Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 9(5): 13522–13536.
DOI:10.3934/math.2024659
Received: 22 February 2024
Revised: 26 March 2024
Accepted: 03 April 2024
Published: 12 April 2024

Research article

On the time complexity of achieving optimal throughput in time division multiple access communication networks

Samer Nofal*

Department of Computer Science, German Jordanian University, Amman, Jordan

* **Correspondence:** Email: samer.nofal@gju.edu.jo.

Abstract: The fundamental problem of finding transmission schedules for achieving optimal throughput in time division multiple access (TDMA) communication networks is known to be NP-hard. Let $\mathcal{N}$ be a scheduled k-time slot TDMA network with n stations and m links. We showed that an optimal link schedule for $\mathcal{N}$ can be computed recursively with a recursion tree of logarithmic depth $O(\ln m)$ in expectation. Additionally, we showed that optimal link schedules for those TDMA networks, with recursion trees of depth meeting the expectation, can be found in time $O(m^{2+\ln k})$. Likewise, we discuss analogous results for computing optimal station schedules of TDMA networks.

Keywords: time complexity; expected depth of recursion tree; exact scheduling; optimal throughput; time division multiple access; communication networks
Mathematics Subject Classification: 05C20, 05C85

1. Introduction

In time division multiple access (TDMA) communication networks, each communicating station uses the whole communication channel for a time slot. Therefore, in a TDMA network, time is divided into k equal time slots; data is transmitted in the time slots. Thus, the communication channel is sequentially time-shared among many stations through nonoverlapping time slots. TDMA networks have received significant attention over the years from researchers worldwide; see, e.g., the excellent reviews of [9, 29, 34].

We view a TDMA communication *network*, denoted by $\mathcal{N}$, as a pair $\mathcal{N} = (\mathcal{S}, \mathcal{T})$, where $\mathcal{S}$ is a set of n communicating *stations* and $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{S}$ is a set of m *transmission links*. For a transmission link $\varphi \in \mathcal{T}$, we denote by $s(\varphi)$ the sender station of φ and by $r(\varphi)$ the receiver station of φ. In TDMA, transmissions that do not *collide* are assigned the same time slot. Following [23], we specify the notion of *collision* in two different scenarios. Concerning the scenarios in which a station can transmit at most one message during a time slot, transmission collisions can be prevented if the following five constraints are all satisfied:

C.1 for all $(a, b) \in \mathcal{T}$, if (a, b) is assigned a time slot t, then for all $(b, c) \in \mathcal{T}$, (b, c) cannot be assigned t.

C.2 for all $(a, b) \in \mathcal{T}$, if (a, b) is assigned a time slot t, then for all $(c, b) \in \mathcal{T}$ with $c \neq a$, (c, b) cannot be assigned t.

C.3 for all $(a, b) \in \mathcal{T}$, if (a, b) is assigned a time slot t, then for all $(c, a) \in \mathcal{T}$, (c, a) cannot be assigned t.

C.4 for all $(a, b) \in \mathcal{T}$, if (a, b) is assigned a time slot t, then for all $(a, c) \in \mathcal{T}$ with $c \neq b$, (a, c) cannot be assigned t.

C.5 for all $(a, b) \in \mathcal{T}$, if (a, b) is assigned a time slot t, then for every $(c, d) \in \mathcal{T}$ with $c \neq a$ such that $(c, b) \in \mathcal{T}$, (c, d) cannot be assigned t.

In less formal words, the constraints C.1 and C.3 ensure that a station cannot send and receive messages simultaneously. C.2 states that a station cannot receive multiple messages simultaneously. C.4 indicates that a station cannot send multiple messages simultaneously. C.5 handles the following: When a station sends a message to another station, the sender station broadcasts to all linked stations, so we want to ensure that all linked stations are not busy receiving messages from other stations.

Regarding the scenarios in which a station must transmit all its messages simultaneously during a time slot, transmission collisions can be prevented if C.1–C.3 are satisfied with the following constraint: A station must send all its messages simultaneously.

C.6 for all $(a, b) \in \mathcal{T}$, if (a, b) is assigned a time slot t, then for all $(a, c) \in \mathcal{T}$, (a, c) should be assigned t.

A *link schedule*, $\Lambda : \mathcal{T} \to \mathbb{N}$, of a network $\mathcal{N}$ is a total mapping that assigns every transmission link in $\mathcal{T}$ a time slot $i \in \mathbb{N}$ such that all conditions C.1–C.5 are satisfied. A *station schedule*, $\Lambda' : \mathcal{S} \to \mathbb{N}$, of a network $\mathcal{N}$ is a total mapping that assigns every station in $\mathcal{S}$ a time slot $i \in \mathbb{N}$, such that all conditions C.1–C.3 and C.6 are satisfied. Assigning a time slot t to a station x implies that all links (x, y) are assigned to t. An *optimal* link (station, respectively) schedule of a network $\mathcal{N}$ is a link (station, respectively) schedule of $\mathcal{N}$ with a minimum number of time slots as to all link (station, respectively) schedules of $\mathcal{N}$. An optimal schedule for a network means optimal throughput is guaranteed.

It is known that the problem of finding an optimal link (or station) schedule is NP-hard; see, e.g., [4, 6]. Our contribution presented in this article can be summarized in the following points:

- We formulate an algorithm for computing exact optimal link schedules of TDMA networks.
- We show that for scheduled k-time slot TDMA networks, optimal link schedules can be computed recursively with a recursion tree of logarithmic depth $O(\ln m)$ in expectation.
- We show that optimal link schedules for those TDMA networks, with recursion trees of depth meeting the expectation, can be found in time $O(m^{2+\ln k})$.
- We formulate an algorithm for computing exact optimal station schedules of TDMA networks and give time complexity results.

In the remainder of this article, Section 2 overviews related work, and Section 3 describes an exact algorithm (Algorithm 1) for computing optimal link schedules for TDMA networks. In Section 4, we discuss the complexity of Algorithm 1. In Section 5, we emphasize the correctness of Algorithm 1 in several lemmas. In Section 6, we discuss analogous results for computing optimal station schedules of TDMA networks, and we conclude the paper in Section 7.

2. Related work

We overview recent TDMA protocol related work where the interested reader will find further references to earlier work. In [19], the authors discussed using TDMA protocol for secure mobile ad hoc networks. The article of [38] exploited TDMA protocol in opportunistic social networks. The work of [16] built an ultraviolet random access collaborative networking protocol based on TDMA protocol. In [39], the authors formulated an online distributed evolutionary optimization of TDMA protocols. The article of [32] studied a cooperative TDMA protocol with reconfigurable intelligent surfaces technology. The work of [31] proposed an opportunistic cooperative TDMA scheme where a source user transmits its data, and the cooperative users opportunistically transmit low-data-rate users' data. The article of [33] discussed an extensible frame structure for TDMA protocol in vehicular ad hoc networks. The work of [33] presented a design and implementation of a TDMA based protocol for sporting applications. In [5], the authors discussed a direct-sequence spread spectrum TDMA with direct detection for a latency-optimized passive optical network. The work of [27] assessed the stability of TDMA mesh networks through flooding route selection. In [37], a discriminative model is discussed for TDMA time slot prediction. The article of [40] studied a link distance division-based TDMA protocol for directional aeronautical relay networks. The work of [13] analyzed TDMA protocols for global mobile satellite communications. In [11], the authors discussed a priority-based TDMA algorithm for medium-range data communication in high-frequency radios. The article of [2] researched TDMA implementation on wireless sensor networks. The research of [22] examined a self-synchronizing TDMA network protocol. The article of [25] studied the backbone packet radio network coloring for time division multiple access link scheduling in wireless multi-hop networks. The work of [14] analyzed a design and performance of the asymmetric TDMA method for improving response time and throughput in tactical radio communication. The chapter of [3] discussed adaptive antennas for TDMA. The article of [15] examined a multichannel TDMA time slot scheduling with link recovery for multi-hop wireless sensor networks. The paper of [1] studied spatial TDMA for visible light communication networks. The work of [17] presented an improved TDMA protocol in underwater wireless sensor networks. The research of [7] investigated a human body communication-based biosensor transmitter for TDMA networks. Lastly, we note that exact and approximate algorithms solving different computational problems of communication networks are widely studied in the literature; see, for example, [8, 10, 12, 18, 20, 21, 24, 26, 28, 30, 35, 36].

3. Algorithm 1

We first give an informal overview of our algorithm for computing an optimal link schedule of a given TDMA network. Afterward, in this section, we will rigorously describe the processes of our algorithm. Thus, our algorithm is a backtracking, recursive procedure that searches for an optimal link schedule for a given TDMA network. For each link in the network, the algorithm assigns a time slot adhering to the collision avoidance conditions C.1 to C.5. A key feature of the algorithm is that once a link is assigned a time slot, the algorithm propagates any effects of this assignment to the concerned links due to any implications of the constraints C.1 to C.5. This propagation process entails further links to be assigned to specific time slots which in turn minimizes the number of links that are waiting to be assigned to a time slot avoiding any transmission collisions. Furthermore, at each link-time slot assignment, the algorithm analyzes the assignment deeply to foresee whether the assignment will eventually lead to a dead-end of the underlying search path; if a dead-end is detected, the algorithm backtracks and saves considerable time. To ensure we find an optimal link schedule (i.e., with a minimal number of time slots) for the given network, our algorithm will not increase the number of time slots unless increasing the number of time slots is the only way to prevent transmission collisions in the under-construction link schedule, provided that no link schedule is found thus far with a less number of time slots than the current number of time slots used in the under-construction link schedule.

Now, we describe our algorithm rigorously. We define the data structures we use in computing optimal link schedules. For a given network $\mathcal{N} = (\mathcal{S}, \mathcal{T})$, let $\lambda : \mathcal{T} \to \mathbb{N}_0$ be a total mapping to represent an under-construction link schedule. For any transmission link x, $\lambda(x) = 0$ indicates that x is not assigned a time slot yet. Let $\mu : \mathcal{T} \to 2^{\{1,...,m\}}$ be a total mapping to associate a transmission link with a set of *feasible* time slots: those that do not violate the conditions C.1–C.5 concerning the under-construction schedule λ. During the scheduling process, we use C to be the set of newly scheduled links whose effects on the unscheduled links are not checked yet, as we will elaborate in what follows. For a mapping $\Gamma : \mathcal{T} \to \mathbb{N}_0$, we define

$$\eta(\Gamma) = \begin{cases} \max\{t \mid \exists\varphi \in \mathcal{T} \text{ with } t = \Gamma(\varphi)\}, & \text{if } \Gamma \neq \emptyset; \\ m + 1, & \text{if } \Gamma = \emptyset \end{cases}$$

to be the maximum time slot assigned to some transmission link under Γ in case Γ is not empty; if $\Gamma = \emptyset$, then, $\eta(\Gamma)$ denotes $m + 1$.

Our exact procedure for finding an optimal link schedule for a given network $\mathcal{N} = (\mathcal{S}, \mathcal{T})$ is formalized in Algorithm 1. Next, we describe in detail the mechanism of the algorithm. To run Algorithm 1, invoke *lnk_sch*$(\mathcal{T}, C, \lambda, \mu)$ with $C = \emptyset$, and for all $\varphi \in \mathcal{T}$, $\mu(\varphi) = \{1, ..., m\}$, $\lambda(\varphi) = 0$. Thus, an optimal link schedule $\Lambda \neq \emptyset$ will be found at the end of the algorithm's execution. Observe that Λ is implicitly inputted as an empty set to the algorithm since it is not included in the procedure's input list (see the caption of Algorithm 1). This is intentionally the case because the algorithm maintains only one copy of Λ, which will contain an optimal link schedule at the end of the algorithm's execution. One may implement Λ as a global variable in computer programming.

Now, we elaborate on the actions of Algorithm 1. As mentioned above, we initially invoke the algorithm with $C = \emptyset$. So, let us skip, for now, the *while* loop at line 1 of Algorithm 1 and move to line 16 in the algorithm, where the algorithm checks if $\mathcal{T}$ is empty. The set $\mathcal{T}$ being empty indicates that all transmission links have been scheduled. Hence, the algorithm concludes that the under-construction schedule λ is now completed and that it is the best link schedule so far constructed, and, subsequently, the algorithm saves λ in Λ and immediately returns to the caller of the procedure *lnk_sch*; see line 16. In line 17, the algorithm selects an unscheduled link $\varphi \in \mathcal{T}$ to assign it a time slot. Referring to lines 18–19, the algorithm tries every feasible time slot $\tau \in \mu(\varphi)$ satisfying that (i) $\tau < \eta(\Lambda)$, and (ii) $\tau \leq \eta(\lambda) + 1$. The case of (i) is obvious because we search for an optimal schedule with the least number of time slots. Hence, there is no need to look for schedules with several time slots equal to or greater than the number of time slots of the so-far-best (nonempty) schedule Λ. Recall, if $\Lambda = \emptyset$, then $\eta(\Lambda) = m+1$. We turn now to the second condition (ii). According to (ii), in a first call to the procedure *lnk_sch*, referring to lines 17–19, we might try one time slot for a first extracted transmission link. For a second extracted transmission link (in a second call to *lnk_sch*), we might try *up to* two time slots. We might try *up to* three time slots for a third extracted transmission link, and so forth. To see the objective from the second condition (ii), take, for instance, a network of three links x, y, z. Referring to line 17, suppose that x is first extracted, then y, and finally z. In this sequence, assign these links to the time slots $(1, 1, 2)$, respectively, then, all assignments in the form $(1, 1, t)$, with $t > 2$, are similar to $(1, 1, 2)$, so there is no need to explore such assignments (i.e., $(1, 1, t)$ with $t > 2$) at all. This is because an assignment in the form $(1, 1, t)$ (with $t \geq 2$) implies that x and y are assigned the time slot 1, while z is assigned a different time slot (i.e., other than 1). In other words, we cannot let the network be idle in the second time slot if we assign x, y, and z (respectively), the time slots $(1, 1, 3)$, for instance.

Being recursive, the algorithm calls itself in line 19 with four parameters: the current unscheduled transmission links $\mathcal{T}$, the newly-scheduled link $\{\varphi\}$, the current under-construction link schedule λ, and the set, μ, that includes the feasible time slots for every link in $\mathcal{T}$. By invoking the procedure *lnk_sch* in line 19, we apply the algorithm again starting from the *while* loop. The set C contains newly-scheduled links. So, the job of the *while* loop is to update the feasible time slots for those unscheduled links in $\mathcal{T}$ that might be affected by the newly assigned time slots for the links in C. In line 3 (within the while loop), the algorithm checks those (unscheduled yet) links α that have their sender or receiver station being the same as the sender or receiver of a link $\beta \in C$. This is to enforce the conditions C.1–C.5. Starting with constraint C.5, the algorithm (lines 4–5) scans those links γ with a sender station $s(\gamma)$ being the same as the sender station of α, whose receiver station is the same as the receiver of the link β. Each of such links, γ, is forbidden to be assigned the time slot assigned to β; thus, in line 6, the algorithm updates the set of feasible time slots for γ accordingly by removing the time slot assigned to β. In line 7, the algorithm examines if the set of feasible time slots of γ is empty. If so, the algorithm returns to the caller of the *lnk_sch* procedure (i.e., the previous instance of the procedure itself). It is pointless to proceed with the search process whenever a link cannot be scheduled without violating condition C.5.

In line 8, the algorithm checks if the set of feasible time slots of γ contains one time slot only. If so, we must assign γ the only possible time slot (see line 9). Now, as noted earlier, the strategy taken by the algorithm is that every time a link (in this case γ) is assigned a time slot, the algorithm adds γ to C (line 10) to impose subsequently any arising implications of this new assignment on the feasible time slots of the unscheduled links in $\mathcal{T}$. This is because we have to maintain the conditions C.1–C.5 being satisfied; note that we already explained how the algorithm maintains C.5. Shortly, we illustrate how the algorithm keeps the conditions C.1–C.4. Referring to line 9, once the algorithm assigned a time slot to γ, the algorithm (line 11) removes γ from the set $\mathcal{T}$ to indicate that γ has been scheduled, and so to let the algorithm look for scheduling the remaining links in $\mathcal{T}$ in the following search stages.

To enforce the conditions C.1–C.4, the algorithm applies lines 12–15. Recall, for a link α that has its sender or receiver station being the sender or receiver of a recently-scheduled link β (see lines 2–3), the algorithm (line 12) removes the time slot $\lambda(\beta)$ from $\mu(\alpha)$, which is the set of feasible time slots of α. In line 13, the algorithm confirms if the link α has no feasible time slots left, and if so, the algorithm returns to the caller of the procedure *lnk_sch*, which is the previous instance of the procedure itself. In line 14, the algorithm examines if $|\mu(\alpha)|$, the size of the set of feasible time slots of α, equals 1. If so, the algorithm concludes that there are no choices other than assigning to α the only feasible time slot possible for α. In line 15, after assigning a time slot to α, α is included in the set of newly-scheduled links C and then is removed from $\mathcal{T}$. Afterward, the *while* loop continues until the algorithm finishes verifying the impact of the recently-scheduled links of C on the remaining unscheduled links of $\mathcal{T}$. Therefore, the *while* loop stops when C is empty. After the *while* loop, the search process continues if $\mathcal{T}$ is not empty (line 16) in the same manner we described above.

Algorithm 1: *lnk_sch*($\mathcal{T}, C, \lambda, \mu$)

```
1  while $C \neq \emptyset$ do
2      extract a link $\beta$ from $C$;
3      foreach $\alpha \in \mathcal{T}$ with $\{s(\alpha), r(\alpha)\} \cap \{s(\beta), r(\beta)\} \neq \emptyset$ do
4          if $r(\beta) = r(\alpha)$ then
5              foreach $\gamma \in \mathcal{T}$ with $s(\gamma) = s(\alpha)$ do
6                  $\mu(\gamma) \leftarrow \mu(\gamma) \setminus \{\lambda(\beta)\}$;
7                  if $\mu(\gamma) = \emptyset$ then return;
8                  if $|\mu(\gamma)| = 1$ then
9                      $\lambda(\gamma) \leftarrow \tau$ where $\tau \in \mu(\gamma)$;
10                     $C \leftarrow C \cup \{\gamma\}$;
11                     $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\gamma\}$;
12         $\mu(\alpha) \leftarrow \mu(\alpha) \setminus \{\lambda(\beta)\}$;
13         if $\mu(\alpha) = \emptyset$ then return;
14         if $|\mu(\alpha)| = 1$ then
15             $\lambda(\alpha) \leftarrow \tau$ where $\tau \in \mu(\alpha)$; $C \leftarrow C \cup \{\alpha\}$; $\mathcal{T} \leftarrow \mathcal{T} \setminus \{\alpha\}$;
16 if $\mathcal{T} = \emptyset$ then $\Lambda \leftarrow \lambda$; return;
17 extract a randomly selected link $\varphi$ from $\mathcal{T}$;
18 foreach $\tau = 1 .. m : \tau \in \mu(\varphi)$ and $\tau < \eta(\Lambda)$ and $\tau \leq \eta(\lambda) + 1$ do
19     $\lambda(\varphi) \leftarrow \tau$; lnk_sch($\mathcal{T}, \{\varphi\}, \lambda, \mu$);
```

4. Complexity of Algorithm 1

The space requirement of Algorithm 1 is restricted by the structures employed by the procedure *lnk_sch*. These structures are $\mathcal{T}$, C, μ, and λ. Note the size of μ is in $O(m^2)$, while the size of $\mathcal{T}$, C, or λ is in $O(m)$. It is now obvious that the algorithm runs in $O(m^3)$ space since, in the worst-case scenario, m instances of the recursive procedure *lnk_sch* might be active simultaneously where every instance holds its version of the set μ that includes the feasible time slots for all links in the inputted network.

Now, we discuss the computer representation of the set $\mathcal{T}$. Observe that we can implement $\mathcal{T}$ as a boolean array Ψ such that $\Psi[x] = true$ if, and only if, $x \in \mathcal{T}$. Thus, extracting a link x from $\mathcal{T}$ requires updating $\Psi[x] \leftarrow false$. Nonetheless, before applying line 17 in Algorithm 1 (i.e., after the *while* loop), we need to resize Ψ by removing all entries with $false$ to reflect the remaining unscheduled links. To this end, for every link in $\mathcal{T}$, we must keep track of its position in the array Ψ. Thereby, for every link x in the initially inputted network, we need to have another array $\mathcal{I}$ to hold the position of x in Ψ. Thus, updating Ψ (by removing all entries with $false$) requires updating the corresponding entries of $\mathcal{I}$. To see these arrays in action, take $\mathcal{T} = \{a, b, c, d\}$, then, the corresponding implementation of $\mathcal{T}$ is an array $\Psi = ((a, true), (b, true), (c, true), (d, true))$ along with another array $\mathcal{I} = (0, 1, 2, 3)$, such that the entries of $\mathcal{I}$ represent the position of a, b, c, and d, respectively, in the array Ψ. Assume we remove b from $\mathcal{T}$. This is performed by $\Psi[\mathcal{I}[b]] \leftarrow (b, false)$ where $\mathcal{I}[b] = 1$. Likewise, removing d from $\mathcal{T}$ means setting $\Psi[\mathcal{I}[d]] \leftarrow (d, false)$. Now, after we finish executing the *while* loop and before extracting a link at line 17 in Algorithm 1, we need to shrink Ψ such that it contains only the entries with *true*, which requires updating $\mathcal{I}$ as well. Thus, the new structures become $\Psi = ((a, true), (c, true))$ and $\mathcal{I} = (0, undefined, 1, undefined)$, which correspond to $\mathcal{T} = \{a, c\}$. Therefore, in summary, whenever we extract a link x from $\mathcal{T}$ (see lines 11 & 15 & 17), we update $\Psi[\mathcal{I}[x]] \leftarrow false$ in constant time. Hence, we differentiate between the cost of extracting a link from $\mathcal{T}$, which can be done in constant time, and the cost of shrinking $\mathcal{T}$, which can be done in linear time $O(m)$. Again, observe that we do not shrink $\mathcal{T}$ every time a link is removed from $\mathcal{T}$. We delay shrinking $\mathcal{T}$ until just before applying line 17 in Algorithm 1, where we extract an unscheduled link from $\mathcal{T}$ to assign it a time slot.

The following two propositions discuss the running time of Algorithm 1.

Proposition 1. *Let $\mathcal{N} = (\mathcal{S}, \mathcal{T})$ be a scheduled k-time slot network with n stations and m links. The recursion tree of Algorithm 1 (running on $\mathcal{N}$) has depth $O(\ln m)$ in expectation.*

Proof: We note that the expected depth of the recursion tree of Algorithm 1 corresponds to the expected size of the call stack of *lnk_sch*, which is equal to m multiplied by the expected running time of line 17. Observe that line 17 is executed at most m times starting from an initial call to *lnk_sch* until we hit the recursion stop condition (i.e., $\mathcal{T} = \emptyset$; see line 16 in Algorithm 1.) Recall that by the effect of the *while* loop of Algorithm 1, the size of $\mathcal{T}$ may shrink considerably each time *lnk_sch* is invoked until $\mathcal{T}$ is empty; see line 16 in Algorithm 1.

To calculate the expected running time of line 17, we define

$$\mathcal{X}_1 : \{((1, ..., i), j) \mid 1 \leq i \leq m \text{ and } 1 \leq j \leq i\} \rightarrow \{0, 1\}$$

to be a random variable for the cost of one run of line 17 (in Algorithm 1), where we extract a randomly selected transmission link j from unscheduled transmission links $(1, ..., i)$ such that $1 \leq i \leq m$. Thus, for a given $i \leq m$, for every j in $(1, ..., i)$, $\mathcal{X}_1(((1, ..., i), j)) = 1$ if and only if for all $l \neq j$, $\mathcal{X}_1(((1, ..., i), l)) = 0$. This is consistent with line 17 in Algorithm 1, where we select and extract one link. The time for extracting one link is constant, whereas no time is spent on the unselected links. Let $s^{(i)}$ denote a randomly selected link to be extracted from $(1, ..., i)$. For a given $i \leq m$, we note that the probability is

$$\mathcal{P}(\mathcal{X}_1(((1, ..., i), s^{(i)})) = 1) = \left(\frac{1}{i}\right)\left(\frac{1}{m}\right).$$

Observe that $\frac{1}{i}$ is the probability of selecting a link $s^{(i)}$ from i links, while $\frac{1}{m}$ is the probability of having a sequence of unscheduled links with size i. Therefore, the expectation, $\mathcal{E}(\mathcal{X}_1)$, of $\mathcal{X}_1$ is

$$\begin{aligned}\mathcal{E}(\mathcal{X}_1) &= \textstyle\sum_{i=1}^{m} \mathcal{X}_1(((1, ..., i), s^{(i)}))\mathcal{P}(\mathcal{X}_1(((1, ..., i), s^{(i)})) = 1) \\ &= \textstyle\sum_{i=1}^{m} (1)\left(\frac{1}{m}\right)\left(\frac{1}{i}\right) \in O\left(\frac{\ln m}{m}\right).\end{aligned}$$

Thus, the expected depth of the recursion tree of Algorithm 1 is equal to m multiplied by the expected running time of line 17:

$$O\left(m\left(\frac{\ln m}{m}\right)\right) = O(\ln m).$$

Proposition 2. *Let $\mathcal{N} = (\mathcal{S}, \mathcal{T})$ be a scheduled k-time slot network with n stations and m links and let the depth of the recursion tree of Algorithm 1 (running on $\mathcal{N}$) match the expectation, i.e., $O(\ln m)$. Thus, Algorithm 1 runs in time $O(m^{2+\ln k})$.*

Proof: Assuming at most k time slots are tried for each randomly selected link (see lines 17–19), the size of the recursion tree with a constant branching factor k and depth $O(\ln m)$ is in $O(k^{\ln m}) = O(m^{\ln k})$. As to the running time of the *while* loop of Algorithm 1, focus on the execution of t recursive calls for the procedure *lnk_sch* starting from the initial call (where $\mathcal{T}$ has all the links of the originally-inputted network), and ending with a call (for *lnk_sch*) where $\mathcal{T} = \emptyset$ (see line 16). In every call of these calls, in the *while* loop, we process $(m_r)_{r \leq t} \leq m$ links, those links that are in C (see lines 1 and 2 in Algorithm 1). In every call (for *lnk_sch*), the number of iterations of the *while* loop equals m_r. Moreover, in the *while* loop, the inner *for* loop (line 3) runs in $O(m^2) = O(m)$ time. This is because, for the first *for* (line 3), we only check those links α that have the sender (or receiver) being the sender or receiver of β. We do the same with the second *for* (line 5), where we only check those links γ with the sender being the sender of α. So, the overall running time of the *while* loop in execution t recursive calls is $O(m(m_1 + m_2 + ...m_t))$. Since $m_1 + m_2 + ... + m_t = m$, the *while* requires $O(m^2)$ time for executing the t recursive calls.

Therefore, the running time of Algorithm 1 is bounded by the size of its recursion tree, $O(m^{\ln k})$, multiplied by the running time of the *while* loop $O(m^2)$. Thus, Algorithm 1 runs in time $O(m^{2+\ln k})$.

5. Correctness proof of Algorithm 1

Algorithm 1 is a search algorithm that exactly computes link schedules for TDMA networks. We emphasize the correctness of Algorithm 1 in the following technical lemmas.

Lemma 1. *Throughout Algorithm 1's execution, for each link, x, in the input network, every time slot in $\mu(x)$ complies with constraints C.1 to C.5.*

Proof: The initial state of μ is consistent with the constraints C.1–C.5 because $\mu(x)$ initially has m time slots for each link in the input network. Certainly, transmission conflicts are avoided if one transmission link is exclusively permitted in a given time slot; recall there are m links in the input network. Now, we need to show that whenever we update μ, μ remains to adhere to C.1–C.5. Observe that μ is updated in two places in the algorithm: line 6 and line 12. Regarding line 6, whenever a time slot is assigned to a link β, for each unscheduled link γ such that there is a link α with $r(\alpha) = r(\beta)$ and $s(\alpha) = s(\gamma)$, we remove from $\mu(\gamma)$ the time slot of β. This is consistent with the constraints C.1–C.5. Regarding line 12, whenever a time slot is assigned to a link β, for each unscheduled link α, we remove from $\mu(\alpha)$ the time slot of β if $s(\alpha) = s(\beta)$, $s(\alpha) = r(\beta)$, $r(\alpha) = s(\beta)$, or $r(\alpha) = r(\beta)$; this is satisfying constraints C.1–C.5.

Corollary 1. *Referring to lines 7 and 13, Algorithm 1 backtracks whenever it sees an unscheduled link x with $\mu(x) = \emptyset$, i.e., x has no feasible time slots left.*

Lemma 2. *Throughout Algorithm 1's execution, λ does not violate constraints C.1 to C.5.*

Proof: The initial state of λ, where each link is not assigned a time slot yet, does not violate the conditions C.1 to C.5. Now, we need to show that whenever λ is updated, λ remains to adhere to C.1–C.5. We note that λ is updated in three places: lines 9, 15, and 19. In these places, we apply $\lambda(x) \leftarrow \tau$, (where x is a link and τ is a time slot) if, and only if, $\tau \in \mu(x)$, where μ is x's eligible time slots that satisfy C1.–C.5 as illustrated in a previous lemma.

Lemma 3. *At the end of Algorithm 1's execution,* Λ *(see line 16) is an optimal link schedule of the input network.*

Proof: Note that Λ is indeed a legal link schedule (i.e., adhering to constraints C.1–C.5). This is because according to line 16 in the algorithm, Λ is equal to λ, which is a legal schedule, as argued in a previous lemma. Still, we need to prove that Λ is optimal. With the intent to establish a contradiction, suppose Λ is not optimal. This means that there is Λ' with $\eta(\Lambda') < \eta(\Lambda)$; recall that $\eta(\Lambda)$ is defined as follows:

$$\eta(\Lambda) = \begin{cases} \max\{t \mid \exists \varphi \in \mathcal{T} \text{ with } t = \Lambda(\varphi)\}, & \text{if } \Lambda \neq \emptyset; \\ m+1, & \text{if } \Lambda = \emptyset. \end{cases}$$

The existence of another link schedule Λ' with $\eta(\Lambda') < \eta(\Lambda)$ is impossible because it contradicts the actions of the algorithm, where we go further in the search for an optimal link schedule by trying one more time slot only if we failed in finding a schedule with fewer time slots (this is the part "$\tau = 1 \,..\, m$" of line 18). Additionally, the condition $\tau < \eta(\Lambda)$ (in line 18) ensures not constructing a schedule with a number of time slots being equal to or greater than the number of time slots of the so-far-found schedule Λ.

6. Analogous results for computing optimal station schedule

Now, we compute optimal *station* schedules of TDMA networks. In Algorithm 2, we present the process of finding an optimal station schedule for a given network. Algorithm 2 employs data structures analogous to the data structures of Algorithm 1. Given a network $\mathcal{N} = (\mathcal{S}, \mathcal{T})$, let $\lambda' : \mathcal{S} \to \mathbb{N}_0$ be a total mapping representing an under-construction station schedule. During the algorithm's execution, a station x with $\lambda'(x) = 0$ indicates that x is not assigned a time slot yet. Let $\mu' : \mathcal{S} \to 2^{\{1,\ldots,n\}}$ be a total mapping to associate a station with a set of *feasible* time slots: those that do not violate the conditions C.1–C.3 & C.6 consistently with the under-construction station schedule λ'. Algorithm 2 uses a set $C' \subseteq \mathcal{S}$ to hold those stations that are newly assigned a time slot. Algorithm 2 will apply the consequences of the newly-scheduled stations held in C as we elaborate shortly. Another construct used in Algorithm 2 is the maximum time slot, $\eta'(\Gamma')$, assigned to a station under some nonempty mapping $\Gamma' : \mathcal{S} \to \mathbb{N}_0$. More precisely, we define

$$\eta'(\Gamma') = \begin{cases} \max\{t \mid \exists x \text{ with } t = \Gamma'(x)\}, & \text{if } \Gamma' \neq \emptyset; \\ n+1, & \text{if } \Gamma' = \emptyset. \end{cases}$$

Referring to Algorithm 2, invoke $st\_sch(\mathcal{S}, C', \lambda', \mu')$ such that $\Lambda' = \emptyset$, $C' = \emptyset$, and for all x, $\mu'(x) = \{1, ..., n\}$ with $\lambda'(x) = 0$; then an optimal station schedule $\Lambda' \neq \emptyset$ will be found at the end of the algorithm's execution. Observe that Λ' is implicitly inputted to the procedure since it is not included in the input list (see the caption of Algorithm 2). This is intentionally the case because the algorithm maintains only one copy of Λ', which will contain, at the end of the execution of the algorithm, an optimal station schedule. One may implement Λ' as a global variable in computer programming. Likewise, $\mathcal{T}$ is supposed to be a global variable rather than a parameter to be passed on every call to the procedure *st_sch*.

Algorithm 2: $st\_sch(\mathcal{S}, C', \lambda', \mu')$

```
while $C' \neq \emptyset$ do
    extract a station $x$ from $C'$;
    foreach $y \in \mathcal{S}$: $(x,y) \in \mathcal{T} \vee (y,x) \in \mathcal{T} \vee \exists z((x,z) \in \mathcal{T} \wedge (y,z) \in \mathcal{T})$ do
        $\mu'(y) \leftarrow \mu'(y) \setminus \{\lambda'(x)\}$;
        if $\mu'(y) = \emptyset$ then return;
        if $|\mu'(y)| = 1$ then
            $\lambda'(y) \leftarrow \tau$ where $\tau \in \mu'(y)$; $C' \leftarrow C' \cup \{y\}$; $\mathcal{S} \leftarrow \mathcal{S} \setminus \{y\}$;
if $\mathcal{S} = \emptyset$ then $\Lambda' \leftarrow \lambda'$; return;
extract a randomly selected station $x$ from $\mathcal{S}$;
foreach $\tau = 1 \,..\, n : \tau \in \mu'(x) : \tau < \eta'(\Lambda')$ and $\tau \leq \eta'(\lambda') + 1$ do
    $\lambda'(x) \leftarrow \tau$; $st\_sch(\mathcal{S}, \{x\}, \lambda', \mu')$;
```

Let us navigate into the actions of Algorithm 2. As we initially invoke the algorithm with $C' = \emptyset$ and $\mathcal{S} \neq \emptyset$, move to line 9 where the algorithm selects an unscheduled station $x \in \mathcal{S}$ to assign for it a time slot. Referring to lines 10–11, the algorithm tries every feasible time slot $\tau \in \mu'(x)$ satisfying that $(\bar{i})$ $\tau < \eta'(\Lambda')$ and $(\overline{ii})$ $\tau \leq \eta'(\lambda') + 1$. The conditions of $(\bar{i})$ & $(\overline{ii})$ are analogous to the earlier conditions (i) & (ii) used in Algorithm 1; to avoid redundant discussions, we refer to the treatment given earlier for (i) & (ii).

Once the algorithm assigns a time slot to a station x, Algorithm 2 calls itself recursively (line 11) passing on the parameters: $\mathcal{S}$, λ', μ', and $\{x\}$ such that $\{x\}$ is to be copied to C' in the new instance of *st_sch*. Note after every invocation to *st_sch*, we have a new instance of the procedure *st_sch*. Referring to the *while* loop in line 1, whenever C' is nonempty, the algorithm (lines 3–7) will impose the consequences of the newly-assigned time slot $\lambda'(x)$ of a station x (extracted from C' in line 2) on the set of feasible time slots $\mu'(y)$ for every station y, such that $(*)$ $(x,y) \in \mathcal{T}$, $(**)$ $(y,x) \in \mathcal{T}$, or $(***)$ $\exists z$ with $(x,z) \in \mathcal{T}$ and $(y,z) \in \mathcal{T}$.

In line 4, the algorithm removes $\lambda'(x)$ from the set of feasible time slots, $\mu'(y)$, of a station y satisfying $(*)$, $(**)$, or $(***)$. This is to maintain the conditions C.1–C.3 and C.6. Recall that we build station schedules in the scenarios where a station must transmit all its messages in one time slot. So, to keep condition C.6 satisfied, by assigning a time slot t to a station x, we let x be able to transmit a message to every station w with $(x,w) \in \mathcal{T}$ during the time slot t. Therefore, it is not hard to see that the conditions $(*)$, $(**)$, and $(***)$, along with the performed action (line 4), maintain the conditions C.1, C.3 and C.2 satisfied.

After removing $\lambda'(x)$ from $\mu'(y)$ (line 4), Algorithm 2 (line 5) checks if the set of feasible time slots, $\mu'(y)$, of y is empty, and if so, the algorithm returns to the caller of *st_sch*. In line 6, the algorithm verifies if $\mu'(y)$, the set of feasible time slots of y has one time slot only, and if so, the algorithm assigns the only possible time slot to y; subsequently, the algorithm (line 7) adds y to C' and then removes y from $\mathcal{S}$. The *while* loop continues until C' is empty, which means there are no more newly scheduled stations. In line 8, the algorithm checks if $\mathcal{S}$ is empty. The set $\mathcal{S}$ being empty indicates that all stations have been scheduled, and, hence, the under-construction station schedule λ' (being completed) is the best station schedule so far constructed. So, the algorithm saves a copy of λ' in Λ' and immediately returns to the caller of the procedure *st_sch*. If $\mathcal{S}$ is not empty, the algorithm continues as described above.

Given the similarity between Algorithms 1 and 2, the space and time analyses of Algorithm 2 are analogous to those discussed for Algorithm 1. Thus, if the input is a scheduled k-time slot network, Algorithm 2's recursion tree has depth $O(\ln n)$ in expectation. Moreover, Algorithm 2 runs in time $O(n^{3+\ln k})$ if the input is a scheduled k-time slot network and the recursion tree has depth matching the expectation. This is because the *while* loop of Algorithm 2 runs in $O(n^3)$ time, whereas the size of Algorithm 2's recursion tree is in $O(n^{\ln k})$. Regarding space complexity, Algorithm 2 requires $O(n^3)$ space for holding at maximum n instances of a two-dimensional array representing μ'.

7. Conclusions

We demonstrated that an optimal link schedule for a TDMA network with n stations and m links can be computed recursively with a recursion tree of logarithmic depth $O(\ln m)$ in expectation. Thereby, we illustrated that optimal link schedules for those TDMA networks, with recursion trees of depth matching the expectation, can be found in time $O(m^{2+\ln k})$. Moreover, we discussed similar results for computing optimal station schedules of TDMA networks. As we discussed throughout the article, the main advantage of the algorithm is that the expected depth of its recursion tree is logarithmic. However, although the time complexity of our algorithm (running with a logarithmic-depth recursion tree) is $O(m^{2+\ln k})$, its running time in the general case remains exponential as the problem of finding an optimal link schedule of a TDMA network is NP-hard.

Future work may investigate if the running time at each node of the recursion tree can be improved, significantly impacting the overall performance of computing optimal link or station schedules of TDMA networks. Likewise, the space complexity of computing TDMA optimal transmission schedules can be examined further for better bounds, especially to make it more practical for real applications. Our research results can also be pursued in other directions that are aligned with the literature. One may consider the impact of different traffic patterns or data rates on the performance of TDAM networks and develop adaptive scheduling algorithms to optimize throughput under varying conditions. Additionally, it would be valuable to build on existing literature and investigate further the trade-off between achieving optimal throughput and minimizing energy consumption in TDMA networks, considering the increasing importance of energy-efficient communication protocols. Likewise, further research may examine the implementation of the proposed algorithm in hardware or real-time systems.

Use of AI tools declaration

The author declares he has not used Artificial Intelligence (AI) tools in the creation of this article.

Conflict of interest

The author declares no conflict of interest.

References

1. P. Adasme, F. Seguel, I. Soto, E. S. Juan, Spatial time division multiple access for visible light communication networks, *2017 First South American Colloquium on Visible Light Communications (SACVLC)*, Santiago, Chile, 2017, 1–6. https://doi.org/10.1109/SACVLC.2017.8267605

2. S. R. Akbar, M. H. H. Ichsan, A. A. Darmawan, Reference broadcast synchronization and time division multiple access implementation on WSN, *Telkomnika (Telecommunication Computing Electronics and Control)*, **17** (2019), 291–298. http://doi.org/10.12928/telkomnika.v17i1.11615

3. S. Andersson, B. Hagerman, M. Berg, H. Dam, U. Forssen, J. Karlsson, et al., Adaptive antennas for global system for mobile communications and time division multiple access (interim standard-136) systems, In: *Handbook of antennas in wireless communications*, Boca Raton: CRC Press, 2002, https://doi.org/10.1201/9781315220031

4. E. Arikan, Some complexity results about packet radio networks (Corresp.), *IEEE T. Inform. Theory*, **30** (1984), 681–685. https://doi.org/10.1109/TIT.1984.1056928

5. X. Chen, W. S. Hu, D. Feng, Direct-sequence spread spectrum time division multiple access with direct detection for latency optimized passive optical network, *Opt. Commun.*, **510** (2022), 127955. https://doi.org/10.1016/j.optcom.2022.127955

6. S. Even, O. Goldreich, S. Moran, P. Tong, On the np-completeness of certain network testing problems, *Networks*, **14** (1984), 1–24. https://doi.org/10.1002/net.3230140102

7. N. Fahier, W. C. Fang, An HBC-based biosensor transmitter for time division multiple access network, *2017 IEEE International Conference on Consumer Electronics-Taiwan (ICCETW)*, Taipei, Taiwan, 2017, 161–162. https://doi.org/10.1109/ICCE-China.2017.7991045

8. H. Falsafain, M. R. Heidarpour, S. Vahidi, A branch-and-price approach to a variant of the cognitive radio resource allocation problem, *Ad Hoc Netw.*, **132** (2022), 102871. https://doi.org/10.1016/j.adhoc.2022.102871

9. S. Faruque, Time division multiple access (TDMA), In: *Radio frequency multiple access techniques made easy*, Cham: Springer, 2019, 35–43. https://doi.org/10.1007/978-3-319-91651-4_4

10. X. Fernando, G. Lazaroiu, Spectrum sensing, clustering algorithms, and energyharvesting technology for cognitive-radio-based internet-of-things networks, *Sensors*, **23** (2023), 7792. https://doi.org/10.3390/s23187792

11. C. V. Giriraja, V. Chirag, C. Sudheendra, S. S. Bellur, T. K. Ramesh, K. N. Meera, Priority based time-division multiple access algorithm for medium range data communication in very high frequency radios, *J. Comput. Theor. Nanos.*, **17** (2020), 290–296. https://doi.org/10.1166/jctn.2020.8664

12. C. Goswami, P. Sharma, R. Bharati, K. C. Rajheshwari, L. P. Maguluri, M. Elangovan, Algorithms for high mobility environment in 5g radio access networks with millimeter wave communications, *Opt. Quant. Electron.*, **56** (2024), 276. https://doi.org/10.1007/s11082-023-05858-7

13. S. D. Ilcev, Analyses of time division multiple access (TDMA) schemes for global mobile satellite communications (GMSC), *International Journal on Marine Navigation and Safety od Sea Transportation*, **14** (2020), 831–837. http://doi.org/10.12716/1001.14.04.06

14. Y. H. Kwon, J. H. Cha, D. S. Kim, Design and performance analysis of asymmetric time division multiple access method for improving response time and throughput in tactical radio communication, *Journal of the Korea Institute of Military Science and Technology*, **21** (2018), 361–369. https://doi.org/10.9766/KIMST.2018.21.3.361

15. J. Lee, W. C. Jeong, B. C. Choi, Multi-channel time division multiple access time slot scheduling with link recovery for multi-hop wireless sensor networks, *Int. J. Distrib. Sens. N.*, **13** (2017), 1550147717726311. https://doi.org/10.1177/1550147717726311

16. C. Li, Z. Y. Xu, J. Y. Wang, J. Y. Zhao, A. L. Qi, J. H. Li, Ultraviolet random access collaborative networking protocol based on time division multiple access, *J. Opt. Commun. Netw.*, **15** (2023), 393–403. https://doi.org/10.1364/JOCN.479471

17. Y. Liang, P. P. Wen, H. X. Wang, Z. F. Zhao, An improved time division multiple access protocol in UWSN, *2017 International Conference on Mechanical, Electronic, Control and Automation Engineering (MECAE 2017)*, Beijing, China, 2017, 208–213. https://doi.org/10.2991/mecae-17.2017.39

18. J. H. Luo, L. Y. Fan, S. Wu, X. T. Yan, Research on localization algorithms based on acoustic communication for underwater sensor networks, *Sensors*, **18** (2017), 67. https://doi.org/10.3390/s18010067

19. K. H. Mohammadani, K. A. Memon, I. Memon, N. N. Hussaini, H. Fazal, Preamble time-division multiple access fixed slot assignment protocol for secure mobile ad hoc networks, *Int. J. Distrib. Sens. N.*, **16** (2020), 1550147720921624. https://doi.org/10.1177/1550147720921624

20. H. Mokari, E. Firouzmand, I. Sharifi, A. Doustmohammadi, Deception attack detection and resilient control in platoon of smart vehicles, *2022 30th International conference on electrical engineering (ICEE)*, Tehran, Iran, Islamic Republic, 2022, 29–35. https://doi.org/10.1109/ICEE55646.2022.9827376

21. H. Mokari, E. Firouzmand, I. Sharifi, A. Doustmohammadi, Resilient control strategy and attack detection on platooning of smart vehicles under DoS attack, *ISA T.*, **144** (2024), 51–60. https://doi.org/10.1016/j.isatra.2023.11.019

22. A. A. Monterrosas, W. D. Kight, E. Gunduzhan, D. M. Coleman, A selfsynchronizing time-division multiple-access network protocol, *2019 Wireless Telecommunications Symposium (WTS)*, New York, USA, 2019, 1–5. https://doi.org/10.1109/WTS.2019.8715529

23. S. Ramanathan, E. L. Lloyd, Scheduling algorithms for multihop radio networks, *IEEE-ACM T. Network.*, **1** (1993), 166–177. https://doi.org/10.1109/90.222924

24. M. R. Rezaee, N. A. W. A. Hamid, M. Hussin, Z. A. Zukarnain, Fog offloading and task management in iot-fog-cloud environment: Review of algorithms, networks and SDN application, *IEEE Access*, **12** 2024, 39058–39080. http://doi.org/10.1109/ACCESS.2024.3375368

25. L. Rocha, D. Sasaki, The backbone packet radio network coloring for time division multiple access link scheduling in wireless multihop networks, *Networks*, **71** (2018), 403–411. https://doi.org/10.1002/net.21767

26. B. Schieber, B. Samineni, S. Vahidi, Interweaving real-time jobs with energy harvesting to maximize throughput, In: *WALCOM: Algorithms and Computation*, Cham: Springer, 2023, 305–316. https://doi.org/10.1007/978-3-031-27051-2_26

27. G. Sebestyen, J. Kopiak, Assessing the stability of time-division multiple access (TDMA) mesh networks through flooding route selection, *2023 IEEE 21st Jubilee International Symposium on Intelligent Systems and Informatics (SISY)*, Pula, Croatia, 2023, 000649–000652. https://doi.org/10.1109/SISY60376.2023.10417892

28. S. Sharma, V. Kumar, K. Dutta, Multi-objective optimization algorithms for intrusion detection in iot networks: A systematic review, *Internet of Things and Cyber-Physical Systems*, **4** (2024), 258–267. https://doi.org/10.1016/j.iotcps.2024.01.003

29. E. Shayo, P. Mafole, A. Mwambela, A survey on time division multiple access scheduling algorithms for industrial networks, *SN Appl. Sci.*, **2** (2020), 2140. https://doi.org/10.1007/s42452-020-03923-4

30. M. Shwetha, S. Krishnaveni, A systematic analysis, outstanding challenges, and future prospects for routing protocols and machine learning algorithms in underwater wireless acoustic sensor networks, *J. Interconnect. Netw.*, **2024** (2024), 2330001. https://doi.org/10.1142/S0219265923300015

31. H. Son, C. Jeong, Virtual mimo beamforming for opportunistic cooperative time division multiple access, *IEEE T. Commun.*, **68** (2020), 7521–7532. https://doi.org/10.1109/TCOMM.2020.3019287

32. H. Son, B. Kwon, Ris-assisted cooperative time-division multiple access, *Sensors*, **24** (2023), 178. https://doi.org/10.3390/s24010178

33. Y. X. Sun, Z. H. Zhang, X. P. Li, S. Xiao, W. B. Tang, An extensible frame structure for time division multiple access medium access control in vehicular ad-hoc networks, *T. Emerg. Telecommun. T.*, **31** (2020), e3912. https://doi.org/10.1002/ett.3912

34. A. B. Tambawal, R. M. Noor, R. Salleh, C. Chembe, M. H. Anisi, O. Michael, et al., Time division multiple access scheduling strategies for emerging vehicular ad hoc network medium access control protocols: a survey, *Telecommun. Syst.*, **70** (2019), 595–616. https://doi.org/10.1007/s11235-018-00542-8

35. P. Tirandazi, A. Rahiminasab, M. Ebadi, An efficient coverage and connectivity algorithm based on mobile robots for wireless sensor networks, *J. Ambient Intell. Human. Comput.*, **14** (2023), 8291–8313. https://doi.org/10.1007/s12652-021-03597-9

36. S. Wang, Y. C. Wang, D. Y. Li, Q. C. Zhao, Distributed relative localization algorithms for multi-robot networks: a survey, *Sensors*, **23** (2023), 2399. https://doi.org/10.3390/s23052399

37. A. Wood, T. Upthegrove, C. Funai, B. Thapa, S. Loos, D. Javorsek, Pulsenet: A discriminative model for time division multiple access time slot prediction, *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*, Rockville, MD, USA, 2022, 1–6. https://doi.org/10.1109/MILCOM55135.2022.10017899

38. J. Wu, L. Yu, F. F. Gou, Data transmission scheme based on node model training and time division multiple access with iot in opportunistic social networks, *Peer-to-Peer Netw. Appl.*, **15** (2022), 2719–2743. https://doi.org/10.1007/s12083-022-01365-w

39. A. Yaman, T. van der Lee, G. Iacca, Online distributed evolutionary optimization of time division multiple access protocols, *Expert Syst. Appl.*, **211** (2023), 118627. https://doi.org/10.1016/j.eswa.2022.118627

40. Z. J. Yan, Q. Q. Li, B. Li, M. Yang, A link distance division based time division multiple access protocol for directional aeronautical relay networks, *Journal of Northwestern Polytechnical University*, **38** (2020), 147–154. http://doi.org/10.1051/jnwpu/20203810147