Allaoui, T.; Yagoubi, MB.; Kerrache, CA.; Tavares De Araujo Cesariny Calafate, CM. (2019). NFK: a novel fault-tolerant K-mutual exclusion algorithm for mobile and opportunistic ad hoc networks. International Journal of Information and Communication Technology. 15(2):176-197. https://doi.org/10.1504/IJICT.2019.102479

# *NFK: a novel fault-tolerant K-mutual exclusion algorithm for mobile and opportunistic ad hoc networks*


Tahar Allaoui* and Mohamed Bachir Yagoubi

LIM,
University of Laghouat,
BP 37G, route de Ghardaia,
Laghouat, Algeria
Email: t.allaoui@lagh-univ.dz
Email: m.yagoubi@lagh-univ.dz
*Corresponding author

Chaker Abdelaziz Kerrache

University of Ghardaia,
P.O. Box 455,
Ghardaia, 47000, Algeria
Email: ch.kerrache@lagh-univ.dz

Carlos T. Calafate

Department of Computer Engineering,
Universitat Politècnica de València,
Camino de Vera, S/N, 46022 València, Spain
Email: calafate@disca.upv.es



**Abstract:** This paper presents a fault-tolerant algorithm ensuring multiple resources sharing in mobile ad hoc networks (MANETs) that is able to handle the known K-mutual exclusion problem in such mobile environments. The proposed algorithm relies on a token-based strategy, and requires information about resources and their use to be carried in routing protocol control messages. This way, our solution avoids any additional exchange of messages. Furthermore, experimental results show that it offers a fast response time. Moreover, we introduce a dual-layer fault-tolerance mechanism that tolerates the faults of several sites at the same time without affecting the well functioning of the system. Simulation results also evidence the high efficiency of our proposal, which achieves reduced overhead and response delay even in the presence of critical situations where multiple simultaneous faults occur.



**Keywords:** NFK; resource sharing; K-mutual exclusion; fault tolerance; mobile ad hoc networks; MANETs.

**Biographical notes:** Tahar Allaoui is currently an Assistant Professor at Amar Telidji University, Laghouat, Algeria. He received his MSc in 2007 and he is currently pursuing his PhD. His research interests are mainly related to k-mutual exclusion, distributed systems, fault tolerant, and network routing.

Mohamed Bachir Yagoubi is currently a Professor at Amar Telidji University, Algeria. He received his MSc in 1992 from Paris 13 University, Paris, France and PhD in Computer Science in 1997 from Evry-val-d'Essonne University, France. His research interests include distributed systems, fault tolerance, security and vehicular networks.

Chaker Abdelaziz Kerrache received his MSc in Computer Science at the University of Laghouat, Algeria, in 2012, and his PhD in Computer Science at the University of Laghouat, Algeria, in 2017. In 2013, he joined the Informatics and Mathematics Laboratory (LIM) as a Research Assistant and the Computer Networks Group (GRC) in 2015 as a visiting PhD student. His research activity is related to trust and risk management, secure multi-hop communications, vehicular networks and UAVs.

Carlos T. Calafate is an Associate Professor in the Department of Computer Engineering at the Technical University of Valencia (UPV) in Spain. He graduated with honours in Electrical and Computer Engineering at the University of Oporto (Portugal) in 2001. He received his PhD in Informatics from the Technical University of Valencia in 2006, where he has worked since 2002. His research interests include ad-hoc and vehicular networks, mobile applications, QoS, network protocols, video streaming and network security.

## 1 Introduction

Nowadays, the development of wireless technologies offers new perspectives in the field of computing. In fact, wireless networks allow users to gain access to information independently of time and place. Due to their many advantages, they gained major interest and an increasing popularity in both academic and industrial domains (Rappaport et al., 1996; Prabhu et al., 2017).

An *ad hoc* network is a set of mobile hosts interconnected by a wireless medium, forming a temporary network without any communications infrastructure support. The mobile ad hoc networks concept assumes that all (or the majority) of the components in the environment are mobile. So, contrary to infrastructure-based networks, no centralised administration is available; instead, mobile hosts themselves form the infrastructure of the network. Also, no assumption or limitation is made regarding the size of the mobile ad hoc network, meaning that the network can potentially contain hundreds of mobile units (Nasipuri, 2004).

The absence of a centralised administration in ad hoc networks, and the possibility of sharing resources between the various sites of the network, can jointly lead to incoherences due to simultaneous access to the same resource from several locations, and so it becomes of utmost importance to guarantee the exclusive access to this resource. This notion gave birth to the mutual exclusion problem (MUTEX) in mobile ad hoc networks (Nasipuri, 2004). This problem can be generalised to the K-mutual

exclusion problem (K-MUTEX) when the system is endowed with several copies of the same resource (Bulgannawar and Vaidya, 1995).

These two problems were adequately addressed in conventional distributed systems, and the proposed algorithms can be classified in two main categories:

1 permission-based algorithms (Saxena and Rai, 2003)

2 token-based algorithms (Swaroop and Singh, 2007).

However, the assumption that all nodes are safe from any kind of faults can lead to unexpected and unwanted situations.

The purpose of the proposed algorithms is to insure the K-MUTEX through the exchange of a reduced number of messages to avoid overloading the network, and to reduce energy consumption. Unfortunately, fault tolerance is rarely considered.

In this paper, we propose a new algorithm to solve the K-MUTEX problem in mobile ad hoc networks that supports fault tolerance. Our solution takes advantage of the AODV routing protocol (Chakeres and Belding-Royer, 2004) to achieve a reduced exchange of messages between the various mobile hosts whenever access to a critical resource takes place. Moreover, we extend AODV control messages to carry both critical resource availability and fault tolerance information.

After ensuring the distribution of token availability data on the fly (together with control messages), our proposal offers a double-layer recovery strategy to face any kind of nodes failure. Thus, our proposal, which we named 'NFK', is an AODV-based, fully distributed, timely, and fault-tolerant K-MUTEX solution for mobile ad hoc networks (MANETs).

The remainder of the paper is organised as follows: in Section 2, we present a review of existing K-MUTEX solutions. In Section 3, we provide an overview of our proposal, and then go through the main algorithms used for building a reliable and safe resource sharing solution in Section 4. Afterward, we present our double-layer recovery strategy to ensure fault-tolerance in Section 5. In Section 6, the simulation environment is described, along with the discussion of simulation results. Finally, some concluding remarks are provided in Section 7.

## 2 Related works

Since the distributed systems and parallel computing have emerged, the resource sharing problem has gained much interest from both academic and industrial research communities. In this section, we summarise the main existing solutions dealing with the K-MUTEX exclusion problem over MANETs. These solutions can be classified as either fault-sensitive or fault tolerant approaches.

### *2.1 Fault-sensitive K-MUTEX approaches for MANETs*

In this section we focus on k-MUTEX solutions without fault tolerance mechanisms. We start by discussing on-cluster centred solutions, and then move on to fully distributed ones.

### 2.1.1 *Cluster-based approaches*

To overcome message complexity, clustering-based architectures are the main existing solutions. Same as all MUTEX solutions, all existing solutions falling under this category are, to the best of our knowledge, token-based solutions.

A cluster-based K-MUTEX algorithm for MANETs is proposed in Haghighat and Mohamadi (2011). In this algorithm, the network must be divided into at least K clusters. Based on the Raymond algorithm (Raymond, 1989), nodes within the cluster get the role of head, gateway or ordinary node, where every cluster head maintains the list of requesting nodes. The cluster head attempts to serve demands using the cluster's own token, but if there is no available token, it forwards the request to the other cluster heads. This technique was proposed to minimise the number of exchanged messages. However, cluster head election may become a serious problem, especially in mobile environments.

In Erciyes (2004), author proposes an architecture consisting of a ring of clusters for distributed mutual exclusion algorithms in the scope of mobile networks. They periodically re-divide the network into new clusters based on the *fixed centred partitioning* algorithm, and then every cluster implements a separate MUTEX algorithm. They applied Ricart-Agrawala and token-based algorithms to evaluate the performance of their architecture. Despite simulation results show that message complexity is considerably reduced, this solution cannot be considered as an adequate option for solving the K-MUTEX problem over MANETs.

Same as Erciyes (2004), Dagdeviren and Erciyes (2007) propose an architecture for MUTEX handling in MANETs. Specifically, they divide their software architecture into three layers. At the lowest layer, the merging clustering algorithm (MCA) periodically partitions the MANET into a number of balanced clusters. At the second layer, the backbone formation algorithm (BFA) provides a virtual ring using the cluster heads found by MCA. The final layer is represented by the implementation of a modified and scaled version of the Ricart-Agrawala algorithm. Unfortunately, this architecture inherits the previously mentioned problems.

### 2.1.2 *Fully distributed approaches*

Solutions falling in this category do not make any clustering assumptions, and their communications are fully distributed. Generally, this category's solutions suffer from the generation of an excessive number of messages.

- *Token-based approaches*:

  A token-based distributed algorithm for supporting MUTEX in opportunistic networks is proposed in Tamhane and Kumar (2012). This approach tries to achieve a trade-off between the dynamic MANET topology and the token passing process, and it can be divided into three main procedures:

  1 request generation

  2 request propagation

  3 token propagation.

Every node in a critical section (CS) case maintains the list of demanding nodes, and then it releases the Token for the first demander and acts as an intermediate for the others. Even if this approach can guarantee that all demanders will be served, it cannot handle link or node failures. The work in Thiare and Naimi (2009) presents a group K-MUTEX algorithm for MANETs where authors assume that there is a unique token initially, and then utilise the partial reversal technique to maintain a token-oriented DAG (directed acyclic graph). Then, a node holding the token can enter the CS, and inform all its requesting neighbours by sending out Okay messages. When a neighbour node receives the Okay message, it can enter the CS if, and only if, it requests for the same resource as the token holder. Additional information messages and the continuous DAG computation are the main problems of this technique. Furthermore, there is no complexity study nor simulation tests that validate this approach.

In another work, Walter et al. (2001) propose a DAG-based k-resources sharing technique for MANETs where the token holders are considered as *sink-points*. In addition, every node maintains a queue of requesting node identities. The problems with this solution are mainly:

1 the assumption that the token is reliable and will never disappear

2 that duplicates do not occur

3 that node failures do not occur.

- *Permission-based approaches*:

  Due to the unpredictable and dynamic topology of MANETs, most of the existing works adopt token passing strategies instead of permission gathering. Hence, only a few works fall under this category.

  A permissions-based technique for solving the K-MUTEX problem is proposed in Masum et al. (2010). The authors of this technique tried to overcome mobility and disconnection problems by gathering enough consensuses for every mobile node intending to enter the CS. They propose creating a specific message containing the information required to reach such consensus, including ProcessID, LastRequestID and LastRequestTS (TimeStamp), among others. Unfortunately, this approach does not take into account MANET's delay sensitivity, nor the additional messages overhead or the energy consumption associated to the transmission process.

### *2.2 Fault-tolerant K-MUTEX approaches for MANETs*

Due the nature of MANETs, resource sharing algorithms should take into account the high failure probabilities of both nodes and communication links. In this part, we provide an overview of the main existing solutions offering fault-tolerance mechanisms.

In Wu et al. (2007), the mobile dual-token MUTEX (MDTM) algorithm for detection of token loss is proposed. The algorithm detects token loss due to links problems only by using two types of tokens, primary and secondary. Each token has its own dynamic ring and a coordinator. Only the primary token grants permission to enter the CS. If a token completes a cycle over the ring and does not find the ID of other token on any of the nodes, it assumes that the latter is lost and must be regenerated. We find that

this approach is not really suitable for MANETs because of the ring assumption. The same authors have proposed another permission-based fault tolerant solution in Wu et al. (2008). It is based on a look-ahead technique similar to medium access techniques. In particular, this technique enforces MUTEX only among the hosts currently competing for the critical section, thus reducing the number of exchanged messages. In addition, they use a predefined timeout to detect mobile hosts' disconnection. Despite supporting high levels of mobility, this permissions-based approach introduces significant delay problems.

Another fault tolerant K-MUTEX technique for MANET was proposed in Moallemi et al. (2007). This solution extends the well-known Naimi-Trehels algorithm (Trehel and Naimi, 1987) with an entropy-based clustering algorithm where cluster heads manage the token passing among requesting nodes. However, authors fail to provide experimental results or analytical proofs of this solution. Hence, the actual performance of this technique remains unknown.

### *2.3 State of the art review considerations*

Based on this hovering upon the existing solutions in the literature, it becomes clear that the most adequate solution for MANET environments is the token-based ones. Moreover, most of the existing solutions rely on pre-formed cluster, a solution that cannot always be applicable, especially in highly dynamic MANETs. In addition, low battery, link quality and node mobility are also among the main challenges for resource sharing algorithms.

To overcome the aforementioned problems, in this paper we propose a double-layer recovery strategy together with a token-passing technique. Moreover, our solution is fully distributed and independent, and so it can be combined with any routing protocol.

In the sections that follow we provide a clear step-by-step description of our solution assumptions, algorithms and performance achieved.

## 3 Proposal overview

In MANETs, all the applications which require a communication between two hosts require at first the establishment of a link between these two hosts. This is insured by the routing protocol of the network. The establishment of links is insured by the routing protocol at the network layer, while the various applications, including those of the K-MUTEX, run at the application layer.

For token-based K-MUTEX algorithms, the requester sends its demand to a single particular site, or it sends its demand towards a set of sites and waits for the answer. In both cases, a high number of message exchanges between sites will take place, leading to network overload. Moreover, to insure this communication, a valid path must be already established by the routing protocol, which also requires message exchanges.

We notice that the use of the shared resource requires an exchange of messages in the lower layer, and another exchange of messages in the upper layer as well. Thus, it quickly becomes obvious that network overload may take place.

Our approach consists of reducing the number of exchanged messages for every CS entry, and of exploiting the inevitable messages of the routing protocol to carry additional information that insures token exchanges between the various sites.

To solve the problem of the K-MUTEX, K tokens are used to satisfy the requests. These tokens are initially held by K sites, and can be exchanged between the various sites. Thus, at any given time, every site can be in one of the following status:

1 the site holds a free token

2 the site uses a token

3 the site does not hold a token.

Our idea consists in adding two fields in the routing table of each site to indicate the presence and the usage of tokens. The first field is the Present_token field; it indicates that the site holds a free token. The second field is the used_token field, which indicates that the site uses a token.

We rely on the AODV routing protocol because it is a reactive algorithm which allows to establish a link between two network nodes by using the shortest path. To achieve this it creates routing tables on demand, assuring that the information is always relevant.

Figure 1 gives an example of the routing tables of three sites (nodes). We notice the distance (cost) between the sites, which is indicated in the table. Figure 1(b) explains the changes in the routing table, where we can notice the two additional bits.

**Figure 1** Proposed messages format extension (see online version for colours)

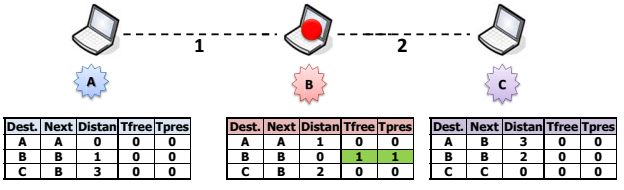



**Table 1** Notations used

| | |
|---|---|
| $AODV$ | Ad hoc on-demand distance vector |
| $K\text{-}MUTEX$ | K-mutual exclusion |
| $CS$ | Critical section |
| $LS_{i,j}$ | Link stability between two nodes '$i$' and '$j$' |
| $\alpha$ | Peak cases avoidance factor |
| $AODVi$ | Routing table of the node '$i$' |
| $V_i(t)$ | Velocity of the node '$i$' at time '$t$' |
| $NS$-2 | Network Simulator 2 |

The site holds a token, meaning that this information will be available in its routing table, and by dint of the routing protocol, this information will be available in the routing tables of the other sites.

When a site wishes to enter the CS, it updates its routing table, and it chooses the closest site among the sites which hold tokens. This information is available in the routing table, and thus a request will be directly sent towards this site without the need to perform a token search. The latter answers favourably to this demand by returning the token. Upon receiving the token, the requesting site allows it to access the CS. Notice that all necessary updates are handled automatically by the routing protocol.

It is worth pointing out that the use of this method avoids additional message exchanges because the information concerning tokens and their location is in general available for all the sites in their routing tables.

Table 1 details the used notations and their meanings.

## 4 Proposal details

The system consists of N sites numbered from 1 to N, and of K resources. To insure the K-MUTEX, we use K tokens that are initially held by the sites in a random manner.

### *4.1 Data design*

Every node i in the system maintains the following local variables:

- Status: a variable which indicates the status of the site.
- Requesting: a Boolean variable indicating if the site asked a critical resource or not, initialised at false.
- Token: a Boolean variable indicating if the site possesses a token or not; it is initialised at True for the sites holding tokens, and at false for the other sites.
- Next: a waiting queue which contains the identities of the requesting sites. This queue is sent with the token to a requesting site in order to satisfy the other sites.
- AODV (X, Y): indicates both the fields which can be manipulated by our algorithm: X indicates the free tokens, and Y indicates the used tokens. In our algorithm, we use also the following messages:
- Request (Token, i): sent by the site i towards the site which holds a token.
- Agreement (Token, Queue): sent by the site which holds the token to a requesting site to allow it to use the CS.
- Reject: sent by a former holder of the token to a requesting site to inform it that it does not hold the token any more.

### *4.2 Proposal functionalities*

In this section we present the code of the algorithm with the explanation of each procedure.

In Algorithm 1, the requesting site sends its demand to the closest holder of a free token, and waits for its reception to enter the CS.

**Algorithm 1** Requesting the CS

```
1: Requesting ← True;
2: if (AODVi_[i]_free token = 0) then
3:     Holder ← Find the adequate holder (AODVi);
4:     Send Request (Token, i) to Holder;
5: end if
6: Wait for (Token = True);
7: Modification in the routing table AODVi (0,1);
8: < Enter the CS >
```

In addition, when a site receives an agreement message, it sets its local variable at true, and updates the waiting queue. Algorithm 2 summarises this process.

**Algorithm 2** Reception of agreement (Token, Queue)

```
1: Token ← True;
2: Next ← Queue;
```

Using Algorithm 3, if a site receives a request message, it is going to send the token towards the requesting site if it does not use the resource. Otherwise, it puts the requesting site on hold. In case the site does not hold a token any more, it is going to send a reject message to the requesting site.

**Algorithm 3** Reception of request (Token, j)

```
1: if (AODVi_[i]_free_token = 1) then
2:     Token ← False;
3:     Send Agreement (Token, Queue) to j;
4:     Modification in the routing table AODVi (0,0);
5: else
6:     if (AODVi_[i]_present_token = 1) then
7:         Next ← Next+{j};
8:         Queue ← Next;
9:     else
10:        Send Reject () to j;
11:    end if
12: end if
```

Following Algorithm 4, when a requesting site receives a reject message, it is going to look for another adequate site holding a free token by consulting its routing table.

**Algorithm 4** Evolutionary algorithm

```
1: Holder ← Find the adequate holder (AODVi);
2: Send Request (Token, i) to Holder;
```

When a site releases the critical section, it sends the token with the waiting queue to the first requesting site in that queue. If the waiting queue is empty, the token remains held by this site. After that, an update of the routing table must be done (see Algorithm 5).

**Algorithm 5** Evolutionary algorithm

```
Requesting ← False;
if (Not_empty (Next)) then
    j ← head(Next);
    Next ← Next – {j};
    Queue ← Next;
    Send Agreement (Token, Queue) to j;
    Token ← False;
else
    Modification in the routing table of AODVi (1,1);
end if
```

In Algorithm 6, the modification in the routing table concerns two variables: the existence of a free token, and the existence of a present token, according to the values of X and Y.

**Algorithm 6** Modification in the routing table AODVi (X,Y)

```
AODVi_[i]_Free_token ← X;
AODVi_[i]_present_token ← Y;
```

Procedure 7 presents the judicious manner that we use to find the closest token holder for the requesting site.

**Algorithm 7** Find the adequate token holder(AODVi)

```
X ← Y;
for Q = 1 to N do
    if (AODVi_[Q]_free_token = 1) then
        Min ← AODVi_[Q]_Distance;
        if Min < X then
            Holder ← Q;
            X ← Min;
        end if
    end if
end for
if (X = Y) then
    for Q = 1 to N do
        if (AODVi_[Q]_present_token = 1) then
            Min ← AODVi_[Q]_Distance;
            if Min < X then
                Holder ← Q;
                X ← Min;
            end if
        end if
    end for
end if
Return (Holder);
```

## 5 K-MUTEX with fault tolerance

In Ad hoc networks, any site may crash at any time for several reasons. This site may contain important information for the well-functioning of the system. In this case, this fault will cause degradation on the functioning of the whole system. Thus, a fault tolerance mechanism is needed to ensure that the system operates properly, even in the presence of faults. This is achieved by choosing backup sites that may take action when some sites crash. In our algorithm, the major problem is the loss of the token or the waiting queue. This loss is caused by the crash of a site holding these two information elements, and so we must ensure that these two data structures are not lost, and if they are lost, the system should be able to regenerate them.

### *5.1 Basic idea*

In our proposal, a double-layer fault-tolerance mechanism, is used to ensure that the system continues to function normally when the sites holding important information crash. Our idea consists in associating with every site holding a token another site called the safe site. This safe site keeps a copy of the waiting queue, and it can even generate a new token when the site holding the token crashes. This duplication allows the system to continue to work properly even after a loss because the critical information to guarantee its functioning is not lost thanks to the second copy held by the safe site. However, notice that the safe site is just another site, and so it may itself crash at any time. This will cause the information held in it to be lost, reason why we add a second layer of recovery to avoid such situations. Hence, for every safe site, we choose another site called the back-to-back site. This site keeps another copy of the waiting queue, and it can act as a safe site when the former crashes. Thus, using our fault tolerance mechanism, we have the site that holds the token and the waiting queue, and for this site a safe site is chosen, and a back-to-back site is chosen for every safe site. This way we can ensure that the important information will not be lost. The choice of these sites can be based on distance. In fact, the safe site and the back-to-back site must be the closest and most stable ones to minimise the number of hops, and also to avoid packets loss in the network. Safe and back-to-back sites are selected in such way that they are the ones having the highest link stability using the equation detailed below.

We consider a link between two sites as stable during a time $t_0$ if they are neighbours moving in the same direction, and with roughly the same velocity in the time interval $[t, t + t_0]$. The link stability between every pair of nodes must be revised periodically due to the nature of communications in mobile networks.

We calculate the link stability $LS_{i,j}$ between two nodes '$i$' and '$j$' as follows:

$$LS_{i,j} = \alpha \cdot LS_{i,j} + (1 - \alpha) \cdot \left[ \frac{1}{\frac{\Delta V_{i,j}(t+\rho)}{\Delta V_{i,j}(t)} \times \frac{D_{i,j}(t+\rho)}{D_{i,j}(t)}} \right] \tag{1}$$

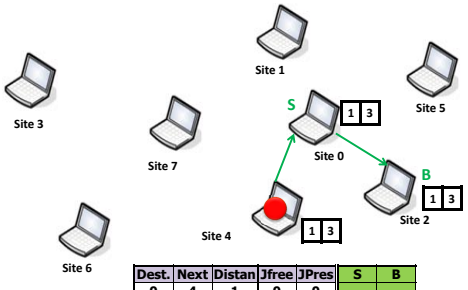
where

- $\alpha$: constant used to avoid the influence of peak cases, such as unexpected braking
- $V_i(t)$: velocity of node '$i$' at time $t$

- $\Delta V_{i,j}(t) = V_i(t) - V_j(t)$; '$i$' and '$j$' speed variations at instant $t$
- $D_{i,j}(t)$: distance between '$i$' and '$j$' at time $t$.

In addition, the selected site should not be used by another site holding token. Figure 2 illustrates the proposed message format extension to ensure the fault-tolerance.

**Figure 2** Proposed message format extension related to the fault-tolerance (see online version for colours)




| Dest. | Next | Distan | Jfree | JPres | S | B |
|---|---|---|---|---|---|---|
| 0 | 4 | 1 | 0 | 0 | - | - |
| 1 | 0 | 2 | 0 | 0 | - | - |
| 2 | 4 | 1 | 0 | 0 | - | - |
| 3 | 4 | 1 | 0 | 0 | - | - |
| 4 | 4 | 0 | 0 | 1 | 0 | 2 |
| 5 | 0 | 2 | 0 | 0 | - | - |
| 6 | 4 | 1 | 0 | 0 | - | - |
| 7 | 4 | 1 | 0 | 0 | - | - |

### *5.2 Fault situations*

We can find several cases of faults in our system, where the fault can occur at the level of:

- the token holder
- a safe site
- a back-to-back site
- both a site holding a token and its safe site at the same time
- both a site holding a token and a back-to-back site at the same time
- both a safe site and its back-to-back site at the same time.

We will now discuss the actions that must be taken for every situation.

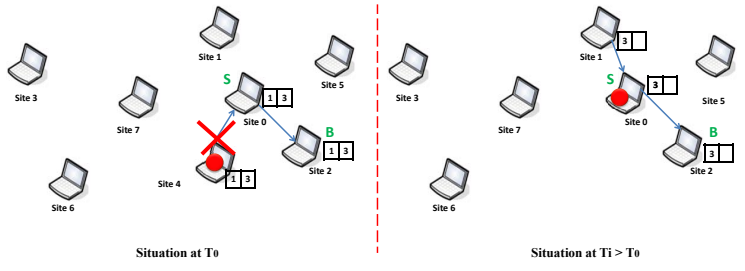
#### *5.2.1 A site holding a token*

When a failure of a site holding a token is detected by its safe site, the latter must act as a token-holding site by following these steps:

- generate a new token
- remove the identity of the failing site from the waiting queue
- send the new token with a copy of the waiting queue to the first requesting site in the queue
- propagate the information to its back-to-back site.

Figure 3 illustrates the tolerance strategy when the fault occurs at the token holder level.

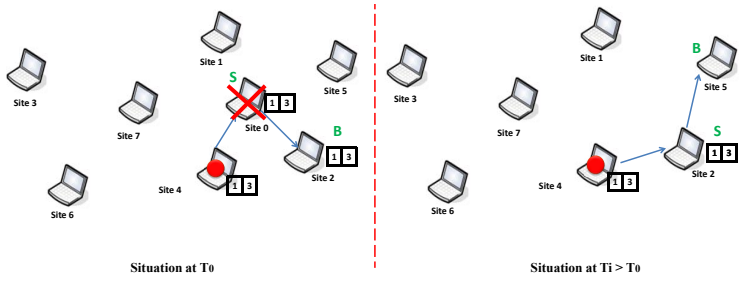
**Figure 3** Tolerance strategy when the fault occurs at the token holder level (see online version for colours)




### 5.2.2 *A safe site*

In this case, the back-to-back site becomes a safe site, and it chooses a new site to become its back-to-back site. The latter will receive a copy of the waiting queue from its safe site, as illustrated in Figure 4.

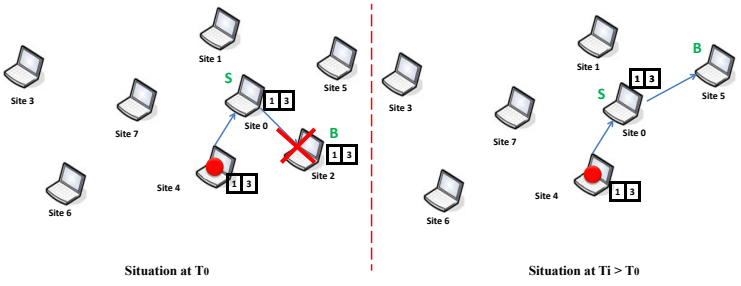
**Figure 4** The tolerance strategy when the fault occurs at the safe site level (see online version for colours)



### 5.2.3 *Back-to-back site*

The safe site chooses a new back-to-back site, and sends a copy of the waiting site to it, as illustrated in Figure 5.

**Figure 5** Tolerance strategy when the fault occurs at the back-to-back site level (see online version for colours)




### 5.2.4 *A site holding a token and its safe site at the same time*

The back-to-back site removes the identities of the failing sites from the queue if they are present in it, generating a new token, and then sending the token with a copy of the waiting queue to the first site in the queue.

**Figure 6** Tolerance strategy when the fault occurs at both the token holder and its safe site at the same time (see online version for colours)

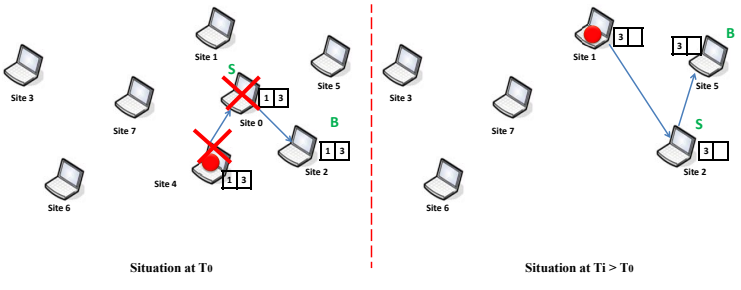



Figure 6 illustrates the tolerance strategy when the fault occurs at both the token holder and its safe site at the same time.

### 5.2.5 *Site holding a token and a back-to-back site at the same time*

The safe site removes the identities of the failing sites from the queue if they exist in it, generate a new token, and sending the token with a copy of the waiting queue to the first site in the queue. Afterward, it selects a new back-to-back site.

**Figure 7** The tolerance strategy when the fault occurs at a site holding a token and a back-to-back site at the same time (see online version for colours)

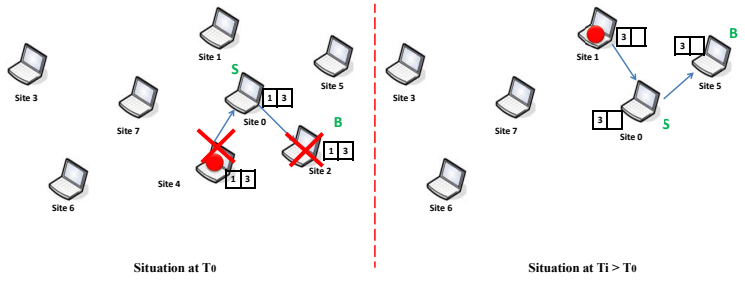



Figure 7 illustrates the tolerance strategy when the fault occurs at both a site holding a token and a back-to-back site at the same time.

### *5.2.6 A safe site and its back-to-back at the same time*

The site holding the token chooses a new safe site and sends a copy of the waiting queue to it. The latter will choose a site as its back-to-back site, and also sends a copy of the waiting queue.

**Figure 8** Tolerance strategy when the fault occurs simultaneously at a safe site and its back-to-back site (see online version for colours)

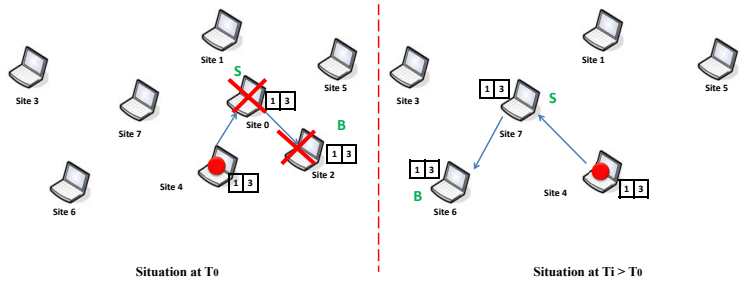



Figure 8 illustrates the tolerance strategy when the fault occurs at both a safe site and its back-to-back site at the same time.

## 6 Performance evaluation

To evaluate our proposal we relied on the NS-2 simulator (Issariyakul and Hossain, 2011) using the IEEE 802.11 standard for communications. Simulations run during 300 seconds, during which mobiles nodes move with a variable speed ranging from 0 to

8 m/s, within a 1 $km^2$ area using the Random WayPoint mobility model. Furthermore, tokens are held by randomly chosen 'k' nodes.

Table 2 summarises the main simulation parameters:

**Table 2** Simulation parameters

| *Scenario* | *Number of requests* | *Communications range (m)* | *Number of nodes* | *Number of resources* | *Number of faults* |
|---|---|---|---|---|---|
| 1 | [3, 18] | 200 | 50 | 5 | 4 |
| 2 | 10 | [50, 500] | 50 | 5 | 4 |
| 3 | 10 | 200 | [20, 50] | 5 | 4 |
| 4 | 10 | 200 | 50 | [3, 15] | 4 |
| 5 | 10 | 200 | 50 | 5 | [2, 10] |

We evaluated the performance of our solution on different scenarios by varying: number of requests, network density, communications range, and number of faults. In addition, for every scenario, we studied the amount of exchanged messages together with the generated delay. We also compared the waiting time and number exchanged messages of our proposal NFK to those obtained by CKM (Haghighat and Mohamadi, 2011), we chose to compare with CKM mainly because it was evaluated within the same environment as ours, and using the same routing strategy.

**Figure 9** Amount of exchanged messages with respect to the number of requests (see online version for colours)

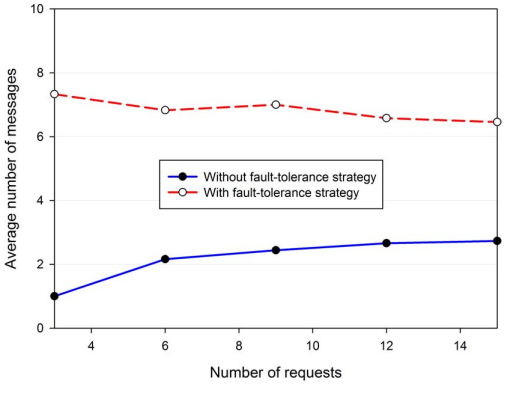



Figure 9 presents the amount of exchanged messages with respect to the number of requests both with and without fault-tolerance mechanisms. It is clear that the average number of messages is much higher for the algorithm with fault tolerance because of the different updates required to implement the required tolerance. However, this double-layer system achieved a similar delay despite the presence of faults, as illustrated in Figure 10.

In addition, in the first part, Figure 10 also depicts that the existence of a failure had a positive effect on certain sites, since if a token-holding site fails, the token is sent to the waiting requesters and then, in the second phase, they experience higher waiting times than the algorithm without tolerance, which is quite logical considering the time taken to resume operation after failure. Furthermore, Figure 10 shows also that thanks to the efficient AODV-based on-the-fly CS information sharing, our full proposal with fault tolerance takes almost the same waiting time as CKM which does not consider any fault tolerance mechanism.

**Figure 10** Average waiting time to enter the CS of NFK compared to CKM with respect to the number of requests (see online version for colours)

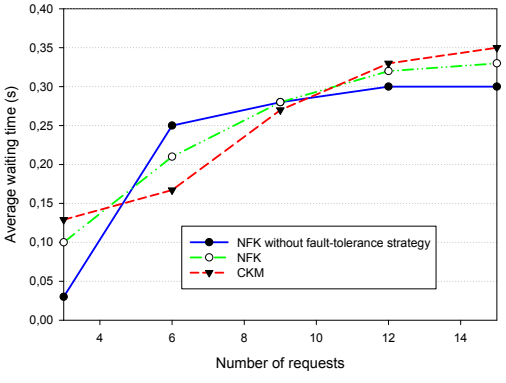
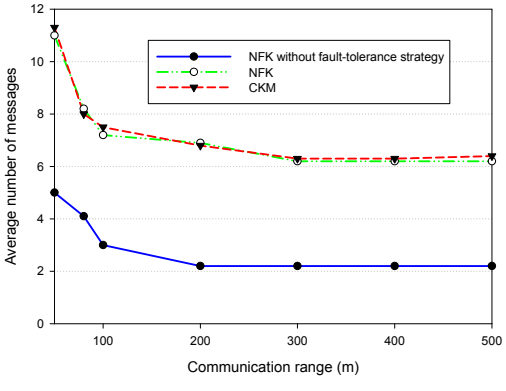



**Figure 11** Amount of messages exchanged of NFK compared to CKM with respect to the communications range (see online version for colours)



We can see from the variation of the communications range in Figure 11 that the amount of exchanged messages logically decreases when the communications range increases since, in this situation, end-to-end communications require fewer hops. Whereas, the cluster-based approach adopted by CKM without fault tolerance requires almost the same number of message only to ensure the K-MUTEX.

Furthermore, the existence of breakdowns has not affected the average waiting delay, as illustrated in Figure 12, thanks to our efficient and fast double-layer recovery strategy.

**Figure 12** Average waiting time to enter the CS with respect to the communications range (see online version for colours)

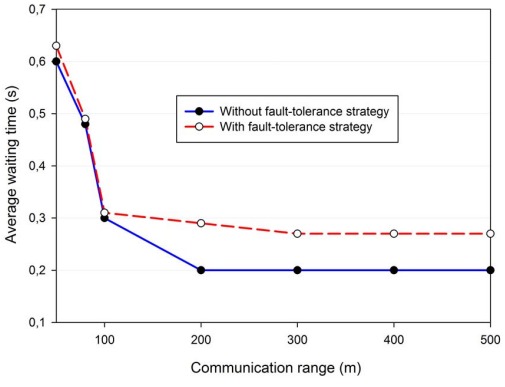



**Figure 13** Amount of exchanged messages with respect to network density (see online version for colours)

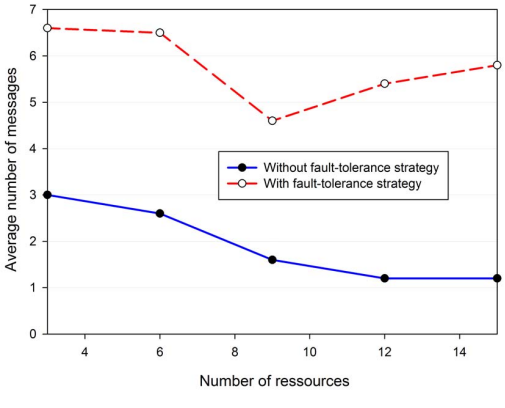


Figure 13 shows an increasing behaviour in terms of the number of exchanged messages in respect to the number of sites because of the number of updates to each failure. However, the waiting time for the fault tolerance algorithm also increases due to the immediate regeneration after a token holder site fails (see Figure 14). This occurs because the duration of the critical section is most influential parameter on the waiting time.

**Figure 14** Average waiting time to enter the CS with respect to network density (see online version for colours)

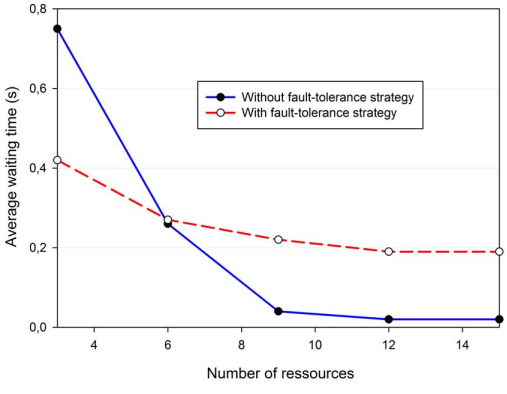
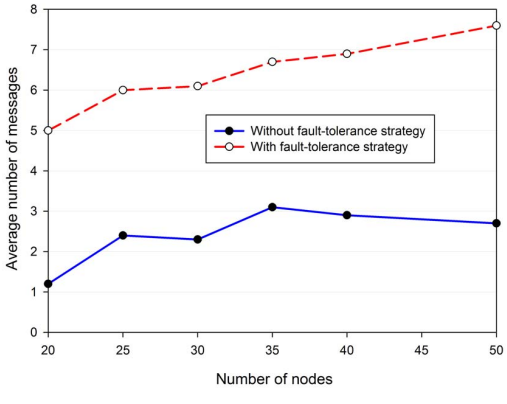



**Figure 15** Amount of exchanged messages with respect to the number of ressources (see online version for colours)



In Figure 15, the unstable behaviour of the curves has a direct relationship with the number of fails of sites acting as token holders, safe or back-to-back. Hence, the wait time is logically higher for the algorithm providing fault tolerance (see Figure 16). Also notice that, since we randomly generated the identities of the nodes where faults occur, multiple token holders can breakdown at the same time.

**Figure 16** Average waiting time to enter the CS with respect of the number of resources (see online version for colours)

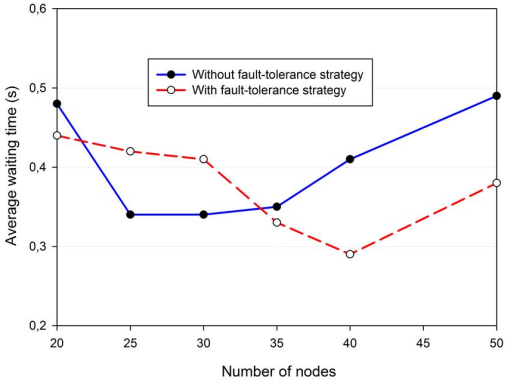
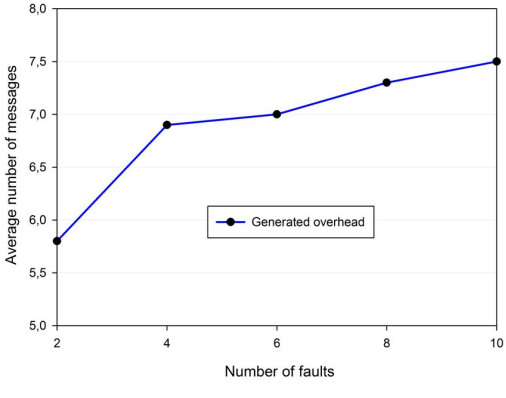
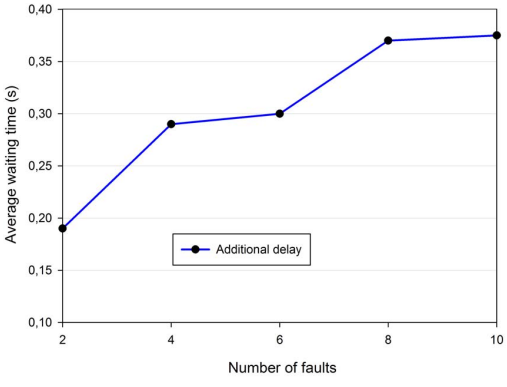



**Figure 17** Amount of exchanged messages in respect of the number of faults (see online version for colours)




From both Figures 17 and 18, we can see that the number of messages and the waiting time increase logically when increasing the number of faults due to the different fault-tolerance updates. However, our proposal is able to sustain low delays (<0.4 seconds in the worst case).

**Figure 18** Average waiting time to enter the CS with respect to the number of faults (see online version for colours)




## 7 Conclusions and future work

In this paper, we presented a new K-MUTEX algorithm for mobile ad hoc networks. We used a new method that can be considered as an innovation in this domain. In particular, our method integrates the information concerning the availability of tokens in the routing tables in order to minimise the number of exchanged messages.

We used a fault tolerant mechanism based on the duplication of information to resist token losses in our system. This dual-layer mechanism overcomes token loss, and permits the well functioning of the system even when several sites fail at the same time. Our study of fault tolerance was limited to the token loss. Other situations of failure include message losses and links failures caused by the network partitioning. These issues will be addressed in future works. Simulation results show that our algorithm guarantees the accesses to the CS while exchanging a reduced number of messages, and introducing a very short waiting time.

Future works may involve the use of the proposed principle and the proposed fault tolerant mechanism in other systems such as the vehicle networks and UAV networks.

## References

Bulgannawar, S. and Vaidya, N.H. (1995) 'A distributed k-mutual exclusion algorithm', *Proceedings of the 15th International Conference on Distributed Computing Systems*, pp.153–160, IEEE.

Chakeres, I.D. and Belding-Royer, E.M. (2004) 'AODV routing protocol implementation design', *Proceedings: 24th International Conference on Distributed Computing Systems Workshops 2004*, pp.698–703, IEEE.

Dagdeviren, O. and Erciyes, K. (2007) 'A software architecture for shared resource management in mobile ad hoc networks', *SOFSEM 2007: Theory and Practice of Computer Science*, pp.224–234, Springer.

Erciyes, K. (2004) 'Cluster based distributed mutual exclusion algorithms for mobile networks', *Euro-Par 2004 Parallel Processing*, pp.933–940, Springer.

Haghighat, A.T. and Mohamadi, M.R. (2011) 'Cluster-based k mutual exclusion for mobile ad hoc networks', *2011 5th International Conference on Application of Information and Communication Technologies (AICT)*.

Issariyakul, T. and Hossain, E. (2011) *Introduction to Network Simulator NS2*, Springer Science & Business Media, Springer Publishing Company, Salmon Tower Building, New York, USA.

Masum, S.M., Akbar, M.M., Ali, A.A. and Rahman, M.A. (2010) 'A consensus-based ℓ-exclusion algorithm for mobile ad hoc networks', *Ad Hoc Networks*, Vol. 8, No. 1, pp.30–45.

Moallemi, M., Moghaddam, M.H.Y. and Naghibzadeh, M. (2007) 'A fault-tolerant mutual exclusion resource reservation protocol for clustered mobile ad hoc networks', *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007*, Vol. 2, pp.528–533, IEEE.

Nasipuri, A. (2004) 'Mobile ad hoc networks', *Handbook of RF and Wireless Technologies*, pp.59–100, Elsevier, USA.

Prabhu, B., Gajendran, E. and Balakumar, N. (2017) 'Smart oil field management using wireless communication techniques', *IJIEST 2016*, January–December, Vol. 2, SSRN [online] https://ssrn.com/abstract=289563.

Rappaport, T.S. et al. *Wireless Communications: Principles and Practice*, Vol. 2, Prentice Hall PTR, New Jersey.

Raymond, K. (1989) 'A tree-based algorithm for distributed mutual exclusion', *ACM Transactions on Computer Systems (TOCS)*, Vol. 7, No. 1, pp.61–77.

Saxena, P.C. and Rai, J. (2003) 'A survey of permission-based distributed mutual exclusion algorithms', *Computer Standards & Interfaces*, Vol. 25, No. 2, pp.159–181.

Swaroop, A. and Singh, A.K. (2007) 'A study of token-based algorithms for distributed mutual exclusion', *International Review on Computers and Software*, Vol. 2, No. 4, pp.302–311.

Tamhane, S.A. and Kumar, M. (2012) 'A token based distributed algorithm for supporting mutual exclusion in opportunistic networks', *Pervasive and Mobile Computing*, Vol. 8, No. 5, pp.795–809.

Thiare, O. and Naimi, M. (2009) 'A group k-mutual exclusion algorithm for mobile ad hoc networks', *10th International Work-Conference on Artificial Neural Networks, IWANN 2009 Workshops, Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, Salamanca, Spain, pp.58–66, Springer.

Trehel, M. and Naimi, M. (1987) 'A distributed algorithm for mutual exclusion based on data structures and fault tolerance', *Proc. IEEE 6th International Conference on Computers and Communications*, pp.35–39.

Walter, J., Cao, G. and Mohanty, M. (2001) 'A k-mutual exclusion algorithm for wireless ad hoc networks', *Proceedings of the First Annual Workshop on Principles of Mobile Computing*, Citeseer.

Wu, W., Cao, J. and Raynal, M. (2007) 'A dual-token-based fault tolerant mutual exclusion algorithm for manets', *Third International Conference, MSN 2007, Mobile Ad-Hoc and Sensor Networks*, Beijing, China, pp.572–583, Springer.

Wu, W., Cao, J. and Yang, J. (2008) 'A fault tolerant mutual exclusion algorithm for mobile ad hoc networks', *Pervasive and Mobile Computing*, Vol. 4, No. 1, pp.139–160.