A Waiting Time Determination Method to Merge Data on Distributed Sensor Data Stream Collection

Tomoya Kawakami[1(✉)], Tomoki Yoshihisa[2], and Yuuichi Teranishi[2,3]

[1] University of Fukui, Fukui, Japan
tomoya-k@u-fukui.ac.jp
[2] Osaka University, Ibaraki, Osaka, Japan
[3] National Institute of Information and Communications Technology, Koganei, Tokyo, Japan

Abstract. We define continuous sensor data with difference cycles as "sensor data streams" and have proposed methods to collect distributed sensor data streams. However, it is required to determine the appropriate waiting time in each processing computer (node) to collect and merge data efficiently. Therefore, this paper presents a method to determine a specific waiting time in each node. The simulation results show that the waiting time affects the loads and processing time to collect the data.

1 Introduction

In the Internet of Things (IoT), various devices (things) including sensors generate data and publish them via the Internet. We define continuous sensor data with difference cycles as a sensor data stream and have proposed methods to collect distributed sensor data streams as a topic-based pub/sub (TBPS) system [8]. In addition, we have also proposed a collection system considering phase differences to avoid concentrating the data collection to the specific time by the combination of collection cycles [4,5]. These previous methods are based on skip graphs [1], one of the construction techniques for overlay networks [3,6,7].

In our skip graph-based method considering phase differences, the collection time is balanced within each collection cycle by the phase differences, and the probabiity of load concentration to the specific time or processing computer (node) is decreased. However, it is required to determine the appropriate waiting time in each node to collect and merge data efficiently. Therefore, this paper presents a method to determine a specific waiting time in each node. The simulation results show that the waiting time affects the loads and processing time to collect the data.

L. Barolli et al. (Eds.): 3PGCIC 2020, LNNS 158, pp. 41–50, 2021.
https://doi.org/10.1007/978-3-030-61105-7_5

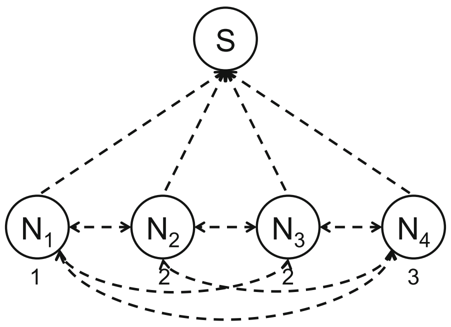


Fig. 1. An example of input setting.

2 Problems Addressed

2.1 Assumed Environment

The purpose of this study is to disperse the communication load in the sensor stream collections that have different collection cycles. The source nodes have sensors so as to gain sensor data periodically. The source nodes and collection node (sink node) of those sensor data construct P2P networks. The sink node searches source nodes and requires a sensor data stream with those collection cycles in the P2P network. Upon reception of the query from the sink node, the source node starts to delivery the sensor data stream via ohter nodes in the P2P network. The intermediate nodes relay the sensor data stream to the sink node based on their routing tables.

2.2 Input Setting

The source nodes are denoted as N_i $(i = 1, \cdots, n)$, and the sink node of sensor data is denoted as S. In addition, the collection cycle of N_i is denoted as C_i.

In Fig. 1, each node indicates source nodes or sink node, and the branches indicate collection paths for the sensor data streams. Concretely, they indicate communication links in an application layer. The branches are indicated by dotted lines because there is a possibility that the branches may not collect a sensor data stream depending on the collection method. The sink node S is at the top and the four source nodes N_1, $\cdots$, N_4 $(n = 4)$ are at the bottom. The figure in the vicinity of each source node indicates the collection cycle, and $C_1 = 1$, $C_2 = 2$, $C_3 = 2$, and $C_4 = 3$. This corresponds to the case where a live camera acquires an image once every second, and N_1 records the image once every second, N_2 and N_3 record the image once every two seconds, and N_4 records the image once every three seconds, for example. Table 1 shows the collection cycle of each source node and the sensor data to be received in the example in Fig. 1.

Table 1. An example of the sensor data collection.

Time	N_1 (Cycle: 1)	N_2 (Cycle: 2)	N_3 (Cycle: 2)	N_4 (Cycle: 3)
0	✓	✓	✓	✓
1	✓			
2	✓	✓	✓	
3	✓			✓
4	✓	✓	✓	
5	✓			
6	✓	✓	✓	✓
7	✓			
...	...	...	...	...

2.3 Definition of a Load

The communication load of the source nodes and sink node is given as the total of the load due to the reception of the sensor data stream and the load due to the transmission. The communication load due to the reception is referred to as the reception load, the reception load of N_i is I_i and the reception load of S is I_0. The communication load due to the transmission is referred to as the transmission load, the transmission load of N_i is O_i and the transmission load of S is O_0.

In many cases, the reception load and the transmission load are proportional to the number of sensor data pieces per unit hour of the sensor data stream to be sent and received. The number of pieces of sensor data per unit hour of the sensor data stream that is to be delivered by N_p to N_q ($q \neq p$; $p, q = 1, \cdots, n$) is $R(p, q)$, and the number delivered by S to N_q is $R(0, q)$.

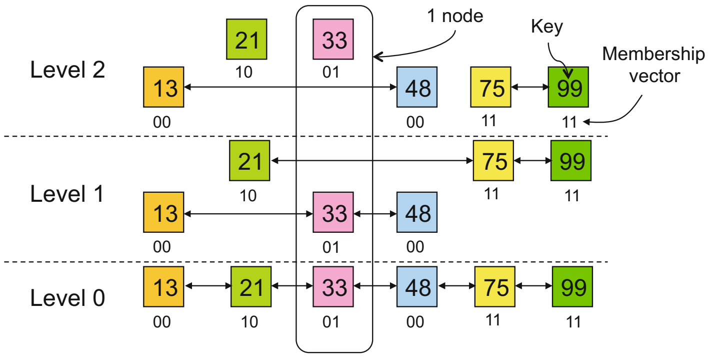


Fig. 2. A structure of a skip graph.

3 Proposed Method

3.1 Skip Graph-Based Collection Considering Phase Differences

Currently we have proposed a large-scale data collection schema for distributed TPBS [8]. [8] assumes the overlay network for the skip graph-based TBPS such as Banno et al [2]. Skip graphs are overlay networks that skip list are applied in the P2P model [1]. Figure 2 shows the structure of a skip graph. In Fig. 2, squares show entries of routing tables on peers (nodes), and the number inside each square shows a key of the peer. The peers are sorted in ascending order by those keys, and bidirectional links are created among the peers. The numbers below entries are called "membership vector." The membership vector is an integral value and assigned to each peer when the peer joins. Each peer creates links to other peers on the multiple levels based on the membership vector.

In [8], we employ "Collective Store and Forwarding," which stores and merges multiple small size messages into one large message along a multi-hop tree structure on the structured overlay for TBPS, taking into account the delivery time constraints. This makes it possible to reduce the overhead of network process even when a large number of sensor data is published asynchronously. In addition, we have proposed a collection system considering phase differences [4,5]. In the proposed method, the phase difference of the source node N_i is denoted as d_i $(0 \leq d_i < C_i)$. In this case, the collection time is represented to $C_i p + d_i$ $(p = 0, 1, 2, ...)$. Table 2 shows the time to collect data in the case of Fig. 1 where the collection cycle of each source node is 1, 2, or 3. By considering phase differences like Table 2, the collection time is balanced within each collection cycle, and the probabiity of load concentration to the specific time or node is decreased. Each node sends sensor data at the time base on his collection cycle and phase difference, and other nodes relay the sensor data to the sink node. In this paper, we call considering phase differences "phase shifting (PS)." Fig. 3 shows an exmple of the data forwarding paths on skip graphs with phase shifting (PS).

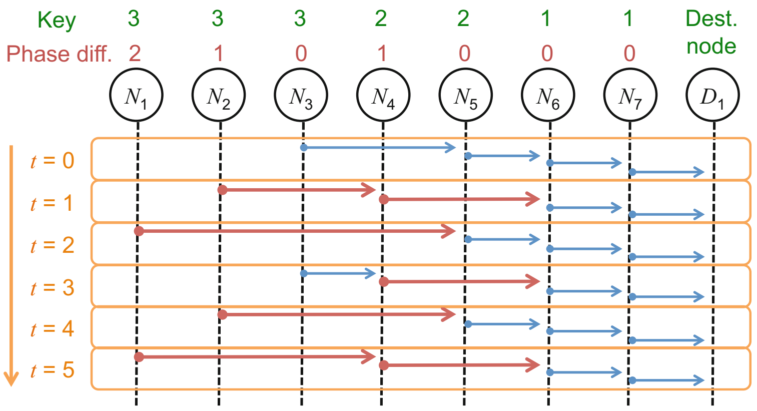


Fig. 3. Sensor data stream collection considering phase differences.

Table 2. An example of the collection time considering phase differences.

Cycle	Phase Diff	Collect. Time
1	0	0, 1, 2, 3, 4, ...
2	0	0, 2, 4, 6, 8, ...
	1	1, 3, 5, 7, 9, ...
3	0	0, 3, 6, 9, 12, ...
	1	1, 4, 7, 10, 13, ...
	2	2, 5, 8, 11, 14, ...

3.2 Determination of the Waiting Time

In the collection scheme shown in [4,5] and Fig. 3, more data are efficiently aggregated on the relay nodes for the destination node if the left side nodes which have longer cycles send data earlier to the next right side nodes. Hence, the possibility of data aggregation can be enhanced if the waiting time to send data is configured longer on the shorter cycle nodes. On the other hand, nodes are required the specific costs to understand indirectly linked nodes on autonomous decentralized overlay networks such as skip graphs. The costs depend on the scale of the overlay networks. Therefore, nodes in the proposed method configure their own waiting time based on the position on the key space. The position on the key space is estimated by their own collection cycles and phase differences. In the processes to determine the waiting time, this paper assumes that all nodes know the maximum waiting time denoted by w_{max}. The maximum waiting time is configured to the shortest cycle node located to the right edge on the key space. Each node configures its own waiting time based on the estimated position and the maximum waiting time to send data earlier than the nodes located right side.

Algorithm 1 shows the flow to determine the waiting time in the proposed method. From the line 1 to 6, the distance to the maximum waiting time node is calculated based on the node's collection cycle and phase difference. From the line 6 to 9, the maximum distance on the key space is calculated by the values of the selectable collection cycles. At the line 10, the relative position on the key space is calculated, and the node's waiting time is determined to shorten the waiting time for longer distance nodes.

Algorithm 1: Determination of the waiting time on each node

```
Input: $C$: list of selectable collection cycles, $c$: node's collection cycle, $d$: node's
       phase difference, $w_{\max}$: max. waiting time
Output: Node's waiting time
1  $p = 0$            // Node's location by its collection cycle and phase difference
2  for $i$ in $C$ do
3      if $c = C_i$ then
4          break
5      $p = p + C_i$
6  $p = p + d$
7  $p_{\max} = 0$  // Max. value by the selectable collection cycles and phase differences
8  for $i$ in $C$ do
9      $p_{\max} = p_{\max} + C_i$
10 return $w_{\max}(1 - p/p_{\max})$
```

4 Evaluation

In this paper, we evaluate the proposed method in simulation.

4.1 Simulation Environments

Table 3 shows the simulation environments. The collection cycle of each source node denoted by C_i is determined at random between 1 and 10. The simulation time denoted by t is from 0 to 2519, which length is the least common multiple of the selectable collection cycles. The number of source nodes is 250, 500, 750, or 1000. The data from the source nodes are forwarded to one destination node and aggregated on the relay nodes based on the configured waiting time. The average communication delay among nodes is 0.005×2^0, 0.005×2^1, 0.005×2^2, or 0.005×2^3. The communication delay on each node is determined under the normal distribution which variance σ^2 is 0.001. The maximum number of the aggregated streams on each node is 10 per time.

We execute the simulation for each environment and compare the results where the maximum waiting time is 0 (no waiting time), 0.5, 0.75, and 1.0. The default values of the number of nodes and the average communication delay are 500 and 0.01, respectively. The simulation is executed ten times for each environment, and the average values of the evaluation indices are calculated as simulation results. The evaluation indices are the maximum instantaneous load, the total loads, the average arrival delay from the source nodes to the destination node, and the maximum arrival delay.

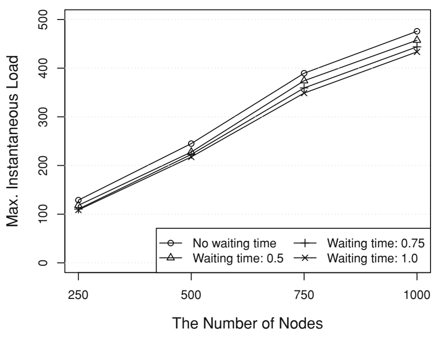
Table 3. Simulation environments.

Item	Value
Collection cycles	1, 2, ..., 10 (Determined at random)
The number of destination nodes	1
The number of source nodes	250, 500, 750, 1000
Avg. communic. delay	0.005, 0.01, 0.02, 0.04
Max. waiting time	0 (No waiting time), 0.5, 0.75, 1.0
Max. number of the aggregated streams	10
Simulation count	10
Evaluation indices	Max. instantaneous load, total loads, avg. arrival delay, max. arrival delay

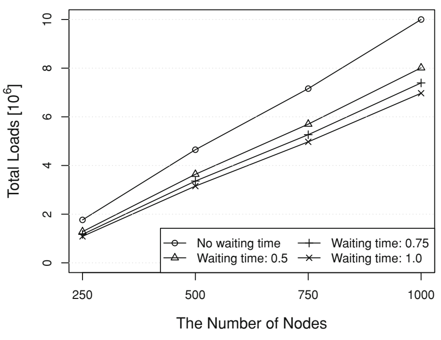
4.2 Results by the Number of Nodes

Figure 4 shows the maximum instantaneous load and the total loads of nodes when the number of nodes on the lateral axis is from 250 to 1000. The average communication delay is 0.01. In Fig. 4a, the maximum instantaneous load decreases by the maximum waiting time because longer waiting time increases the possibility of data aggregation. However, the falling rate is 9% compared to the case of no waiting time (the maximum waiting time is 0 and no data aggregation) even if the number of nodes is 1000 and the maximum waiting time is 1.0. In Fig. 4b, also the total loads decreases by the maximum waiting time, and the falling rate is higher than the result of the maximum instantaneous load. The maximum falling rate is nearly 30% compared to the case of no waiting time.

Figure 5 shows the average arrival delay and the maximum arrival delay when the number of nodes on the lateral axis is from 250 to 1000. The average communication delay is the same, 0.01. In Fig. 5a, the loads become lowest when the maximum waiting time is 1.0, however, the average arrival delay is over 1.0 and moves to the next time. On the other hand, the maximum of the increased amount is 0.1 in the case of no waiting time. The influence is not serious if the application allows a little increase of the arrival delay. In addition, the number of nodes does not have a large influence on the average arrival delay in this simulation environment because the proposed method uses skip graphs which can keep the number of hops nearly $\log n$. On the other hand, Fig. 5b shows that the maximum arrival delay is more affected by the number of nodes compared to the average arrival delay. The growth rate is decreased by the maximum waiting time. When the number of nodes is 1000 and no waiting time, the maximum arrival delay increases by 34% compared to the case of 250 nodes.

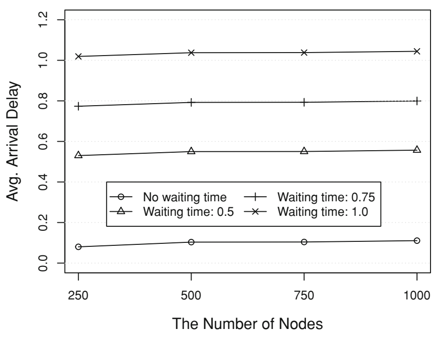


(a) The maximum instantaneous load

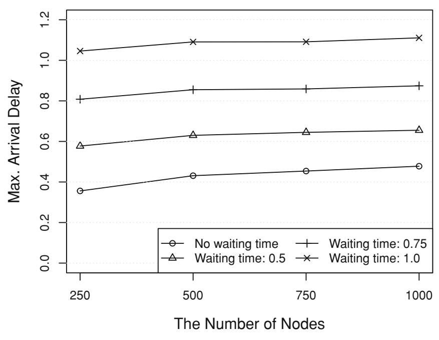


(b) The total loads

Fig. 4. Loads by the number of nodes.



(a) The average delay



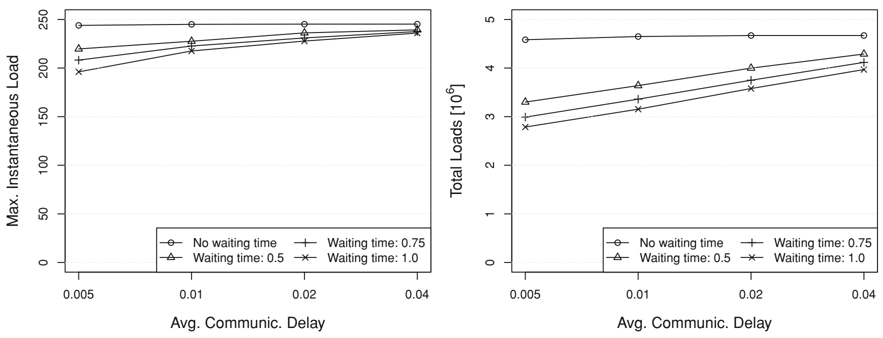
(b) The maximum delay

Fig. 5. Arrival delays by the number of nodes.

4.3 Results by the Average Communication Delay

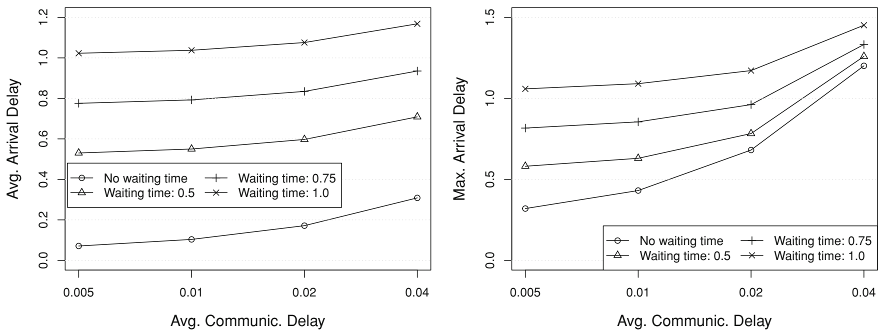
Figure 6 shows the maximum instantaneous load and the total loads of nodes when the average communication delay on the lateral axis is from 0.005 to 0.004. The number of nodes is 500. In Fig. 6a, the maximum instantaneous load increases by the average communication delay because longer communication delay decreases the possibility of data aggregation. In Fig. 6b, also the total loads increases by the average communication delay, and the growth rate is higher than the result of the maximum instantaneous load. When the average communication time is 0.04 and the maximum waiting time is 1.0, the total loads increase by 43% compared to the case where the average communication time is 0.005.

Figure 7 shows the average arrival delay and the maximum arrival delay when the average communication delay on the lateral axis is from 0.005 to 0.04. The number of nodes is the same, 500. Similar to the results by the number of nodes, the average arrival delay is over 1.0 and moves to the next time when the maximum waiting time is 1.0. In addition, the average communication delay affects



(a) The maximum instantaneous load (b) The total loads

Fig. 6. Loads by the average communication delay.



(a) The average delay (b) The maximum delay

Fig. 7. Arrival delays by the average communication delay.

the average arrival delay and the maximum arrival delay. When the average communication time is 0.04 and the maximum waiting time is 0 (no waiting time), the maximum arrival delay increases nearly by four times compared to the case where the average communication time is 0.005.

5 Conclusion

We have proposed a skip graph-based collection system for sensor data streams considering phase differences. In this paper, we proposed a method to determine a specific waiting time in each node. The simulation results show that the waiting time affects the loads and processing time to collect the data.

In future, we will evaluate the proposed method in various environments such as another distribution for the communication delays among nodes.

Acknowledgements. This work was supported by JSPS KAKENHI Grant Number 18K11316, G-7 Scholarship Foundation, and Research Grants from the University of Fukui.

References

1. Aspnes, J., Shah, G.: Skip graphs. ACM Trans. Algorithms **3**(4), 1–25 (2007)
2. Banno, R., Takeuchi, S., Takemoto, M., Kawano, T., Kambayashi, T., Matsuo, M.: Designing overlay networks for handling exhaust data in a distributed topic-based pub/sub architecture. J. Inform. Process. **23**(2), 105–116 (2015)
3. Duan, Z., Tian, C., Zhou, M., Wang, X., Zhang, N., Du, H., Wang, L.: Two-layer hybrid peer-to-peer networks. Peer-to-Peer Netw. Appl. **10**, 1304–1322 (2017)
4. Kawakami, T., Yoshihisa, T., Teranishi, Y.: A load distribution method for sensor data stream collection considering phase differences. In: Proceedings of the 9th International Workshop on Streaming Media Delivery and Management Systems (SMDMS 2018), pp. 357–367 (2018)
5. Kawakami, T., Yoshihisa, T., Teranishi, Y.: Evaluation of a distributed sensor data stream collection method considering phase differences. In: Proceedings of the 10th International Workshop on Streaming Media Delivery and Management Systems (SMDMS 2019), pp. 444–453 (2019)
6. Legtchenko, S., Monnet, S., Sens, P., Muller, G.: RelaxDHT: a churn-resilient replication strategy for peer-to-peer distributed hash-tables. ACM Trans. Auton. Adapt. Syst. **7**(2), 1–18 (2012)
7. Shao, X., Jibiki, M., Teranishi, Y., Nishinaga, N.: A virtual replica node-based flash crowds alleviation method for sensor overlay networks. J. Netw. Comput. Appl. **75**, 374–384 (2016)
8. Teranishi, Y., Kawakami, T., Ishi, Y., Yoshihisa, T.: A large-scale data collection scheme for distributed topic-based pub/sub. In: Proceedings of the 2017 International Conference on Computing, Networking and Communications (ICNC 2017) (2017)