KICS The Korean Institute of Communications and Information Sciences

Available online at www.sciencedirect.com

ScienceDirect

ICT Express 10 (2024) 895–901

www.elsevier.com/locate/icte

A simple and efficient Distributed Trigger Counting algorithm based on local thresholds

Seokhyun Kim[a], Giorgia Fattori[b], Yongsu Park[b,*]

[a] RobinTech, 6F V169, 602, Yeongdong-daero, Gangnam-gu, Seoul, 06083, Republic of Korea

[b] Department of Computer Science, Hanyang University, Wangshimriro 222, Seongdong-gu, Seoul 04763, Republic of Korea

Received 8 January 2024; received in revised form 18 April 2024; accepted 17 May 2024

Available online 24 May 2024

Abstract

Consider a large-scale distributed system in which each computing device is observing triggers from an external source. Distributed Trigger Counting (DTC) algorithm is used to detect the state of the system when the aggregated number of the observed triggers reaches a predefined value. In this paper, we propose a simple and efficient DTC algorithm: Cascading Thresholds (CT). We mathematically show that CT is an optimal DTC algorithm in terms of the total number of exchanged messages among the devices (*message complexity*). For the maximum number of received messages per device (*MaxRcv*), CT is sub-optimal. The average message complexity of CT is $O(N \log(W/N))$, and *MaxRcv* of it is $O(k \log(W/N) + N)$, where W is the number of triggers to be detected, N is the number of devices, and k is the degree of a node in the tree-like structure. Compared to the previous optimal algorithm (TreeFill), CT is much simpler: in our implementation the code size is about 2.5 times smaller. Also, unlike TreeFill CT does not require complicated mechanisms including distributed locking. Experimental results show that CT has a lower message complexity and *MaxRcv* compared to the previous work (CoinRand and RingRand). Furthermore, CT and TreeFill show a similar performance. From its simplicity, CT is more practical than previous work including TreeFill, CoinRand and RingRand.

Keywords: Internet-of-Things; Distributed counting; Sensor network; Distributed system

1. Introduction

Consider a large-scale networked system in which the participating devices are counting triggers from an external source. The Distributed Trigger Counting (DTC) algorithm is to raise an alarm when the total number of triggers counted by all the devices reaches a predefined value. DTC algorithms can be used for many real-world situations. E.g., effective monitoring is crucial to maintain sensor and data networks. Sensor and data networks are typically used to keep track of physical and environmental conditions such as traffic flow, animal behavior, military movements, and weather patterns. E.g., an alarm is triggered if the number of vehicles on a highway exceeds a certain limit, or if the population of a specific species surpasses another limit. In this way, DTC algorithms can be used for environment surveillance [1] with sensor networks [2–4] and also can be used for taking global snapshots [5].

We use the following notations throughout this paper: W denotes the total number of triggers to be detected, and N is the number of devices in the distributed system. We consider only the case where $W \gg N$, otherwise counting triggers becomes an easy problem [6,7]. To estimate the efficiency and scalability of our algorithm, we define *message complexity* to be the total number of exchanged messages among the devices. To show how message load is evenly distributed among the devices, we define *MaxRcv* to be the maximum number of received messages per device.

Garg et al. suggested a tree-based DTC algorithm and a centralized one in order to design distributed snapshot algorithms for large-scale distributed systems [5]. Their centralized algorithm has an optimal message complexity, but *MaxRcv* is higher than the optimal one. Chakaravarthy et al. suggested a sub-optimal algorithm, CoinRand, for both message complexity and *MaxRcv* using a tree-like network topology [6]. They also devised RingRand [6] which relies on the ring-topology

* Corresponding author.

E-mail addresses: sam@robintech.cloud (S. Kim), gioftt@hanyang.ac.kr (G. Fattori), yongsu@hanyang.ac.kr (Y. Park).

Peer review under responsibility of The Korean Institute of Communications and Information Sciences (KICS).

https://doi.org/10.1016/j.icte.2024.05.005

network. Kim et al. suggested TreeFill, an optimal DTC algorithm [8]. However, it uses a more complicated distributed communication protocol than [6]. Emek et al. improved lower bounds on DTC algorithms and proposed the probabilistic DTC algorithm, which significantly improve compared to the previous results [9]. Chang et al. also suggested the DTC algorithm that can work with any network topology [7]. Lee et al. proposed a simple DTC algorithm [10] for special cases but does not provide mathematical complexity analysis. Kim et al. proposed an efficient probabilistic DTC algorithm [11] but it has some false negatives. In 2021, Chang et al. proposed a new DTC algorithm [12] for arbitrary dynamic network topology without any global assumption. They did not analyze message complexity and experimentally showed that this algorithm has a high message complexity. Recently, efficient data-driven consensus control schemes have been proposed for the distributed event-triggered multi-agent systems (MAs) [13,14], where they guarantee the asymptotic consensus of MAs. The distributed consensus problem is similar to DTC but slightly different: it shares asymptotically common value globally using complex differential computation. In 2022, El-Hayek et al. solved the broadcasting problem for dynamic rooted trees in the linear time complexity under the presence of the adversary who hinders the protocol [15]. If we use this, the DTC problem may be solved under adversarial attacks.

In this paper, we present a simple and efficient DTC algorithm: Cascading Thresholds (CT). CT is inspired by [6] (that has sub-optimal message complexity) and uses a similar network topology compared to it. Our contributions are as follows.

- We prove that CT has no false negatives in the sense that when W events have occurred, CT does not fail to raise an alarm.
- We prove that CT is optimal in terms of average message complexity and sub-optimal in terms of average *MaxRcv*.
- Experimental results show that compared with CoinRand [6] and RingRand [6], CT has much lower message complexity and *MaxRcv*. Also, CT has similar performance to the optimal algorithm (TreeFill) [8] in terms of message complexity and *MaxRcv*.
- CT uses much simpler distributed communication protocol than the optimal DTC algorithm, TreeFill [8]. This is because (unlike CT) each node in TreeFill should maintain the state information of other nodes. Also, TreeFill requires distributed locking for serialized trigger processing. Due to its simplicity, we believe that CT is more practical than any other schemes.

This paper is organized as follows: Section 2 explains the proposed algorithm, CT. In Section 3, we provide mathematical analysis of CT. Experimental results are shown in Section 4. In Section 5, we conclude this paper.

2. Proposed DTC algorithm: Cascading Thresholds (CT)

Let W be the number of triggers to be detected and the total number of devices be $N = k^h$ where k is an even integer and $k \geq 4$. (Even though CT can be easily extended for general cases for $N \neq k^h$, we omit it due to a lack of space.) Devices are organized in a multi-leveled k-ary tree-like hierarchy, as follows. A total of N devices correspond to the N leaf nodes in Level-h. At level l $(0 \leq l < h)$, there are k^l nodes. Therefore, among N devices $k^0+k^1+\cdots+k^{h-1} = \frac{k^h-1}{k-1}$ devices have dual-roles and they are also associated with nodes in the Level-0 $\sim$ $(h-1)$. E.g., Fig. 1 shows an example of hierarchical structure when $k = 4$, $N = 16$, and $h = 2$. We define Level-h as the leaf level and all the other levels as aggregating levels.

The proposed algorithm operates on a round basis. Let $\hat{w}$ be the number of triggers that have not yet been detected at the start of each round. The initial value is $\hat{w} = W$. The detailed description of the algorithm is as follows.

1. **(Detect-message generation routine)** Recall that all N nodes are at the leaf level (level-h). In round i, each node has a local threshold $\tau_{leaf} = \lfloor \hat{w}/(2N) \rfloor$. Also, each node x $(1 \leq x \leq N)$ has a variable $C(x)$ to count the number of observed triggers in x at this round. When a node x detects a trigger, x increases $C(x)$ by 1. If x has observed τ_{leaf} triggers, $C(x)$ is cleared to 0, and x chooses a node y assigned to level-$(h-1)$ uniformly at random and sends a *Detect*-message to y.
2. **(Detect-message propagation routine)** At aggregating level l $(1 \leq l < h)$, there are k^l nodes. Each aggregating node y at Level-l maintains another counter variable $D(y)$ to indicate the number of *Detect*-messages received by the node y in the current round. Upon receiving a *Detect*-message, the node y increments $D(y)$ by 1. When the counter variable $D(y)$ reaches the threshold $\tau_{aggregating} = \lfloor k/2 \rfloor$, the node y clears $D(y)$ as 0 and sends a *Detect*-message to the randomly-selected node at the upper level $(= l-1)$. If we repeat this procedure, the root node eventually receives at least k *Detect*-messages, by Theorems 1 and 2, which are described in Section 3. If the root receives k messages, it goes to End-of-round procedure.
3. **(End-of-round procedure)** The root node computes the total number of received triggers by all nodes and updates $\hat{w}$ by using a simple broadcast and upcast procedure in the spanning tree, which is explained in the previous work [6,8]: The aggregation notification is broadcasted to all the nodes in a recursive top-down manner. Similarly, aggregation values are computed from the leaf nodes to the root node in a recursive bottom-up manner. Thus, the root node updates $\hat{w}$ and broadcasts it to all the nodes, again in a recursive fashion. Upon receiving the updated $\hat{w}$, all the nodes clear $C(x)$ and $D(x)$, and update τ_{leaf} for the next round.

   If the number of not yet detected triggers exceeds $2N$, a new round starts by going back to the Detect-message generation routine. Otherwise, it goes to the Final rounds procedure.
4. **(Final rounds procedure)** Every trigger makes a *Detect*-message. Instead of sending *Detect*-messages to level $h-1$, those are sent to the root. If the root counts all remaining triggers, it raises an alarm.

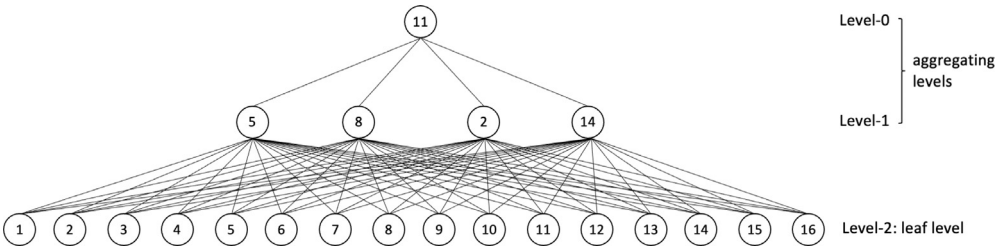

Fig. 1. An example of CT in Round 1 when $N = 16$, $k = 4$, and $h = 2$.

3. Analysis

First, we show CT has no false negatives in Section 3.1 and then analyze average message complexity and average *MaxRcv* in Section 3.2.

3.1. Mathematical analysis on CT

We first show 2 properties of CT in Theorems 1 and 2.

Theorem 1. *Regardless of the trigger distribution in each round, at the leaf level, always at least N* Detect*-messages are sent to the nodes assigned to level-$(h-1)$ before a total of $\hat{w}$ triggers occur.*

Proof. Let us calculate the maximum number of triggers that can occur when N *Detect*-messages are sent to the nodes assigned to level-$(h-1)$. The maximum number of triggers that each node can receive without generating a *Detect*-message is $\tau_{leaf} - 1 = \lfloor \hat{w}/(2N) \rfloor - 1$. Therefore, the maximum number of triggers that can occur when N *Detect*-messages are sent is less than $\hat{w}$ because: $\lfloor \hat{w}/(2N) \rfloor \cdot N + (\lfloor \hat{w}/(2N) \rfloor - 1) \cdot N < \hat{w}$. ■

Recall that when an aggregating node receives $\lfloor k/2 \rfloor$ *Detect*-messages, it creates a new *Detect*-message and sends it to a randomly chosen node at the upper level. If we repeat this procedure, the root node eventually receives at least k *Detect*-messages, by Theorem 2.

Theorem 2. *Recall that there are k^i nodes in aggregating level i $(0 < i < h)$. In level-i, always at least k^i* Detect*-messages are forwarded to the upper level before a total of k^{i+1}* Detect*-messages have been received from the lower level.*

Proof. Let us calculate the maximum number of received *Detect*-messages from the lower level when k^i *Detect*-messages are sent to the nodes assigned to the upper level. The maximum number of *Detect*-messages that each node can receive without generating a *Detect*-message is $\lfloor k/2 \rfloor - 1$. Therefore, the maximum number of *Detect*-messages received at Level-i when k^i *Detect*-messages are forwarded to the upper level is less than k^{i+1} because: $\lfloor k/2 \rfloor \cdot k^i + (\lfloor k/2 \rfloor - 1)k^i < k^{i+1}$. ■

By Theorem 1, regardless of the trigger distribution in each round, at the leaf level, always at least $k^h = N$ *Detect*-messages are sent to the nodes assigned to Level-$(h-1)$ before a total of $\hat{w}$ triggers occur. By Theorem 2, the root at Level-0 eventually receives k *Detect* messages before a total of $\hat{w}$ triggers occur. This means, CT always goes to the End-of-round procedure and the final rounds procedure. This also implies that CT has no false negatives.

3.2. Analysis on message complexity and MaxRcv

Average number of rounds: Consider the round i. Remember that when the root at level 0 receives k *Detect*-messages, it goes to the next round. At Level-1, there are k nodes. To send a *Detect*-messages from Level-1 to the root, the node at level 1 should receive at least $(1/2)k$ messages from Level-2. When it goes to the next round, the expected value of the variable $D(y)$ is $(1/2)k(1/2)$. This implies that to send a message from Level-1 to the root, each node at level 1 has received $(1/2)k + (1/2)k(1/2) = 3k/4$ messages from level-2 in average. Since there are k nodes at Level-1, they have received $3k^2/4$ *Detect*-messages from Level-2 in average.

Suppose that in Level-2, nodes have sent $3k^2/4$ *Detect*-messages. Since there are k^2 nodes in Level-2, in average each node have sent (3/4) *Detect*-message. To send a *Detect*-message, the node should have received at least $(1/2)k$ messages from Level-3. Hence, to send $(3k^2/4)$ messages, k^2 nodes have received $(3/4)k^2(k/2)$ messages from lower-level. When it goes to the next round, in average each node's $D(y)$ is $(1/2)k(1/2)$. Hence, in average k^2 nodes have received $(3/4)k^2(k/2) + (k/2)(1/2)k^2 = (5/8)k^3$ messages from Level-3.

For Level-3, to send $(5k^3/8)$ messages, in average k^3 nodes have received $(5/8)(k^3)(k/2) + (k/2)(1/2)k^3 = (9/16)k^4$ messages from Level-4.

If we generalize, for Level-j, in average k^j nodes have received $\frac{2^j+1}{2^{j+1}}k^{j+1}$ messages from Level-$j+1$. Therefore, for Level-h where h is the height, $(1/2 + 1/(2^h))k^h \approx (1/2)N$ *Detect*-messages have been generated in average. This implies that $(\hat{w}/2N)(1/2)N = \hat{w}/4$ triggers have occurred and the remaining trigger is $\hat{w} - \hat{w}/4 = 3\hat{w}/4$. In the final round, $\hat{w}_f = W(\frac{3}{4})^f < 2N$. So we can conclude that the number of rounds, $f = O\log(W/N)$.

Message complexity for each round: (From previous paragraphs,) for each round, the number of generated *Detect*-messages is $k + \frac{3}{4}k^2 + \frac{5}{8}k^3 + \cdots + (1/2 + \frac{1}{2}^h)k^h \leq k + k^2 + k^3 + \cdots + k^h = \frac{k^h-1}{k-1} = \frac{N-1}{k-1}$. Hence, The number of *Detect*-messages for each round is $O(N)$.

In the end-of-round, the root requests the number of triggers that have occurred so far to all nodes. If we assume that CT

uses the spanning tree, the number of messages for propagating the request is $O(N)$ and it gets the sum through the spanning tree again ($= O(N)$), and then this is shared by all nodes through the spanning tree: $O(N)$. Therefore, the number of messages occurring in each round is $O(N)$.

Message complexity: The average number of rounds is $\log(W/N)$, and the average number of messages per round is $O(N)$. In the final round, the number of remaining messages is less than $2N$. Hence, the message complexity is $O(N \log(W/N))$.

***MaxRcv* analysis:** Consider that in each round, an average number of *Detect*-messages is less than $\frac{N-1}{k-1}$. This means that in average, each node has sent/received far less than 1 message. Since the communication bottleneck is the root, we calculate *MaxRcv* for the root, which receives k messages for each round. In the end-of-round, if aggregation is performed through the $k-ary$ spanning tree, the root node sends $2k$ messages and receives k messages, and the internal node receives $k+1$ messages and sends $2k$ messages. Each leaf node receives 2 messages and sends 1 message. Therefore, in each round, *MaxRcv* is for the root and is $k+k+2$. The average number of rounds is $O(\log(W/N))$ and in the final round the root receives less than $2N$ messages, so *MaxRcv* $= O((2k+2)(\log(W/N))+2N) = O(k(\log(W/N))+N)$.

4. Simulation results

We wrote simulation code using NetLogo 6.4.0 [16] to compare the message complexity and *MaxRcv* of CT with other DTC algorithms (TreeFill, RingRand and CoinRand). The simulation code can be found at https://github.com/SeokhyunKim/dtc-algos. Fig. 2 shows the simulation screenshot of CT algorithm. By using the "DTC-ALGO" drop-down menu at the top left, we can select one of the DTC algorithms (currently CT). We can enter experimental parameters (N, k, W) in the input fields at the top left. After pressing the "Setup" button, we can press the "go" button to start the simulation. During simulation the number of detected triggers will be displayed in a graph at the bottom of Fig. 2. After the simulation ends, the number of rounds, message complexity, and *MaxRcv* are displayed in the center of Fig. 2.

We used the following parameters: the number of triggers to be detected (W) was 500,000. The number of nodes (N) was 4^i, where $2 \le i \le 5$. We set $k = 4$ for CT. We repeated experiments 20 times to get the average value of message complexity and *MaxRcv*.

Message complexity: Fig. 3 shows the message complexity of CT, TreeFill [8], RingRand [6] and CoinRand [6]. As shown in this figure, among them TreeFill has the smallest number of messages. While CT has slightly larger message complexity than TreeFill, it has much smaller message complexity than CoinRand and RingRand. E.g., when $N = 1024$, message complexity of CT is only about 1.19 times larger than that of TreeFill while that of CoinRand is 2.87 times bigger. As the number of nodes increases, the difference in message complexity also increases. Especially, RingRand shows the fastest increase.

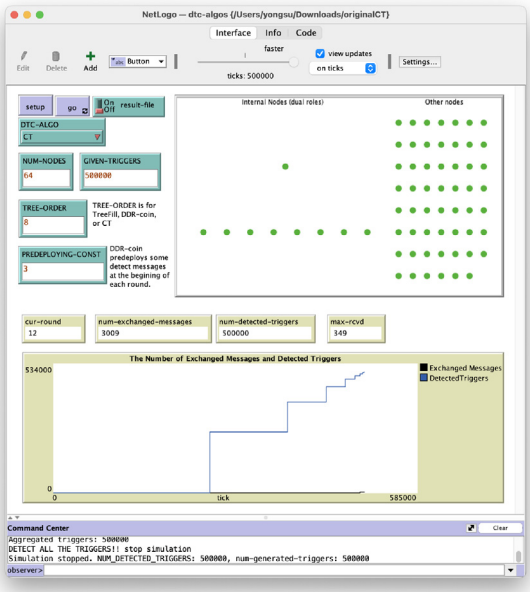

Fig. 2. Simulation of CT algorithm using NetLogo when $N = 64$, $k = 8$, $W = 500\,000$, and $h = 2$.

CT has similar message complexity with TreeFill until around $N \le 256$, but the difference increases for $N > 256$. Even though CT has a much simpler algorithm, message complexity of CT is not much bigger than TreeFill's one.

***MaxRcv*:** Fig. 4 shows the comparison of *MaxRcv* of CT, TreeFill, RingRand, and CoinRand. As shown in this graph, TreeFill and CoinRand show similar *MaxRcv*. *MaxRcv* of CT is lower than the one of CoinRand when N is small (about less than 20). However, the *MaxRcv* of CT goes bigger as N becomes larger, which can be explained by mathematical analysis of *MaxRcv* in Section 3, *MaxRcv* $= O(k(\log(W/N))+N)$. This is due to the final rounds procedure of CT, where all the messages go directly to the root and the *MaxRcv* of the root becomes $O(N)$. If we change the final rounds procedure as follows, *MaxRcv* of CT reduces signinifanctly and it is about the same as that of CoinRand or TreeFill, which is shown in 'enhanced CT' line of Fig. 4.

(Enhanced Final rounds procedure) Every trigger makes a *Detect*-message. Instead of sending *Detect*-messages to level $h-1$, those are sent to a level $l' = \max_l (w_i \ge 2k^l)$. The root node can start next round similarly after receiving k *Detect*-messages. In the next round, if $\hat{w}$ is less than a pre-defined small value e (e.g. $e = 200$), all the triggers go directly to the root. The root counts all remaining triggers and then raises an alarm. Rigorous mathematical analysis of this enhanced version is complicated so we will leave it as future work (simplified analysis is in the Appendix).

Comparison with the complexity analysis results: For CT, we compared the complexity analysis results and the simulation results. Our findings indicate that there are not so significant differences for both message complexity and *MaxRcv*, which are shown in Figs. 3 and 4 (CT-Analysis).

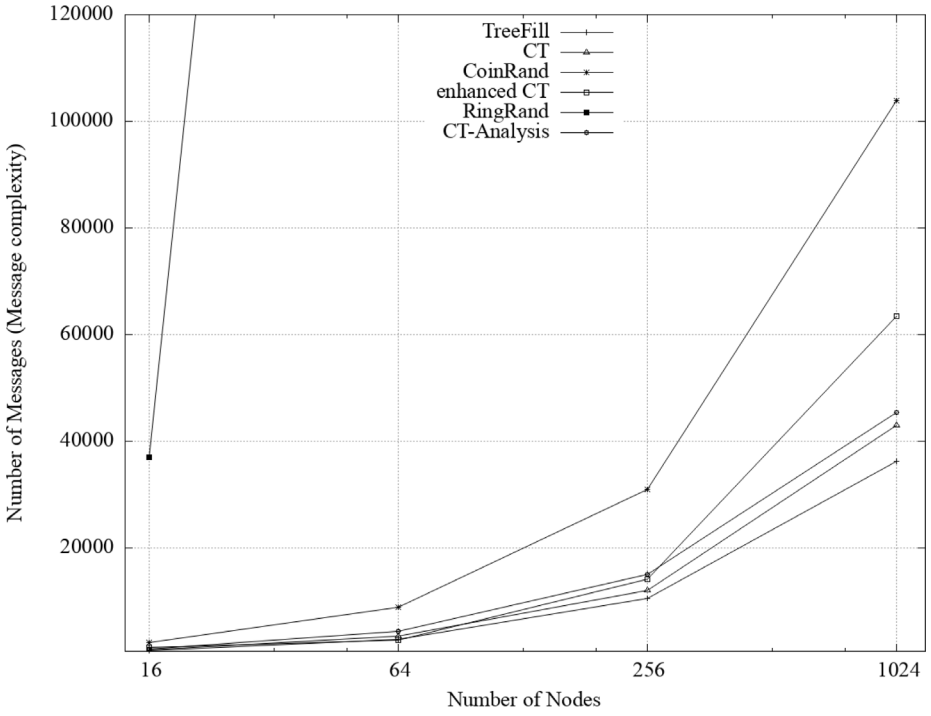

Fig. 3. Comparison of the numbers of messages of CT, enhanced CT, TreeFill, RingRand, and CoinRand when the number of nodes are 4^i where $2 \leq i \leq 5$. The number of triggers (W) is 500,000. $e = 200$ for the enhanced CT. CT-Analysis indicates the mathematical analysis results in Section 3.1.

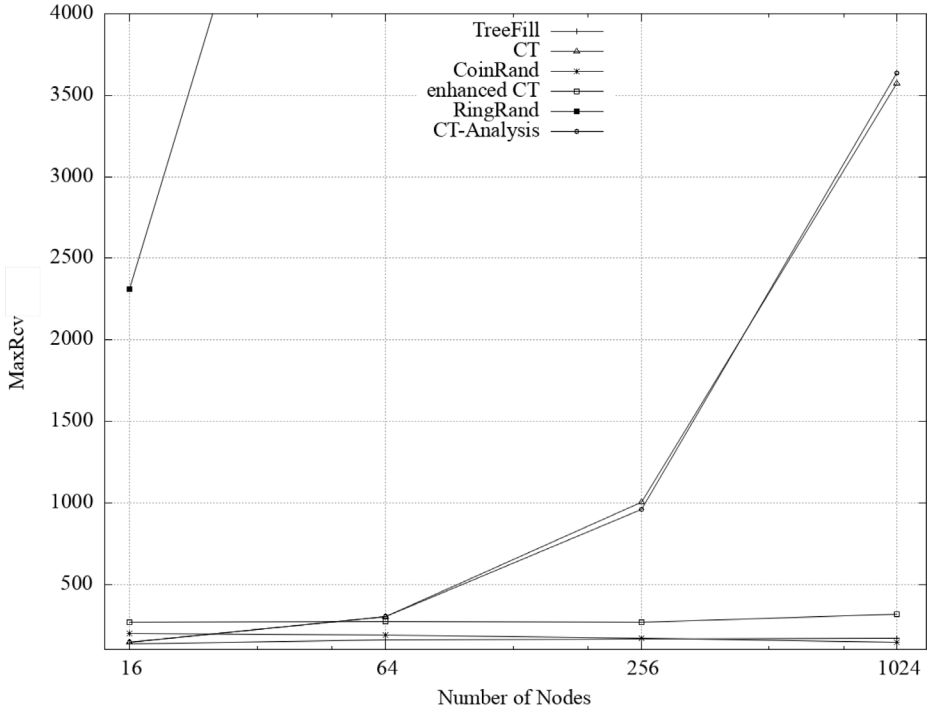

Fig. 4. Comparison of $MaxRcv$ of CT, enhanced CT, TreeFill, RingRand, and CoinRand, when the number of nodes is 4^i where $2 \leq i \leq 5$. The number of triggers (W) is 500,000. $e = 200$ for the enhanced CT. CT-Analysis indicates the mathematical analysis results in Section 3.2.

5. Conclusion

Distributed Trigger Counting (DTC) algorithm is used to detect the state of the system when the aggregated number of the observed triggers reaches a predefined value. In this paper, we propose a simple and efficient DTC algorithm: Cascading Thresholds (CT). We mathematically show that CT is an optimal DTC algorithm in terms of the total number of exchanged messages among the devices (*message complexity*). For the maximum number of received messages per node (*MaxRcv*), CT is sub-optimal. The average message complexity of CT is $O(N \log(W/N))$, and *MaxRcv* of it is $O(k \log(W/N) + N)$,

where W is the number of triggers to be detected, N is the number of devices, and k is the degree of a node in the tree-like structure. Experimental results show that CT has a lower message complexity and *MaxRcv* compared to the previous work (CoinRand and RingRand). Also, CT and TreeFill show a similar performance. Compared to previous work (TreeFill, CoinRand, RingRand), CT has a much simpler procedure: e.g., in our implementation, the code size is about 2.5 times smaller than TreeFill. Also, unlike TreeFill CT does not require complicated mechanisms including distributed locking. From this, we believe that CT is more practical than any other scheme.

CRediT authorship contribution statement

Seokhyun Kim: Conceptualization, Data curation, Formal analysis, Methodology, Software, Writing – original draft, Writing – review & editing. **Giorgia Fattori:** Formal analysis, Resources, Writing – review & editing. **Yongsu Park:** Data curation, Formal analysis, Methodology, Resources, Writing – original draft, Writing – review & editing, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by the National Research Foundation of Korea (NRF) grant, grant number RS-2023-00244071.

Appendix

Firstly, we analyze the message complexity and the *MaxRcv* for the enhanced CT (in short, eCT) as follows. Note that the final rounds of eCT can be considered as CT without final rounds with a smaller tree-like structure where $\hat{w} \leq 2N$. Hence, we use the analysis results in Section 3.2 to get the message complexity of final rounds: $\mathrm{O}(N \log(\hat{w}/N)) = O(N)$ and *MaxRcv* of final rounds: $O(k \log(\hat{w}/N) + N) = O(N)$, which implies that eCT has the same complexity as CT.

Since the resulting bounds are not so tight, we further analyze the number of rounds when $k = 4$. Then, we analyze the *MaxRcv* for eCT for $k = 4$. To do this, we will slightly simplify the problem by introducing some assumptions in the proof sketch.

Theorem 3. *In the eCT, when $k = 4$ and $N = 4^h$, the average number of rounds $= O(\log(W/N) + \log_4 N)$.*

Proof (*sketch*)**.** The eCT is identical to CT except for the final round(s), in which the number of rounds for non-final is $O(\log(W/N))$, as shown in Section 3.2. In the final rounds, $\hat{w} \leq 2N$.

From now on we analyze the number of rounds when $\hat{w} \leq 2N = 2 \cdot 4^h$. From Section 3.2, the average number of *Detect*-messages generated at Level-h is $(1/2)N = (1/2)4^h$. In eCT, every trigger makes a *Detect*-message, this implies that $(1/2)4^h$ triggers have occurred on average. Then, in each round, the remaining triggers, $\hat{w}$, are reduced by $(1/2)4^h$ on average until $\hat{w} < (1/2)4^h$. The number of rounds for this process is at most $(2N = 2 \cdot 4^h)/((1/2)4^h) = 4$ on average.

Now, the number of triggers left is $\hat{w} \leq (1/2)4^h = 2 \cdot 4^{h-1}$. The remaining problem is the exactly same as the case when in k-ary tree-like hierarchy we delete Level-$(h-1)$ while maintaining Levels-0 $\sim$ $(h-2)$ because for every trigger *Detect*-message goes to Level-$(h-2)$. If we use the same above analysis procedure, the number of rounds is 4 on average. Now, $\hat{w} \leq 2 \cdot 4^{h-2}$. (Precisely speaking, if $\hat{w}$ is small, we can delete more levels, which will reduce the number of rounds further. However, precise analysis is difficult so we will skip this part.)

If we repeat this procedure, in short, for every 4 rounds, we can delete one level in the structure. Since originally h levels exist, the maximal number of rounds is $4h = 4 \log_k N$. Hence, in eCT, when $k = 4$ and $N = 4^h$, the average number of rounds $= O(\log(W/N) + \log_4 N)$. ■

Recall that in Section 3.2 we already analyzed that the *MaxRcv* is $2k + 2$ for each round. By Theorem 3, the average number of rounds is $O(\log(W/N) + \log N)$. Hence, *MaxRcv* $= O((2k+2)(\log(W/N)+\log N)) = O(k(\log(W/N)+\log N))$. Note that this is not complete analysis because in the proof sketch of Theorem 3 we used some assumptions to simplify the problem. We will leave as future work the rigorous full proof and the cases in which $k \neq 4$.

References

[1] Aishwarya D., Minu R.I., Edge computing based surveillance framework for real time activity recognition, ICT Express 7 (2) (2021) 182–186.

[2] H.I. Kobo, A.M. Abu-Mahfouz, G.P. Hancke, A survey on software-defined wireless sensor networks: Challenges and design requirements, IEEE Access 5 (2017) 1872–1899.

[3] Cesar Pablos, Ángel G. Andrade, Guillermo Galaviz, Modulation-agnostic spectrum sensing based on anomaly detection for cognitive radio, ICT Express 9 (3) (2023) 398–402.

[4] Adiyasa Nurfalah, Suhono Harso Supangkat, Eueung Mulyana, Effective & near real-time track-to-track association for large sensor data in Maritime Tactical Data System, ICT Express (2023).

[5] Rahul Garg, Vijay K. Garg, Yogish Sabharwal, Efficient algorithms for global snapshots in large distributed systems, Parallel Distrib. Syst. IEEE Trans. 21 (5) (2010) 620–630.

[6] V. T. Chakaravarthy, A. R. Choudhury, Y. Sabharwal, Improved algorithms for the distributed trigger counting problem, in: Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International, 2011, pp. 515–523.

[7] Che-Cheng Chang, Jichiang Tsai, Distributed trigger counting algorithms for arbitrary network topology, Wirel. Commun. Mob. Comput. 16 (16) (2016) 2463–2476.

[8] Seokhyun Kim, Jaeheung Lee, Yongsu Park, Yookun Cho, An optimal distributed trigger counting algorithm for large-scale networked systems, SIMULATION: Trans. Soc. Model. Simul. Int. (2013).

[9] Y. Emek, A. Korman, Efficient threshold detection in a distributed environment: extended abstract, in: Proceedings of the 29th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, PODC '10, ACM, New York, NY, USA, 2010, pp. 183–191.

[10] Jaeheung Lee, Yongsu Park, A simple and efficient tree-based algorithm for the distributed trigger counting problem, Electronics 11 (7) (2022).

[11] Seokhyun Kim, Yongsu Park, DDR-coin: An efficient probabilistic distributed trigger counting algorithm, Sensors 20 (22) (2020).

[12] Che-Cheng Chang, Jichiang Tsai, Tien-Yu Chang, On the distributed trigger counting problem for dynamic networks, J. Internet Technol. 21 (4) (2021).

[13] Yifei Li, Xiangdong Liu, Haikuo Liu, Changkun Du, Pingli Lu, Distributed dynamic event-triggered consensus control for multi-agent systems under fixed and switching topologies, J. Franklin Inst. 358 (8) (2021) 4348–4372.

[14] Yifei Li, Xin Wang, Jian Sun, Gang Wang, Jie Chen, Data-driven consensus control of fully distributed event-triggered multi-agent systems, Sci. China Inf. Sci. 66 (5) (2023/02/15) 152202.

[15] Antoine El-Hayek, Monika Henzinger, Stefan Schmid, Brief announcement: Broadcasting time in dynamic rooted trees is linear, in: Proceedings of the 2022 ACM Symposium on Principles of Distributed Computing, PODC '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 54–56.

[16] Uri Wilensky, NetLogo, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 2023, http://ccl.northwestern.edu/netlogo/.