Performance Improvement of Tree Algorithm Using Adaptive Splitting Algorithms

Warakorn Srichavengsup[1*] Kanticha Kittipeerachon[2]

[1*,2]*Computer Engineering Robotics and Technology Research Laboratory : CERT, Faculty of Engineering, Thai-Nichi Institute of Technology, Bangkok, Thailand*

*Corresponding Author. E-mail address: warakorn@tni.ac.th

Received: 22 December 2021; Revised: 22 July 2022; Accepted: 14 September 2022
Published online: 22 December 2022

Abstract

In this paper, we propose the Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms that can be used in conjunction with the existing tree algorithms. binary tree and ternary tree algorithms divide users involved in a collision into a constant number of groups. Splitting users into a fixed number of groups without taking into account the number of collision-related users results in lower channel utilization. Therefore, the proposed algorithms are designed to improve the performance of tree algorithms by adjusting the number of groups to be split to match the number of users involved in the collision. It can be observed from the results that Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms perform better binary tree and ternary tree algorithms in terms of average delay, which indicates that the proposed algorithms can be used to enhance the efficiency of the tree algorithms. In particular, the Adaptive Splitting Type-2 algorithm offers the best performance.

Keywords: Adaptive splitting algorithms, Contention resolution, Tree algorithm

I. INTRODUCTION

In the era of high-speed communication, there is a quickly growing demand for large data transmissions. One problem in transmitting large and continuous data is a data collision. Data collisions occur when multiple users would like to transmit data at the same time. In order to alleviate the collision problem, several MAC protocols have been proposed. MAC protocols can be broadly classified into two types: contention-free MAC protocols and contention-based MAC protocols. In contention-free MAC protocols, a channel is allocated equally to each user. Therefore, there is no collision of data. Examples of contention-free MAC protocols include: Time Division Multiple Access (TDMA) [1], [2], Frequency Division Multiple Access (FDMA) [3], [4] and Code Division Multiple Access (CDMA) [5], [6]. One limitation of contention-free MAC protocols is in the event that the user has no data to send. A channel that is reserved for the user will be wasted because other users cannot use that channel. In contrast, contention-based MAC protocols are suitable for situations where there are many users in the system and each user may have different transmission requirements. Examples of contention-based MAC protocols are as follows: ALOHA protocol and its variants [7], [8], Carrier Sense Multiple Access (CSMA) protocols [9], [10] and tree-based algorithms [11]–[13]. Tree-based algorithms can support a large number of users and are able to resolve data collisions very well.

The principle of solving the data collision of tree-based algorithms can be described in detail as follows. When a collision occurs, the users involved in the collision are divided into subgroups. If another collision occurs in a subgroup, the users involved in that collision are divided again into subgroups. The collision-resolving process continues until all users successfully access the channel. The tree-based algorithm has a weakness in that the users involved in collisions are always split into a fixed number of subgroups regardless of the number of users involved in collisions. For example, in the case of two users involved in a collision, the appropriate number of subgroups is 2. Dividing the users into 3 or more subgroups will cause the idle slot and inefficient access to the channel. Therefore in this paper we introduce the Adaptive Splitting algorithms to enhance the performance of the well-known tree algorithm.

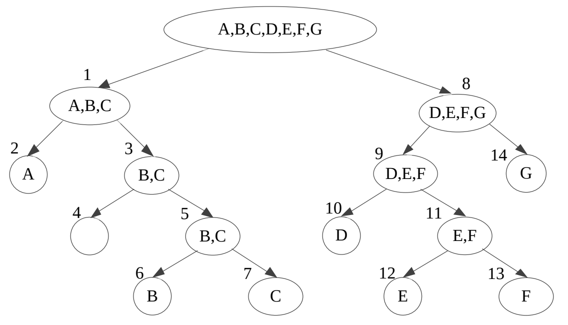
This paper is structured as follows: Section II, we describe the process of solving the collision of known tree algorithms. The Adaptive Splitting algorithms are explained in Section III. The results and discussion will be illustrated in Section IV and followed by a conclusion in Section V.

II. REVIEW OF EXISTING TREE-BASED ALGORITHMS

In this section, we will describe the collision resolution mechanisms of binary tree and ternary tree algorithms.

A. Binary Tree Algorithm

Binary tree algorithm was developed by Capetanakis in 1979 [14] and Tsybakov and Mikhailov in 1978 [15]. For the binary tree, when a collision occurs all relevant users are randomly split into two groups. Fig. 1 shows an example of the collision resolution mechanism of binary tree algorithm. From the figure, we can see that in the first slot, there are 3 users accessing the same slot, resulting in data collision. The users involved in the collision are divided into 2 groups. We found that another collision occurred in 3rd slot. The users involved in the collision are divided into 2 groups, and the collision resolution will continue until all users have completed access to the channel. We can see from the figure that the total number of slots used is 14 slots.

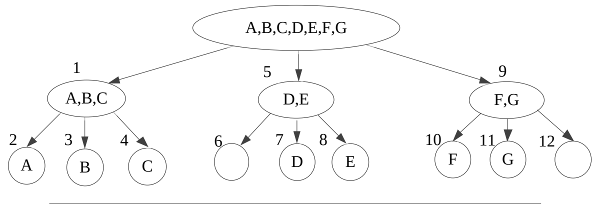


1	2	3	4	5	6	7	8	9	10	11	12	13	14
A,B,C	A	B,C		B,C	B	C	D,E,F,G	D,E,F	D	E,F	E	F	G

Figure 1: Collision resolution mechanism of binary tree algorithm.

B. Ternary Tree Algorithm

Binary tree algorithm has a significant degradation in performance when there are too many users in the system. This is because even if collision-related users are split into two groups, there is still a high chance of a collision in each group. In order to solve the collision problem in the event of a large number of users, an algorithm that divides the users into 3 groups is proposed. This algorithm is called ternary tree algorithm [16]. Fig. 2 displays an example of the collision resolution mechanism of ternary tree algorithm. The contention resolution mechanism of ternary tree algorithm is almost the same as the binary tree except that for every collision all involved users are split into new 3 groups. From the figure, the total number of slots used is 12 slots.



1	2	3	4	5	6	7	8	9	10	11	12
A,B,C	A	B	C	D,E		D	E	F,G	F	G	

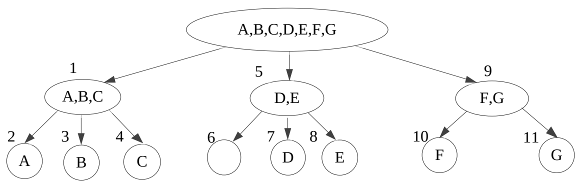
Figure 2: Collision resolution mechanism of ternary tree algorithm.

III. ADAPTIVE SPLITTING ALGORITHMS

In this section, we shall explain the collision resolution mechanisms of the proposed algorithms.

A. Adaptive Splitting Type-1 Algorithm

The Adaptive Splitting Type-1 algorithm has a collision resolution mechanism similar to the ternary tree algorithm. The difference is that when the number of remaining users in the system is equal to 2, the users involved in the collision are split into 2 groups instead of 3. For example, in 9th slot of Fig. 3, there is a collision between last 2 users. These 2 users are divided into 2 groups.



1	2	3	4	5	6	7	8	9	10	11
A,B,C	A	B	C	D,E		D	E	F,G	F	G

Figure 3: Collision resolution mechanism of Adaptive Splitting Type-1 algorithm.

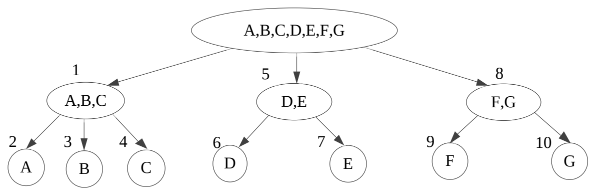
The pseudo code of Adaptive Splitting Type-1 algorithm is given in Algorithm 1.

Algorithm 1 : Adaptive Splitting Type-1 algorithm

```
while the number of remaining users in the system is more than 0
    if number of remaining users is equal to 2
        users involved in the collision are split into 2 groups else
        users involved in the collision are split into 3 groups
    subtract the number of successful users from the number of remaining users
end while
```

B. Adaptive Splitting Type-2 Algorithm

The Adaptive Splitting Type-2 differs from Adaptive Splitting Type-1 as follows: In the case of Adaptive Splitting Type-1, users are split into 2 groups only if the collision occurs between the last 2 users. While in all cases of collision between two users, Adaptive Splitting Type-2 always divides the user involved in the collision into 2 groups. For example, in 5th and 8th slots of Fig. 4, there are collision between 2 users. The users involved in the collision are divided into 2 groups.



1	2	3	4	5	6	7	8	9	10
A,B,C	A	B	C	D,E	D	E	F,G	F	G

Figure 4: Collision resolution mechanism of Adaptive Splitting Type-2 algorithm.

The pseudo code of Adaptive Splitting Type-2 algorithm is given in Algorithm 2.

Algorithm 2 : Adaptive Splitting Type-2 algorithm

```
while the number of remaining users in the system is more than 0
    if number of users involved in the collision is equal to 2
        users involved in the collision are split into 2 groups else
        users involved in the collision are split into 3 groups
    subtract the number of successful users from the number of remaining users
end while
```

IV. RESULTS AND DISCUSSION

In this section, we shall first investigate the performance of tree algorithms with different number of split groups under light loads. Let the variable Q represent the number of groups divided. As can be seen in Fig. 5, the average delay increases with the number of users in the system. This is because the more users in the system, the higher the chance of the collisions and it will take more time to resolve the collision. Furthermore, binary tree provides the best performance when the number of users in the system is less than or equal to 2. Whereas ternary tree gives the best results when the number of users in the system is greater than or equal to 3.

When increasing the number of split groups to 4 and 5, poor performance is observed. This is because splitting the users into too many groups will significantly increase the idle slots, resulting in lower performance.

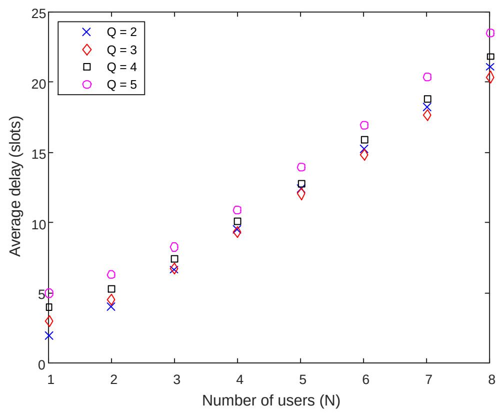


Figure 5: The average delay vs the number of users with varied number of split groups in case of light loads.

Fig. 6, demonstrates the performance comparison of tree algorithms with different number of split groups under heavy loads. It is found that the average delay tends to increase with an increase of the number of users. This decrease in performance is due to the increased number of collisions caused by a large number of user attempts to access the channel. It can be noticed that the result is similar to the case of light loads with the following details: Ternary tree gives the best results. Moreover, dividing users into 2 and 4 groups provide similar performance. While dividing users into 5 groups gives low performance.

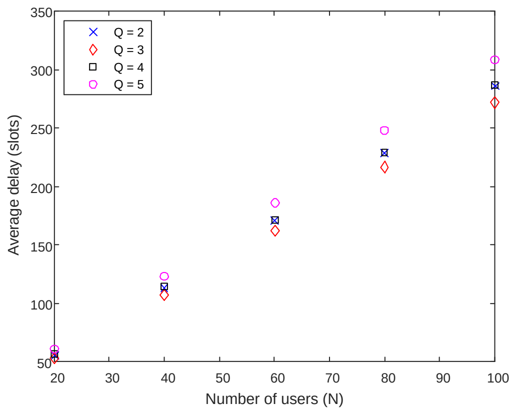


Figure 6: The average delay vs the number of users with varied number of split groups in case of heavy load loads.

Now we would like to turn our attention to investigate the effects of the number of split groups on the number of idle slots and the number of collision slots. Figs. 7 and 8 display the relation between the number of idle slots and the number of users with different number of split groups under light and heavy load respectively. As we can see, the number of idle slots increases with the number of split groups. This is because in the case of the same number of users, the greater the number of split groups, the more idle slots are likely to occur.

Figs. 9 and 10 illustrate the relation between the number of collision slots and the number of users under light and heavy load respectively. As we can see, the number of collision slots decreases with the number of split groups. This is because when the number of split groups is large, there is a low chance of the collision.

Considering the total number of collision and idle slots as shown in Figs. 11 and 12, it can be noticed that when the number of users is less than or equal to 2 splitting the users involved in a collision into 2 groups provides the best performance. However, when the number of users involved in a collision is greater than or equal to 3, splitting users involved in a collision into 3 groups offers superior performance. From the results, we can conclude that the number of split groups has a significant effect on the performance of the system. Therefore, the number of split groups should be carefully chosen to match the number of users involved in the collision.

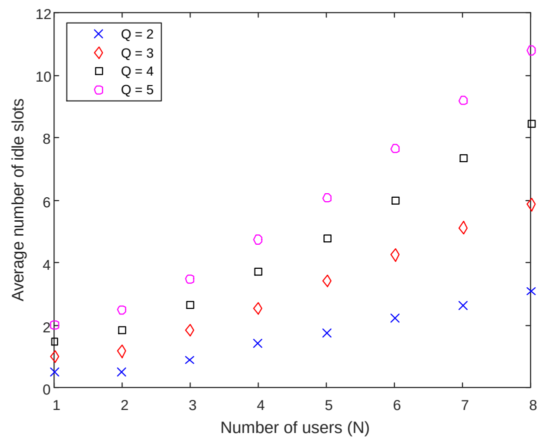


Figure 7: The average number of idle slots vs the number of users with varied number of split groups in case of light loads

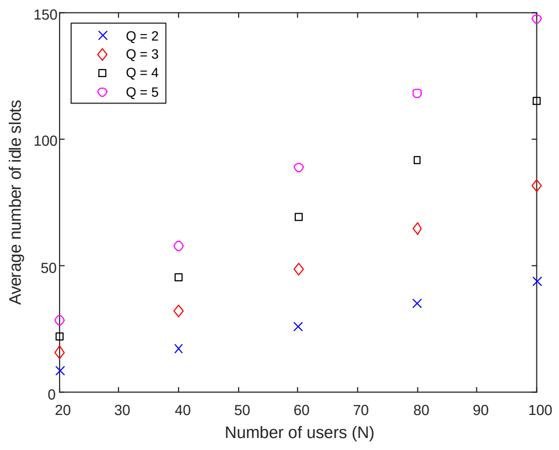


Figure 8: The average number of idle slots vs the number of users with varied number of split groups in case of heavy loads.

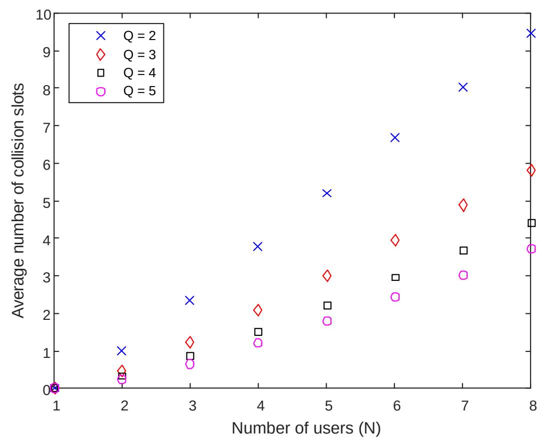


Figure 9: The average number of collision slots vs the number of users with varied number of split groups in case of light loads.

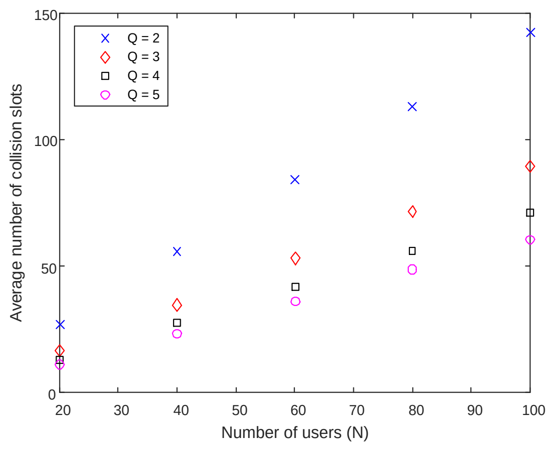


Figure 10: The average number of collision slots vs the number of users with varied number of split groups in case of heavy loads.

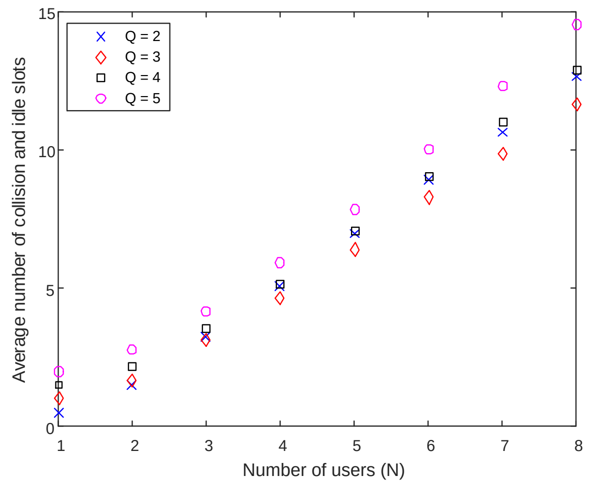


Figure 11: The average number of collision and idle slots vs the number of users with varied number of split groups in case of light loads.

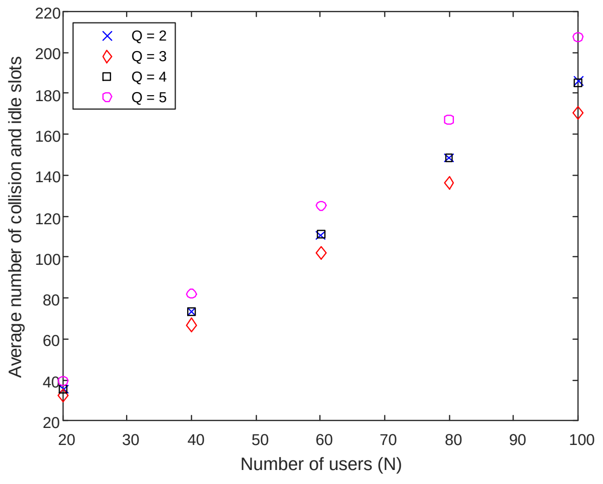


Figure 12: The average number of collision and idle slots vs the number of users with varied number of split groups in case of heavy loads.

Now we would to compare the performance of binary tree, ternary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms. For the Adaptive Splitting Type-1 algorithm, the users involved in a collision are divided into 2 groups when the number of remaining users in the system is 2. Whereas the users involved in a collision are divided into 3 groups when the number of remaining users in the system is greater than or equal to 3. For the Adaptive Splitting Type-2 algorithm, the users involved in a collision are divided into 2 groups if the collision occurs between 2 users. The users involved in a collision are divided into 3 groups when the collision occurs between 3 or more users.

Figs. 13 and 14 illustrate the performance comparison among binary tree, ternary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms under light and heavy loads respectively. It is revealed that when there is one or two users in the system, the binary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms have the same performance. This is because all three algorithms divide the number of users involved in the collision into 2 groups. Whereas a ternary tree which divides 2 collision-related users into 3 groups gives lower performance due to more idle slots. Moreover, when the number of users in the system increases, the Adaptive Splitting Type-2 algorithm offers the best performance. This is because it can effectively reduce idle slots with small increase in collision slots. While binary tree algorithm gives the lowest performance because even if a large number of users are divided into 2 groups, each group still has a large number of users, causing frequent collisions.

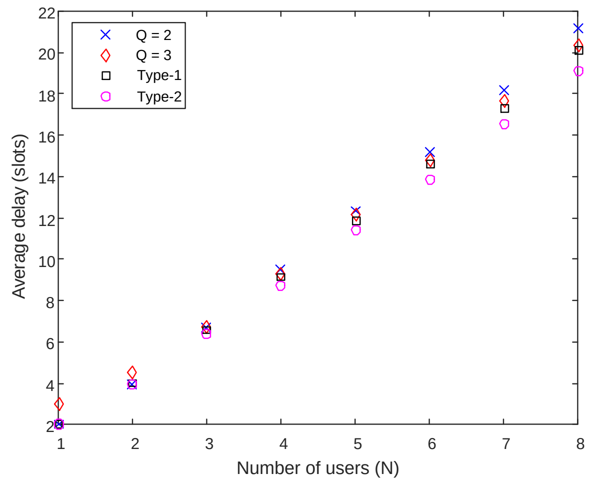


Figure 13: The average delay vs the number of users for binary tree, teranary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms in case of light loads.

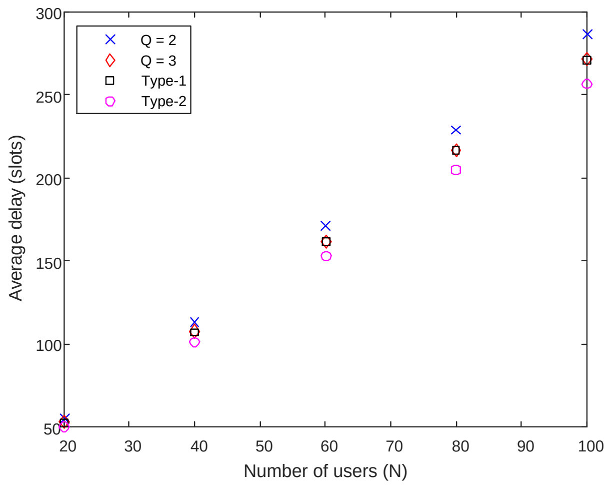


Figure 14: The average delay vs the number of users for binary tree, teranary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms in case of heavy loads.

Figs. 15 and 16, show the average number of collision slots for the system with light and heavy loads respectively. When comparing all algorithms, it is found that in the event that there are more than 2 users in the system, binary tree algorithm has the highest number of collisions. This is because the number of users in the system is relatively much higher than the number of split groups, so there is a high probability of collisions. Furthermore, it can be observed that the Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms have the higher average number of collision slots than ternary tree. This is because Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms divide users into 2 or 3 groups depending on the number of users involved in the collision. When users are divided into 2 groups, there is a greater chance of collision.

Figs. 17 and 18 demonstrate the relation between the average number of idle slots and the number of users under light and heavy loads. It is apparent that ternary tree algorithm has the highest number of idle slots. This is because Tree algorithm definitely divides the users into 3 groups while binary tree divides the users into 2 groups and Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms divide users into 2 or 3 groups. The larger the number of split groups, the greater the idle slots are also more likely to occur. In addition, it is found that the Adaptive Splitting Type-1 algorithm can help reduce idle slots slightly compared to ternary tree algorithm while the Adaptive Splitting Type-2 algorithm can greatly reduce idle slots.

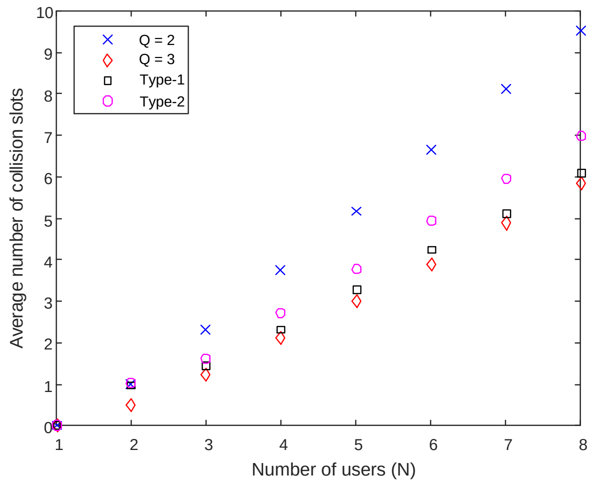


Figure 15: The average number of collision slots vs the number of users for binary tree, teranary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms in case of light loads.

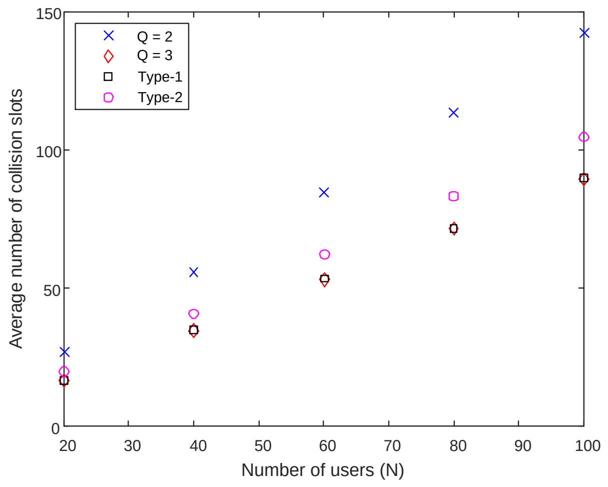


Figure 16: The average number of collision slots vs the number of users for binary tree, teranary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms in case of heavy loads.

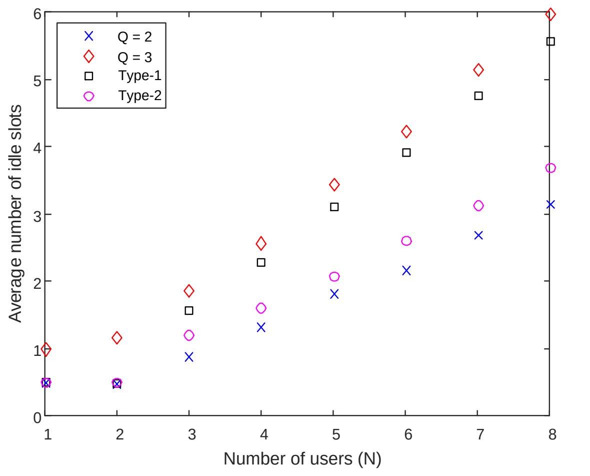


Figure 17: The average number of idle slots vs the number of users for binary tree, teranary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms in case of light loads.

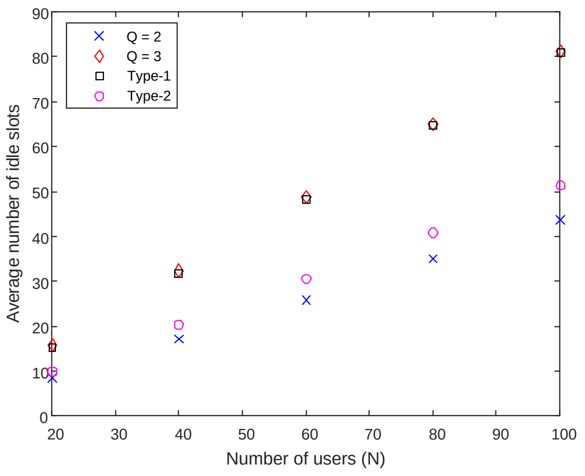


Figure 18: The average number of idle slots vs the number of users for Binary tree, teranary tree, adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms in case of heavy loads.

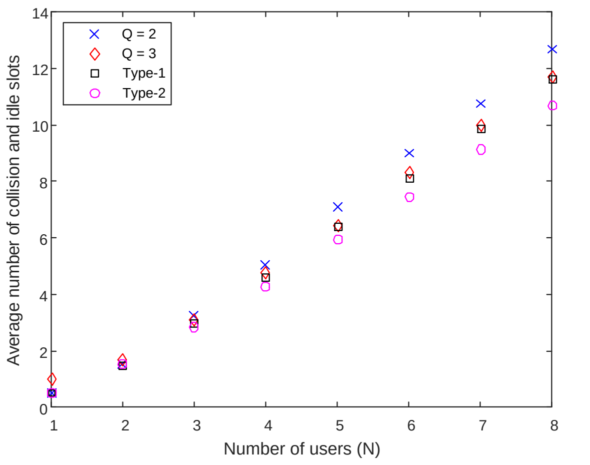


Figure 19: The average number of collision and idle slots vs the number of users for binary tree, teranary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms in case of light loads.

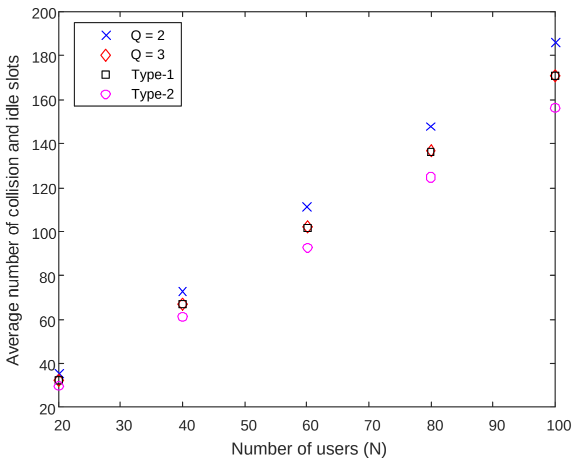


Figure 20: The average number of collision and idle slots vs the number of users for binary tree, teranary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms in case of heavy loads.

Figs. 19 and 20 illustrate the average number of collision and idle slots as a function of the number of users. These Figures reveal that Adaptive Splitting Type-2 has the best efficiency in reducing wasted slots and thus providing the superior system performance.

V. CONCLUSION

In this paper, we have proposed Adaptive Splitting algorithms to improve the performance of existing tree algorithms. From the results, it can be seen that the number of split groups is a significant factor that affects the performance of the system, hence must be chosen cautiously. When comparing among binary tree, ternary tree, Adaptive Splitting Type-1 and Adaptive Splitting Type-2 in terms of delay performance, we found that both Adaptive Splitting Type-1 and Adaptive Splitting Type-2 algorithms perform better than binary tree and ternary tree algorithms. In particular, the Adaptive Splitting Type-2 algorithm obviously outperforms other algorithms. Especially, in the case of heavy loads

Adaptive Splitting Type-2 algorithm gives a delay of 10–20% better than other algorithms. This is because it can adjust the number of split groups to suit the number of users involved in collisions. In case of collision between two users, the number of subgroups divided should be 2, but in case of collision between more than 2 users, the number of subgroups divided should be 3.

REFERENCES

[1] T. Zhang and Q. Zhu, "EVC-TDMA: An enhanced TDMA based cooperative MAC protocol for vehicular networks," *J. Commun. Netw.*, vol. 22, no. 4, pp. 316–325, Aug. 2020.

[2] J. Lee, H. Noh, and J. Lim, "TDMA-based cooperative MAC protocol for multi-hop relaying networks," *IEEE Commun. Lett.*, vol. 18, no. 3, pp. 435–438, Mar. 2014.

[3] J. Zhang, L. Yang, L. Hanzo, and H. Gharavi, "Advances in cooperative single-carrier FDMA communications: Beyond LTE-advanced," *IEEE Commun. Surv. Tut.*, vol. 17, no. 2, pp. 730–756, May 2015.

[4] M. Geles, A. Averbuch, O. Amrani, and D. Ezri, "Performance bounds for maximum likelihood detection of single carrier FDMA," *IEEE Trans. Commun.*, vol. 60, no. 7, pp. 1945–1952, Jul. 2012.

[5] B. Smida, S. Affes, K. Jamaoui, and P. Mermelstein, "A multicarrier CDMA space–time receiver with full-interference-suppression capabilities," *IEEE Trans. Veh. Technol.*, vol. 57, no. 1, pp. 363–379, Jan. 2008.

[6] X. Peng, K. Png, Z. Lei, F. Chin, and C. C. Ko, "Two-layer spreading CDMA: An improved method for broadband uplink transmission," *IEEE Trans. Veh. Technol.*, vol. 57, no. 6, pp. 3563–3577, Nov. 2008.

[7] H. Noh, J. Lee, and J. Lim, "ANC-ALOHA: Analog network coding ALOHA for satellite networks," *IEEE Commun. Lett.*, vol. 18, no. 6, pp. 957–960, Jun. 2014.

[8] H. Baek, J. Lim, and S. Oh, "Beacon-based slotted ALOHA for wireless networks with large propagation delay," *IEEE Commun. Lett.*, vol. 17, no. 11, pp. 2196–2199, Nov. 2013.

[9] J. Tong, L. Fu, and Z. Han, "Throughput enhancement of full-duplex CSMA networks using multiplayer bandits," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 11807–11821, Aug. 2021.

[10] A. Maatouk, M. Assaad, and A. Ephremides, "Energy efficient and throughput optimal CSMA scheme," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 316–329, Feb. 2019.

[11] W. Srichavengsup, K. Kittipeerachon, and C. Thongwan, "Improving the performance of IEEE 802.11 DCF with constant contention window by reducing the wasted time slots," *J. Eng. Digit. Technol. (JEDT),* vol. 8, no. 2, pp. 39–47, Dec. 2020.

[12] Y. C. Lai and L. Y. Hsiao, "General binary tree protocol for coping with the capture effect in RFID tag identification," *IEEE Commun. Lett.*, vol. 14, no. 3, pp. 208–210, Mar. 2010.

[13] A. J. E. M. Janssen and M. J. M. de Jong, "Analysis of contention tree algorithms," *IEEE Trans. Inform. Theory*, vol. 46, no. 6, pp. 2163–2172, Sep. 2000.

[14] J. I. Capetanakis, "Tree algorithms for packet broadcast channels," *IEEE Trans. Inform. Theory*, vol. 25, no. 5, pp. 505–515, Sep. 1979.

[15] B. S. Tsybakov and B. A. Mikhailov, "Random multiple packet access: Part-and-try algorithm," *Problems Inform. Transmiss.*, vol. 16, no. 4, pp. 65–79, Oct. 1980.

[16] P. Mathys and P. Flajolet, "Q-ary collision resolution algorithms in random-access systems with free or blocked channel access," *IEEE Trans. Inform. Theory*, vol. 31, no. 3, pp. 217–243, Mar. 1985.