]. There are several researches based on the P (publish)/S (subscribe) paradigm in the area of sensor network communications to address the problem of querying sensors from subscribing nodes in order to minimize the number of sent result packets [10, 16]. In P/S paradigm systems, a query node periodically runs an algorithm to identify the sensors it wishes to track and "subscribe" to these sensors of its interesting topics, and the sensors periodically "publish" [10, 16]. In a grid cell of WSN, sensor nodes might lead to making overlapping multicast groups organized by subscriber's interests and using this grouping, a priori known to all subscriber nodes, query nodes can be quite easily mapped onto topics [16]. These overlapping groups are prevailing in such networks and forcing researchers to reconsider issues about novel types of group communication facilities to satisfy the complicated requirements stated earlier [16, 18]. In this paper, we present a causal order guaranteeing multicast protocol based on sensor brokers as publishers that aggregate the information of results in sensor networks, periodically gossiping about the result messages to the query nodes to subscribe according to their interest topics. Although subscribers join and leave are highly perturbed in such sensor networks, some sensors can form overlapping multicast groups and query nodes as subscribers receive the results of the queries based on gossip-style dissemination. Each sensor broker manages a vector per overlapping multicast group, that represents its knowledge for each member of the group and use a vector whose dimension is the number of groups to time-stamp message. Each sensor broker needs for each message piggybacking of only one value per group, so a message carries only one vector. Also, the broker makes information of the causally ordered delivery list and attaches the list to all messages. It gossips about them to their subscribers in overlapping multicast groups. This protocol guarantees causally ordered delivery of messages in a highly reliable manner using gossip-style dissemination protocols, yet still achieving high degree of both scalability and reliability rather than those of the protocols by traditional reliable group communication.

2. System Model

In the distributed system, a group consists of a set of processes. Processes join and leave the system dynamically and have ordered distinct identifiers. The process maintains a local membership list called a "local view". It can send unicast messages to another process through the communication network. A finite set of processes communicate only by exchanging messages over a fully connected, point-to-point network. Processes communicate using the primitives *send(m)* and *receive(m)*. Communication links are fair-lossy, but correct processes can construct reliable communication links on top of fair-lossy links by periodically retransmitting messages. Each member performs operations according to a local clock. Clock rates at all members are the same. Runs of the system proceed in a sequence of rounds. Members may undergo two types of failures, both probabilistic in nature. The first is process failure. There is an independent, per-process probability of at most that a process has a crash failure during the finite duration of a protocol. Such processes are called faulty. Processes that survive despite the failures are correct. The second type of failures is message omission failure. There is an independent, per-message probability of at most δ that a message between non-faulty processes experiences a send omission failure. The union of all message omission failure events and process failure events are mutually independent. For simplicity, we do not include process recovery in the model. Also, we expect that both and δ are small probabilities. There are no malicious faults, spurious messages, or corruption of message i.e. we do not consider Byzantine failures.

In proposed protocols, a group of processes is defined through two primitives, PMCAST and PDELIVER, which use gossip protocols to provide probabilistic reliability in networks. Processes communicate with these two pairs of primitives, PMCAST and PDELIVER, which model unreliable communication associated with probability α of successful message transmission. We refer to probability α as the expected reliability degree. These primitives are as follows: (Integrity) For any message *m*, every correct process PDELIVER *m* at most once, and only if *m* was previously PMCAST by *send(m)*. (Validity) If a correct process *p* PMCASTs a message *m* then *p* eventually PDELIVERs *m*. (Probabilistic Agreement) Let *p* and *q* be two correct processes. If *p* PDELIVERs a message *m*, then with the probability of α, *q* PDELIVERs *m*. In other terms, the only probabilistic property is Agreement. This probabilistic notion of agreement also captures a weakly consistent membership of local view, typical for large scale settings.

3. The New Application-Level Causal Multicast

3.1 Basic Idea

We assume that in our proposed protocol based on sensor brokers, some sensors designed as brokers can form overlapping multicast groups and query nodes subscribe to the brokers publishing their interest topics. The mapping of subscribers and brokers (publishers) is entirely driven by the application matching their interest queries. Recently, much research has been devoted to designing broker selection methods that best suits application needs [10, 16]. Common sensors can update information periodically to some of its brokers based on gossip communication protocols or other highly reliable communications and the brokers might aggregate the query results by combining reports from several sensors [10, 16]. There is a two-dimensional area of interest (AoI), which the sensor broker publishes messages to a particular topic, while query nodes subscribe to all the topics that match their interests. It is possible that there are two or more brokers in a grid and each broker can know every other broker. All brokers representing a sensor grid and having interests of their subscribers can participate in overlapping multicast groups. But, if all subscribers in a grid lose their interests in information published on a particular topic, the brokers representing the grid send leave messages to the corresponding overlapping multicast groups and then all of their group members are updated by leaving brokers. The sensor brokers periodically gossip about the messages of the results, while guaranteeing the message delivery order, such as causally ordered delivery, with aggregating the information of the results in overlapping multicast groups. In this protocol based on sensor brokers, we present a causal order guaranteeing multicast protocol supporting overlapping multicast groups and useful for distributed applications with a variety of collaboration features, such as chat windows, white boards, p2p video and other media streams, and coordination mechanisms requiring causally ordered delivery of messages. In this protocol, because every broker knows every other brokers, it manages a vector per group, that represents its knowledge for each member of the group, of the number of message multicast by this member within this group, as same as a member in the protocol of Birman et. al. [2]. In the protocol [2], each member, p_i has to manage a vector per group and each message has to piggyback the whole set of these vectors. It is correct but expensive. Therefore, we propose the protocol similar to that of [12] that needs for each message piggybacking of only one value per group. So a message carries only one vector whose dimension is the number of groups. There is a trade-off between the optimality in terms of the delay in the delivery of messages and the size of vectors carried by messages. The choice of a particular protocol actually depends on various kinds of factors in the distributed applications [12].

For data dissemination between sensor brokers as publishers and subscribers, every sensor broker makes the causally ordered delivery list by aggregating information based on VTs of overlapping multicast groups and attaches the list to all messages. It gossips about the result messages with the causally ordered delivery list to their subscribers and periodically updates the digest information of causally ordered delivery list using gossip protocols. Therefore, this proposed protocol might result in its very low cost compared with the cost incurred by traditional reliable group communication protocols [2] because this protocol makes up transient message losses between publishers and subscribers and deals with subscribers' leave using gossip communication protocols[16]. That is, the processing of traditional group communication protocol [2] is different from that of the protocol based on gossip [3, 5]. This assumption has led to rigorously establish many desirable features of gossip protocols like scalability, reliability, and efficiency and a wide range of higher functions, which include information dissemination, aggregation, and network management [10, 16].

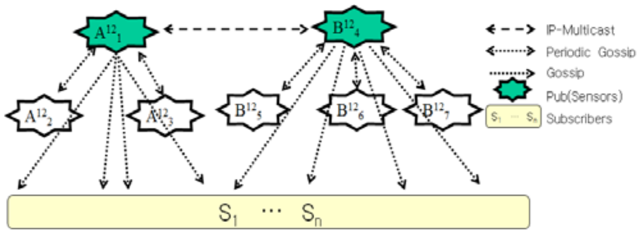


Fig. 1 An example of each sensor broker publishing messages to subscribers by gossip-style disseminations

3.2 Detailed Description

In figure 1, we can see that each sensor broker $A^{12}{}_1$, $B^{12}{}_4$ involved in overlapping multicast groups, Group1 and Group2, respectively in a grid A and B of a sensor network. Each broker publishes desired messages of query results to all query nodes subscribing to their brokers based on interest topics by gossip-style disseminations. Figure 2 shows that all sensor brokers implement the rules that guarantee causally ordered delivery of messages in a sensor network like in Fig. 1. In this paper, we implement the protocol similar to that of [12] in each broker. In figure 2, G_i is the set of groups that p_i is member of.

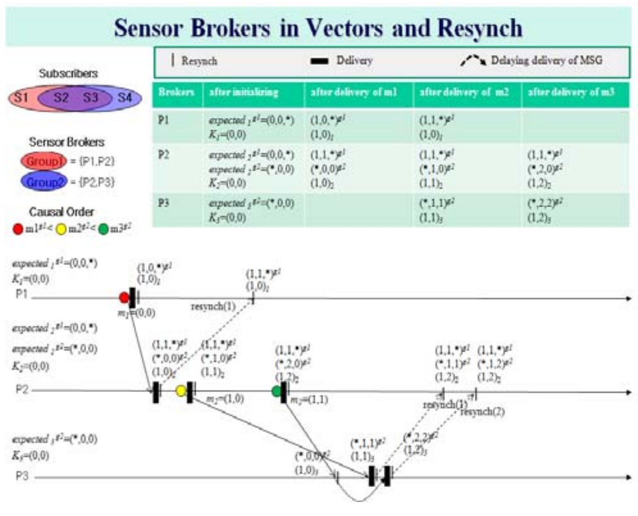


Fig. 2 An example of each sensor broker processing the information for the causally ordered delivery based on the highest time-stamped value and each vector per group

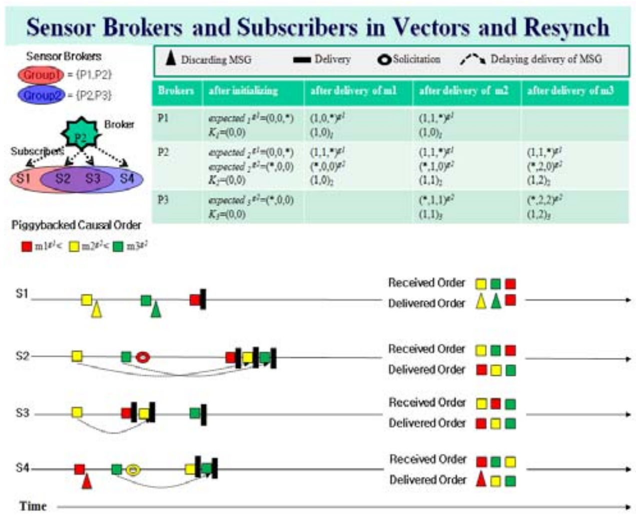
Each sensor broker p_i manages a local array $expected^{gx}{}_i[1..n]$(initialized to (0, ..., 0)) where n is the number of broker members constituting the group, g_x; an entry has the following meaning: [12]

for $g_x \in G_i$, $p_j \in g_x$:
$expected^{gx}{}_i[j] = \boldsymbol{\alpha}$: It indicates p_i's knowledge that the next timestamp used by p_j to multicast within g_x will be greater or equal to $\boldsymbol{\alpha}$.

In order to guarantee causally ordered delivery of messages, each p_i manages a vector K_i of size |G| (one entry per group) and uses the vector K_i to timestamp messages. The meaning of K_i initialized to (0, ..., 0) is as follows:

$K_i[x] = \boldsymbol{\alpha}$: $\boldsymbol{\alpha}$ is the highest timestamp value concerning g_x and known by p_i.
If $p_i \in g_x$, $K_i[x] = expected^{gx}{}_i[i]$ holds[12].

This "$expected^{g1}{}_1 = (0, 0,*)$" means as follows: "process p_1 belongs to g_1 and each entry, p_1 and p_2 is a member of group g_1 and p_3 does not participate in the group g_1". For each message generated by a member, each vector, $expected^{gx}{}_i[j]$ and $K_i[x]$ is incremented by one respectively. So, if a member p_1 belonging to g_1 generates a multicast message, then $expected^{g1}{}_1$ and K_1 is $(1, 0, *)$ and $(1, 0)$ respectively. In the causal order of $p_1(m_1^{g1})\text{->}p_2(m_2^{g2})\text{->}p_2(m_3^{g2})$ of Fig. 3, sensor brokers aggregate the information about the causally ordered delivery list based on the following condition becoming true: for $\forall g_x \in G_i$, $\forall p_j \in g_x$: $expected^{gx}{}_i[j] \geqq K_i[x]$. When p_3 knows there is a message m_2^{g2} preceding m_3^{g2} by verifying piggybacked vector information, p_3 should delay the delivery of the message m_3^{g2} after the receipt of the predecessor, m_2^{g2}.



Brokers	after initializing	after delivery of m1	after delivery of m2	after delivery of m3
P1	$expected_1^{g1}=(0,0,*)$ $K_1=(0,0)$	$(1,0,*)^{g1}$ $(1,0)_1$	$(1,1,*)^{g1}$ $(1,0)_1$	
P2	$expected_2^{g1}=(0,0,*)$ $expected_2^{g2}=(*,0,0)$ $K_2=(0,0)$	$(1,1,*)^{g1}$ $(*,0,0)^{g2}$ $(1,0)_2$	$(1,1,*)^{g1}$ $(*,1,0)^{g2}$ $(1,1)_2$	$(1,1,*)^{g1}$ $(*,2,0)^{g2}$ $(1,2)_2$
P3	$expected_3^{g2}=(*,0,0)$ $K_3=(0,0)$		$(*,1,1)^{g2}$ $(1,1)_3$	$(*,2,2)^{g2}$ $(1,2)_3$

Fig. 3 An example of each sensor broker gossiping about messages including a causal order delivery list for subscribers

Figure 3 shows that all sensor brokers gossip about messages piggybacking a causal order delivery list to subscribers associated with their interesting topics. This example of Fig. 3 illustrates sensor brokers(publishers) , $g_1=\{p_1,p_2\}$ and $g_2=\{p_2,p_3\}$ and subscribers=$\{s_1,s_2,s_3,s_4\}$. The subscribers=$\{s_2,s_3\}$, overlapped in two groups receive all messages from g_1 and g_2. Subscriber s_1 receives messages only from g_1 and s_4 receives messages only from g_2. On receiving gossip messages, a subscriber validates its receipt of predecessor messages according to causal order delivery list, $m_1^{g1}\text{->}m_2^{g2}\text{->}m_3^{g2}$ piggybacked by the messages. Also, there are undesired messages are sent to a subscriber, forcing it to discard them. When s_1 belonging to g_1 receives gossip messages from broker p_2, it discards m_2^{g2} and m_3^{g2} without delivering them to the application layer because s_1 does not belong to g_2. Subscriber s_2 requests m_1^{g1} to the latest gossip sender after receiving m_2^{g2} and m_3^{g2} because it knows that the predecessor, m_1^{g1} is not received by verifying piggybacked causal order list, $m_1^{g1}\text{->}m_2^{g2}\text{->}m_3^{g2}$. Subscribers$_4$ requests m_2^{g2} to the latest gossip sender after receiving m_3^{g2} but it discards m_1^{g1} because it does not belong to g_1.

Figure 4 shows that a sensor broker, p_i implements the proposed protocol in this paper. In procedure INITIALIZE, p_i initializes vectors $expected^{gx}{}_i[j]$ and $K_i[x]$ for each group g_x. In procedure SEND_MULTICAST, p_i sends multicast messages to broker members constituting each group, g_x. In procedure SEND_GOSSIP, p_i gossips about multicast messages to subscribers randomly selected as gossip-targets in its local view. If there are no new messages, p_i gossips about the summary of causal order delivery list. In procedure RECEIVE_RESYNCH, on receiving resynch (the highest timestamp information) message from p_j, p_i updates $expected^{gx}{}_i[j]$ accordingly, like the protocol of [12]. In procedure RECEIVE_MULTICAST, p_i receives multicast messages from p_j and processes them according to piggybacked vectors. So, p_i puts the received message into the *Pending_List* and then updates $expected^{gx}{}_i[j]$ to $k+1$. If necessary, $expected^{gx}{}_i[j]$ is updated and the new timestamp is multicast using resynch message. If the delivery condition, $\forall g_x \in G_i$, $\forall p_j \in g_x$: $expected^{gx}{}_i[j] \geqq K_i[x]$ is satisfied and the message is delivered to the application layer. In procedure SEND_SOLICITATION, subscriber s_i requests the not-received predecessor messages to the latest gossip sender, p_i, after verifying piggybacked causal order list from sensor broker p_i. In procedure RECEIVE_SOLICITATION, on receiving solicitation message, p_i gossips about the requested messages by attaching causal order lists to the subscriber s_i.

4. Performance Evaluation

In this section, we compare average throughput of our protocol based on gossip-style dissemination protocol between sensor brokers and subscribers with that of a previous protocol based on traditional reliable group communication using resynch messages carrying a timestamp indicating that the next phase in a member will possibly send messages [12]. In this comparison, we rely on a set of parameters referred to Bimodal Multicast [3] and LPBCast [5] for gossiping parameters.

And we assume that processes gossip in periodic rounds, gossip period is constant and identical for each process and maximum gossip round is log N. The probability of network message loss is a predefined 0.1% and the probability of process crash during a run is a predefined 0.1% using UDP/IP. The group size of each sub-figure is 32(2), 64(4), 128(8) and 256(16).

```
Procedure INITIALIZE
Expected[$i$]$^{gx}$ (Vector timestamp for each group $g_x$)
K[$g_x$] (MAX timestamp values for each group $g_x$)


Procedure SEND_MULTICAST
for all interest groups, $x$ do
Unreliable_Multicast($msg$, K[$g_x$], $x$) to $g_x$
Expected[$i$]$^{gx}$ = Expected[$i$]$^{gx}$ + 1
K[$g_x$] = Expected[$i$]$^{gx}$

Procedure SEND_GOSSIP
select subscribers as gossip-target in Local_View
for each subscriber do
DIGEST = the summary of delivered msgs in causal order
Gossip_MSG($msg$ with DIGEST)
if Periodically GOSSIP then
Gossip_MSG(DIGEST)
do Garbage_Collection

Procedure RECEIVE_RESYNCH($time$)
Expected[$j$]$^{gx}$ = $timestamp$

Procedure RECEIVE_MULTICAST
put ($msg$, K[$g_x$], x) in Pending_List
Expected[$j$]$^{gx}$ = K[$g_x$] + 1
if Expected[$i$]$^{gx}$ < K[$g_x$] + 1 then
Expected[$i$]$^{gx}$ = K[$g_x$] + 1
K$_i$[$g_x$] = K[$g_x$] + 1
Multicast_Resynch(K[$g_x$] + 1)
for every $msg$ in Pending_List do
for all interest groups, $x$ do
Expected[$j$]$^{gx}$ >= K[$g_x$] then
deliver $msg$ to the application
remove $msg$ from Pending_List
K$_i$[$g_y$]=$max$(K[$g_y$],K$_i$[$g_y$]) in all groups, $y$, not interested
call Procedure SEND_GOSSIP

Procedure SEND_SOLICITATION
check DIGEST piggybacked with msg
if there are msg, not received then
call for SOLICITATION to the Sensor_Broker

Procedure RECEIVE_SOLICITATION
DIGEST = the summary of delivered msgs in causal order
Gossip_MSG($msg$ with DIGEST) to the Subscriber
```

Fig. 4 Formal form of our proposed protocol

Figure 5 shows the average throughput as a function of perturb rate for various group sizes. The x-axis is the group size (the number of overlapping groups) and the y-axis is the number of messages processed in the perturb rate, (a)20%, (b)30%, (c)40% and (d)50%. In the four sub-figures from 6(a) to 6(d), the average throughput of causally ordered delivery protocol based on sensor broker(publishers) by gossiping about messages to subscribers is not a rapid change than that of the protocol based on traditional reliable group communication by resynch messages [12] among members. Especially, the two protocols are compared to each other in terms of scalability by showing how many members execute by phases; in each phase each member multicasts exactly one message. In perturbed networks with members join and leave, synchronous multicast-based executions [12] are very expensive because events of sending and receiving messages are governed by application that do not always progress synchronously.

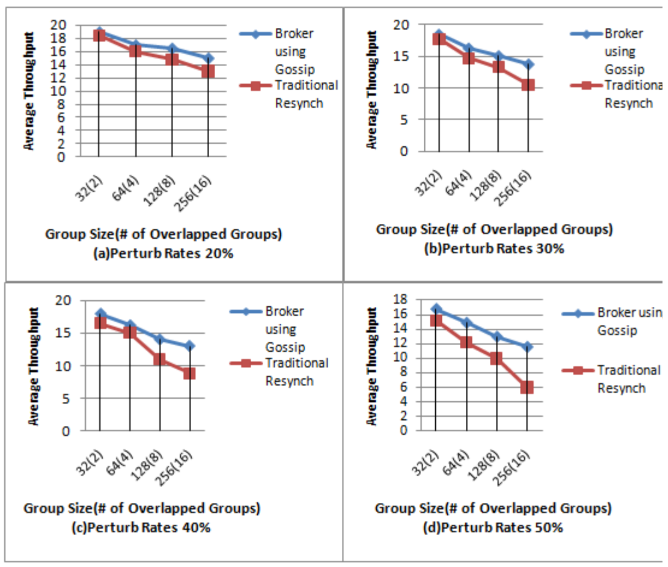


Fig. 5 Simulation results

The proposed protocol based on sensor brokers is more scalable because communications between brokers (publishers) and subscribers are based on gossip-style disseminations and sensor brokers among themselves use traditional reliable group communications. We know that which approach is more preferable depends on the user applications. If the underlying network is a communication bus, multicasting resynch messages to all the other members consists of only one operation. But, in the network layers not using some sort of ACK mechanism to ensure reliability, such a use of additional resynch message is very expensive. In gossip-style dissemination networks, there is no ACK mechanism because members periodically gossip about the summary of received messages. So, we argue that gossip-style dissemination approach outperform computations in a distributed fashion without any synchronous computation and in terms of memory overhead. There are no needs of big memory between brokers (publishers) and subscriber because subscribers receive aggregated causal order delivery list from them without any information of computation.

5. Related Work

Recently, there is a multicast platform based on gossip technique, Quicksilver Multicast Platform [15]. QSM is built in two layers. One extends a system such .NET to support live objects by embedding them in the .NET common language runtime, as well as focusing on the hooks connecting the objects to the .NET type system and the Windows shell (the GUI that interprets mouse actions). Quicksilver's second layer provides the scalable and extensible communication infrastructure needed to make the objects "live" and "distributed" [14]. Live distributed objects [14] are designed to offer developers a scalable multicast infrastructure such as QSM that's tightly integrated with a runtime environment. A live object can be understood as a distributed mechanism through which a group of software components communicate with each other, share data, exchange events, and coordinate actions in a decentralized, p2p fashion. A live object can represent, for example, a streaming video, a news channel, a collaboratively edited document, a replicated variable, or a fault-tolerant service. If live objects were to take off, they could be the gateway to an active, trustworthy Web. An active Web based on live objects could be a world with millions of IPTV streams and live electronic health records that integrate regional medical providers, or banking systems that could trade "live" financial instruments. And such kind of collaborative applications will need to combine two types of content: traditional web service hosted content, such as data from geographic and topologic map servers, image repositories, and other databases, with a variety of collaboration features, such as chat windows, white board, p2p video and other media streams, and coordination mechanisms [9].
Early work in gossip-style protocol, Birman et al. [3] proposes bimodal multicast thanks to its two phases: a "classic" best-effort multicast such as IP multicast is used for the first rough dissemination of messages. The second phase assures reliability with a certain probability by using a gossip based retransmissions. But Lpbcast [5] proposes gossip-style broadcast membership mechanisms based on a partial view without a global view. Each process has a randomly chosen local view of the system. Lpbcast [5] is a

completely as a decentralized membership protocol because of no dedicated messages for membership management based on gossips. In P. Eugster et. al. [6], the protocol deals with the case of multicasting events only to subsets of the processes in a large group by relying on a specific orchestration of process as a superimposition of spanning trees.

As a fundamental problem in distributed computing, much effort has been invested in solving atomic broadcast. Early work such as [2] mostly focuses on stronger notions of Agreement and also membership than the proposed protocols discussed in this paper. Also, there are some works to solve atomic delivery order in gossip-style protocol such as atomic probabilistic broadcast (apbcast) [7], a hybrid approach implemented for publish/subscribe programming. Its deterministic ordering of messages ensures the consistency of the delivery order of broadcast messages and its probabilistic propagation of broadcast messages and order information provides a high level of reliability in the face of an increasing number of process failures because of more heroic efforts by making use of the membership of delegates. Probabilistic Atomic Broadcast (pabcast) [8] is fully probabilistic by mixing message atomic ordering and propagation, basing these on gossips without a membership of delegates. But, a promising approach for increasing scalability is to weaken the deterministic ordering guarantees to make the properties of dependencies between broadcast messages probabilistic. Also, it does not give the guarantees achieved for the consistency of the delivery order of overlapping groups.

To ensure the causal order in [2], each process manages a vector of integers per group and each message is time-stamped with the whole set of vectors of the sending process. This protocol is correct but expensive. So, in [12], the paper proposes a protocol that needs for each message the piggybacking of only one value per group, so a message carries only one vector of integers whose dimension is the number of groups. In a real system, the choice of a particular protocol actually depends on several factors. We use the protocol, like that of [12] because our protocol stands for structured sensor networks in a pre-planned manner.

And there are researches based on the P (publish)/S (subscribe) paradigm in the area of sensor network communications to approach the problem of querying sensors from mobile nodes [10, 16]. Directed Diffusion [10] can be seen as publish-subscribe mechanism, which is implemented using the tree-based architecture rooted at the publisher. SENSTRACT [16] is mapping from queries to topics and the corresponding underlying sensor network structure. SENSTRACT [16] is a tree-based P/S system structured by service providers as roots, representing one of the data-centric routing protocols for data dissemination of sensor networks. Cross Reality is about connecting "location-specific 3D animated constructs" in virtual worlds to in-building sensors [11]. MIT has also created a whole portal network that maps sensors to virtual worlds, called the Ubiquitous Sensor Portal [11]. There are 45 portals currently in the Media Lab, each one featuring a myriad of environmental sensors - such as motion, light and sound level, vibration, temperature, and humidity. They have a small touch-screen display and audio speaker, for user interaction.

6. Conclusions

In this paper, we present a causal order guaranteeing multicast protocol based on sensor brokers as publishers that aggregate the information of results in sensor networks, periodically gossiping about the result messages to the query nodes to subscribe according to their interest topics. Although subscribers join and leave are highly perturbed in such sensor networks, some sensors can form overlapping multicast groups and query nodes as subscribers receive the results of the queries based on gossip-style dissemination. Each sensor broker manages a vector per overlapping multicast group that represents its knowledge for each member of the group, and uses a vector whose dimension is the number of groups to time-stamp message. Each sensor broker needs for each message piggybacking of only one value per group, so a message carries only one vector. Also, the broker makes information of the causally ordered delivery list and attaches the list to all messages. It gossips about the aggregated result messages based on topics with the information of causal order list to their subscribers. Therefore, this protocol guarantees causally ordered delivery of messages in a highly reliable manner using gossip-style dissemination protocols, yet still achieving high degree of both scalability and reliability rather than those of the protocols by traditional reliable group communication.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on Sensor Networks”, IEEE Communications Magazine. Vol. 40, No. 8, 2002, pp. 102-114.

[2] K. Birman, A. Schiper and P. Stephenson, "Lightweight Causal and Atomic Group Multicast", ACM Transactions on Computer Systems, Vol.9, No.3, pp.272-314, 1991.

[3] K. Birman, M. Hayden, O. Ozkasap. Z. Xiao, M. Budiu and Y. Minsky, "Bimodal Multicast," ACM Transactions on Computer Systems, Vol.17, No.2, pp.41-88, 1999.

[4] D. Culler, D. Estrin, and M. Srivastava, “Overview of Sensor Networks”, IEEE Computer, Vol. 37, No. 8, pp.41-49, 2004.

[5] P. Eugster, R. Guerraoui, S. Handurukande, P. Kouznetsov, and A.-M. Kermarrec, "Lightweight probabilistic broadcast",

ACM Transactions on Computer Systems, Vol.21, No.4, pp.341-374, Nov. 2003.
[6] P. Eugster and R. Guerraoui, "Probabilistic Multicast", Proceedings of the 2002 International Conference on Dependable Systems and Networks, pp.313-324, Vienna, Austria, June 2002.
[7] P. Eugster, "Atomic Probabilistic Broadcast", EPFL, IC_TECH_REPORT_200303.
[8] P. Felber, and F. Pedone, "Probabilistic Atomic Broadcast", in Proceedings of 21st IEEE Symposium on Reliable Distributed Systems (SRDS'02), Osaka, Japan, Oct. 2002, pp.170-179.
[9] D. Freedman, Ken. Birman, K. Ostrowski, M. Linderman, R. Hillman, and A. Frantz, "Enabling Tactical Edge Mashups with Live Objects", in Proceedings of the 15th International Command and Control Research and Technology Symposium(ICCRTS '10), Information Sharing and Collaboration Processes and Behaviors Track. Santa Monica, CA, USA, Jun. 2010.
[10] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion: A scalable and robust communication paradigm for sensor networks", In Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00), pp.56-67, Boston, MA, Aug, 2000.
[11] J. Lifton, M. Laibowitz, D. Harry, N.-W. Gong, M. Mittal, and J.A. Paradiso, "Metaphor and Manifestation—Cross-Reality with Ubiquitous Sensor/Actuator Networks", IEEE Pervasive Computing, Vol.8, No.3, pp.24-33, Jul.-Sep. 2009.
[12] A. Mostefaoui and M. Raynal, "Causal Multicasts in Overlapping Groups: Towards a Low Cost Approach", Proceedings of the 4th IEEE International Conference on Future Trends in Distributed Computing Systems, pp.136-142, 1993.
[13] C. Meesookho, S. Narayanan, and C. S. Raghavendra, "Collaborative Classification Applications in Sensor Networks", in Proceedings of Sensor Array and Multichannel Signal Processing Workshop, Rosslyn, USA, pp. 370-374, Aug. 2002.
[14] K. Ostrowski, K. Birman, and D. Dolev, "Live Distributed Objects: Enabling the Active Web", IEEE Internet Computing, Vol.11, Issue 6, pp.72-78, Nov.-Dec. 2007.
[15] K. Ostrowski, K. Birman, and D. Dolev, "QuickSilver Scalable Multicast", 7th IEEE International Symposium on Network Computing and Applications (IEEE NCA 2008). pp.9-18, Cambridge, July 2008.
[16] S. Pleisch and K. Birman, "SENSTRAC: Scalable Querying of SENSor Networks from Mobile Platforms Using TRACking-Style Queries", International Journal of Sensor Networks. Vol.3, Issue 4, pp.266-280, June 2008.
[17] L. Rodrigues, R. Baldoni, E. Anceaume, and M. Raynal, "Deadline-Constrained Causal Order," in Proceedings of Third IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2000, pp. 234-243.
[18] J. Yicka, , B. Mukherjeea, and D. Ghosal "Wireless sensor network survey", Computer Networks, Vol.52, Issue 12, pp.2292-2330, Aug. 2008.

CHAYOUNG KIM B.S. and M.S. degrees from the Sookmyung Women's University, Seoul, Korea, in 1996 and 1998, respectively and Ph.D. degree from the Korea University in 2006. From 2005 to 2008, she was a senior researcher in Korea Institute of Science and Technology Information, Korea, where she has been engaged in National e-Science of Supercomputing Center. Since 2009, she has been a researcher at Contents Convergence Software Research Center in Kyonggi University, Korea. Her research interests include distributed computing, group communications and peer-to-peer computing.

JINHO AHN(Corresponding author) received his B.S., M.S. and Ph.D. degrees in Computer Science and Engineering from Korea University, Korea, in 1997, 1999 and 2003, respectively. He has been an associate professor in Department of Computer Science, Kyonggi University. He has published more than 70 papers in refereed journals and conference proceedings and served as program or organizing committee member or session chair in several domestic/international conferences and editor-in-chief of journal of Korean Institute of Information Technology and editorial board member of journal of Korean Society for Internet Information. His research interests include distributed computing, fault-tolerance, sensor networks and mobile agent systems.

---

* Corresponding author: Tel.: +82 31 249-9674