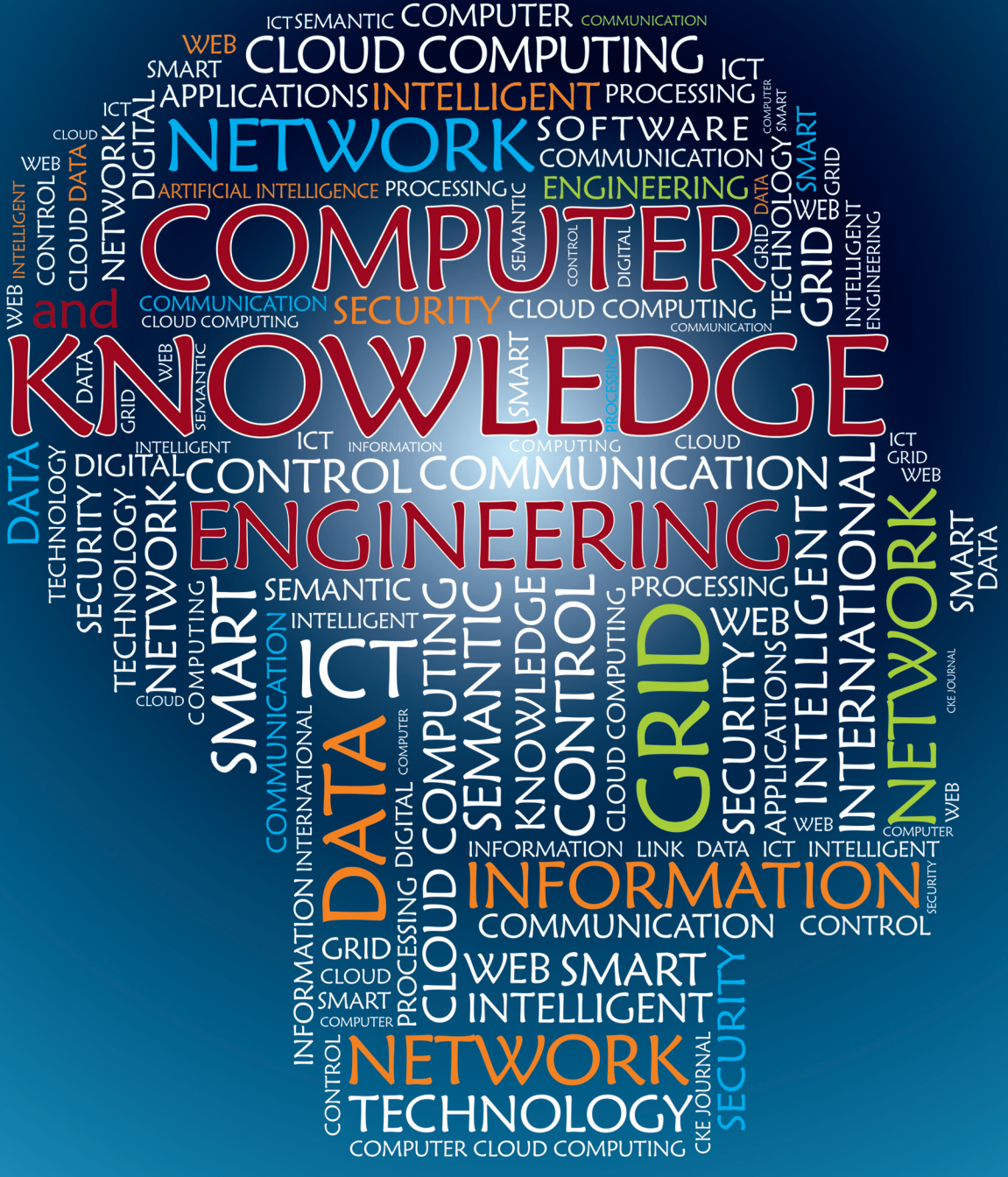
FACULTY OF ENGINEERING

Journal of Computer and Knowledge Engineering

Vol. 1, No.2. 2017, ISSN : 5238-5453



Serial No. 2

Journal of
COMPUTER AND KNOWLEDGE ENGINEERING

Ferdowsi University of Mashhad

ISSN:5238-5453

Journal of Computer and Knowledge Engineering
Faculty of Engineering, Ferdowsi University of Mashhad
P. O. Box. 91775-1111, Mashhad, I.R. IRAN
Tel: +98 51 38806024, Fax: +98 51 38763301, Email: ejour@um.ac.ir, Site: cke.um.ac.ir

CONTENTS

Journal of Computer and Knowledge Engineering, Vol. 1, No.2. 2017.
DOI: 10.22067/cke.v1i2.57729

Migration Management in Sensor-Cloud Networks

Farahnaz Farazestanian Seyed Amin Hosseini Seno*

Abstract: Placement of virtual sensors in servers of cloud environment is one of the most important issues in resource management of sensor-cloud networks. Virtual sensors allocate resources of cloud servers to themselves, run different applications and use equivalent physical sensors. Selecting an appropriate method for placement of virtual sensors on cloud servers has a significant impact on resource management and energy consumption. Also, because of the real-time requirements of sensor-cloud networks, traffic-aware placement is necessary to minimize delay in a network and respond to user requests in the shortest possible time. A new method for traffic-aware placement and migration of virtual sensors in sensor-cloud networks is proposed using the unique characteristics of sensor-cloud networks such as grouping virtual sensors and sharing them among different applications. This approach tries to meet the needs of users and the quality of service expected by the applications. Also, the resources of cloud servers are managed properly in this approach. The results of simulation show an optimization in energy consumption and also a reduction in traffic costs and reduction in service level agreement violation.

Keywords: Sensor-cloud networks, virtual sensors, resource management, migration, energy consumption, traffic.

1. Introduction

Sensor-cloud networks are proposed to overcome limitations of sensor networks and provide new services for users as a new generation of cloud computing. These networks benefit from cloud infrastructure for sharing physical sensors among different users. Also, some problems of sensor networks such as the limitation of data storage and the processing of data are eliminated [1-3].

Sharing physical sensors among different applications is performed by virtual sensors. Each virtual sensor is a software sensor, which is created as an intermediary for the connection between physical sensors and applications [4, 5]. Each virtual sensor is related to one or several physical sensors that can aggregate their data and perform some calculations on them. Virtual sensors can communicate with each other and form a virtual sensor group. It is also possible to share virtual sensors between different applications [6]. Virtual sensors allocate resources of cloud servers to themselves, including CPU, memory, storage, and bandwidth. They can respond to various requests of users. Virtualization technology enables the creation of virtual sensors in virtual machines of cloud servers. This technology uses different placement methods of virtual machines to manage allocation of cloud server's resources.

A review of existing literature in the field of sensor-cloud networks shows that the placement of virtual sensors on cloud servers has been neglected by the designers of these networks. In the available approaches, there is no mention of the impact of migration on meeting real-time needs of these networks. So, it seems that a new approach for migrating virtual sensors can have a significant impact on energy saving and network traffic and also meeting the quality of service expected by the applications.

2. Background

One of the challenging issues in data centers of clouds is the placement of virtual machines in cloud servers. Many approaches have been proposed to solve this problem. These approaches are meant to meet different objectives, such as increasing the efficiency of server resources, reduce energy consumption, reduce network traffic, or meet the quality of service (QoS) expected by the applications. However, it seems that this issue has not been addressed yet in sensor-cloud networks, and no method has been presented for optimization of this problem. What has been stated in sensor-cloud networks is a general case of creating virtual sensors on cloud servers [6]. In this case, after receiving requests of (from) users for using a special physical sensor, the sensor-cloud infrastructure retrieves the related template of that physical sensor from the repository. Then, it tries to provide the virtual sensor on the existing server. If the existing server is not capable of meeting the resource requirements of the virtual sensor, a new server is selected. In this paper, virtual sensor placement is not considered as an optimization problem, and there is no preference for selecting servers running virtual sensors. Also, in other research studies such as M. Yuriyama's work [3], general methods have been used and virtual sensor placement is not an independent issue. Therefore, the allocation of virtual machines in the cloud environment are studied in order to present a new method for placement of virtual sensors on sensor-cloud networks.

One of the proposed solutions for the virtual machine placement problem in a cloud environment is considering the available resources and capacities on cloud servers. Power consumption in the data center is related to processor, memory, storage and network interface. Among these, the processor has the highest proportion of power consumption. Therefore, optimizing the utilization of the processor is an important factor in the virtual machine placement problem [7]. The capacity limitations of the resources that are mentioned as server constraints, means that total resources allocated to all virtual machines running on a server cannot be more than the total capacity of that server [8, 9]. If the

Manuscript received July 24 , 2016; accepted. November 6, 2017.
F. Farazestanian, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran, fa.farazestanian@stu.um.ac.ir.
*S. A. Hosseini Seno, Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran, hosseini@um.ac.ir.

constraints on a server are not met and the needed resources are not fully allocated to virtual sensors, virtual machine performance may be affected, and the expected quality of service is not achieved.

Migration that is one of the technologies in the cloud environment is the solution for distributing workloads on cloud servers. It helps avoid overloading the servers to maintain proper performance. Live migration of virtual machines is used to move virtual machines from one server to another during the running time and to respond to their workload changes [10, 11]. Xen is a virtual machine monitor (VMM) that manages and monitors the migrations of virtual machines on data centers [12]. This VMM is used in the simulation of the proposed algorithm as the hypervisor in cloud servers.

In multi-tier applications where each part of the application is executed on a separate virtual machine, dependencies between virtual machines are considered as an optimization parameter, which can lead to traffic on the network [8]. The algorithm in [13, 14] aims to migrate virtual machines so that network traffic is minimized and server-side constraints are satisfied. This algorithm considers a dependency graph between dependent virtual machines. The cost of migration is calculated as a function of traffic demand between each pair of dependent virtual machines and the distance between them. Distance is defined as the latency, delay or number of hops between each pair.

D. S. Dias et al. [15] have considered traffic dependencies between virtual machines. The algorithm tries to find dependent virtual machines by using the concept of graph community. In this method, virtual machines are considered as nodes of a graph and the dependencies between them are depicted by edges. The edge weight is used to determine the degree of relationship between a virtual machine and other virtual machines. This weight shows the level of traffic between each pair of virtual machines. Thus, with more dependency between virtual machines, a greater weight will be expected. After creating traffic matrix and identifying virtual machine communities [16], each community must be placed in one section of the servers called racks. Each rack contains a number of servers which are connected to one switch. The purpose is to place dependent virtual machines as close as possible to each other. The solution of this problem involves the amount of CPU and memory required by each community and the available resources in each rack. This problem is defined as a bin packing problem. H. T. Vu et Al. in [17] have tried to simultaneously reduce network traffic and energy consumption and increase the utilization of the processor by developing this algorithm. In this method, a tree is created such that in its lowest level, dependent virtual machines are placed as sibling nodes after finding dependent virtual machines. The host of the node which is selected for migration may be under-utilized or over-utilized. According to this, the destination is the host where the increase of energy consumption is minimum. Also the total distance between the destination host and dependent virtual machines should be minimum.

One of the proposed methods for making decisions about the start time of a migration is using thresholds according to CPU utilization. In the double-threshold method, the amount of CPU utilization is bounded between the upper and lower thresholds. Exceeding these amounts will force the server to execute migration [7]. If CPU utilization violates the low threshold, all the virtual machines on the server are forced to migrate and then the server will shut down. Also, if the CPU utilization exceeds the high threshold, the server should select one virtual machine for migration. Migrating the virtual machine will cause reduction in the CPU's workload. Consequently performance of the applications will also be managed in addition to reduction in energy consumption. Since the workload of applications is continuously changing, the CPU utilization also has continuous changes over time. So, the placement of virtual machines must be constantly optimized. Several heuristic algorithms have been proposed by A. Beloglazov et al. [18] with the ability to adapt to these changes. These algorithms are designed to optimize power consumption, but they do not consider traffic dependencies between virtual machines. In the method proposed in this study, these algorithms are used to select a virtual machine for migration.

Consolidation of virtual machines on a single server is also considered in some papers. Inappropriate consolidation of heterogeneous virtual machines on cloud servers will not only affects computing performance of applications, but it also reduces energy efficiency resulting in more energy waste [19]. Drop in efficiency also leads to violation of service level agreement (SLA). Placing one virtual machine on different servers can cause different energy efficiency levels due to the unique characteristics of the virtual machine. Because of the different effects of consolidations of virtual machines on a single server, implementing virtual machines to servers will have more complexity. Due to the characteristics of consolidated virtual machines, power consumption and overall efficiency will change. This is because of internal conflicts among consolidated virtual machines, such as cache contentions, conflicts at functional units of the CPU, disk scheduling conflicts, and network transfer conflicts. M. F. H. Bhuiyan et al. [20] have shown that a blind consolidation of virtual machines on a server will lead to reduced power consumption and energy loss. The idea of the researchers in [21] is placing virtual machines with the same dominant resources on different servers at the best effort. In this case, the resource competition between different virtual machines on the same server is significantly reduced. At the initial placement of virtual machines on servers, virtual machines are classified according to the dominant resource consumption. Then they are arranged in descending order according to their priorities. In each category, the virtual machine with the highest priority is assigned to the server with the greatest resource available of the category's type. Since the migration of virtual machines reduces efficiency, the machine that has a lower priority will be selected for migration.

In the placement of virtual machines in the cloud environment and the methods in the field of network traffic in the cloud, it is seen that the network traffic is considered between virtual machines and relevant applications or data transfer between data centers and cloud servers. However, in sensor-cloud networks the traffic can occur because of the characteristics of the network. The possibility of creating virtual sensors and virtual sensor groups has many benefits and simultaneously, the availability of new features for users of sensor-cloud networks, may cause unwanted effects on the performance of the network. When virtual sensors participate in a virtual sensor group, data communication among the members of that group will happen. According to

the data collection rate of the associated physical sensors, the traffic load among the group members may change. In addition, if the locations of the servers of a group are far from each other, traffic between the servers will be imposed on network and traffic overhead can ensue. As a result, it may fail in responding to user requests and sending data to the applications. On the other hand, sharing virtual sensors between different applications will add the number of virtual sensor groups continuously, which increases the importance of this issue.

Energy consumption by servers of the cloud is also a significant problem. Many factors that should be considered can affect this and a suitable approach to optimize the energy consumption should be adopted. Placement of virtual sensors on cloud servers, as well as how to choose the host servers can be one of the factors. Also, using virtualization in cloud environments offers another feature called migration that can be used to perform this optimization.

Another challenging issue in cloud environments is resource management. Distribution of virtual machines on the cloud servers is one of the most complex issues in recent years, and various methods have been proposed to solve it. Meeting the quality of service expected by the applications depends on efficient allocation of resources needed by them in the running time. It has a significant impact on the quality of applications. Also, an inappropriate combination of virtual machines on a server can cause resource competition and reduce server performance. Because different combinations of virtual machines on a server have a great impact on energy consumption, the homogeneity of virtual machines on a server should be also considered in the placement process.

The main objective of the proposed method is introducing a traffic-aware algorithm for placement of virtual sensors by using live migration of virtual machines to distribute the workload. This is more important because of the grouping feature of virtual sensors. Resource management is taken into consideration with regard to energy efficiency. This approach should provide acceptable performance and hence it is expected that a service level agreement shall be fulfilled.

3. The proposed method

The method proposed for traffic-aware migration of virtual sensors in sensor-cloud networks in this study is looking for a suitable destination for migration. In this method, the virtual sensors are classified based on their dominant usage of resources, including processor, memory, and network bandwidth. When finding a suitable destination for migration, virtual sensors with the same dominant resource usage are distributed on different servers. Moreover, the destination of migration during a replacement of virtual sensors is selected according to virtual sensor groups. In this case, the destination server is selected as close as possible to the servers of dependent virtual sensors. Also, the destination server is selected based on the minimum energy increase after the placement. Assumptions necessary for the implementation of this method are described in section 4.1 and details of this method are explained in Section 4.2.

3-1. Assumptions and Limitations

In the proposed method, we assume that each physical sensor is associated with the cloud via the related virtual sensor. Each virtual sensor is running on a virtual machine on cloud servers. In this method, host servers of applications and host servers of virtual sensors are considered to be separate. The proposed algorithm tries to find an optimized allocation of virtual sensors on cloud servers and does not take into account the allocation of applications.

The proposed allocation method is designed by considering the two main characteristics of sensor-cloud networks. These features include the ability to create virtual sensor groups and meeting the needs of real-time data. Different types of data communication between the physical and virtual sensors are based on S. Madria's research [22, 23], including one-to-one, one-to-many and many-to-many communication between physical and virtual sensors. Based on the possibility of grouping virtual sensors, data communication and dependencies between virtual sensors of a group are used as parameters in the proposed method [6]. It is assumed that the virtual sensors will have no data communication with each other, except in the case of placement in a virtual sensor group.

In the proposed method, the data required by different applications are considered only in real-time situations, and the data stored in the data centers are not considered. Real-time requirements include traffic control to minimize the response time to the applications.

The initial placement of virtual sensors is based on M. Yuriyama's work [6]. When a client request is sent to the cloud environment for using a special physical sensor, the user will be assigned to that virtual sensor if a virtual sensor has already been created for that physical sensor. Otherwise, sensor-cloud infrastructure tries to create a new virtual sensor for each physical sensor based on predefined templates. To this, server resources should be reserved for the virtual machine which is running the virtual sensor. In this method, only three resource centers that include CPU, memory, and network bandwidth are exploited for each virtual sensor.

Each virtual sensor may dominantly benefit from one of the available resource centers. For example, a virtual sensor in a group may mostly need the processor to perform data integration and necessary calculations. Also, the virtual sensor whose equivalent physical sensor samples high volume data from the environment (e.g. video data) may need more memory. Physical sensors with a high sampling rate can also be an example of equivalent virtual sensors that are using a lot of bandwidth. Each virtual sensor is placed in one of the groups of dominant processor, memory, or network consumption, according to its dominant resource consumption. This classification can be obtained by monitoring the resource usage of virtual machines on a server [21]. According to the description of virtual sensor in S. Kabadayi's work [4, 5], the data type and sampling rate are determined by the user at the time of creating a new virtual sensor. These two parameters can also be used in the classification.

3-2. Migration of virtual machines

For optimization of the network status, cloud servers must use migration algorithms. Each migration algorithm should answer two questions: 1) which server should perform migration? And which virtual sensor should be migrated? In addition, 2) which server will be the destination of the migration? The main objective of the proposed algorithm here is to answer the second question. To address the first

question, the algorithms presented by A. Beloglazov et el. [18] are used.

3-2-1. Selecting server and a virtual machine for migration

An important factor that affects the efficiency offered by cloud servers is the workload of virtual machines running on them. Among server resources, the CPU has the most significant impact on the performance of the virtual machines. If the workload on a server exceeds its acceptable level, the server will not be able to meet the needs of virtual machines and virtual machines will lose their quality. On the other hand, if the workload is too low on the server, the server resources will be wasted. This causes energy loss. So, optimizing CPU usage level is very important and effective, in this regard.

There are two general approaches to identify overloaded hosts: 1) Adaptive utilization threshold based algorithm, 2) non-threshold based algorithm. After detection of overloaded hosts, the virtual machine for migration should be selected. A number of algorithms for detecting overloaded servers and selecting the appropriate virtual machine for migration are proposed by A. Beloglazov et al. [18]. These algorithms belong to the second category. Based on the results of the simulations and evaluations that have been done in this current work, the LrMmt algorithm has the best result in energy savings and SLA. This algorithm includes two local regression algorithms (Lr), one used to select the overloaded host and another to calculate the minimum migration time (Mmt) for selecting the virtual machine for migration. Due to the desirable results produced by this algorithm, it is used in the proposed method in this study.

To identify under-loaded servers, a simple method is used. In this way, after finding overloaded servers and migration destinations, the remaining servers are arranged in order of CPU utilization and the server with the lowest utilization is selected. This server tries to migrate all virtual machines running on it, while the destination servers do not experience overloading. After migration is complete, the server is set to switch to sleep mode.

After detecting over loaded and under loaded servers, virtual machines running virtual sensors are selected for migration. The destination server of migration is selected based on the allocation algorithm that will be presented in the next section. The pseudo code of this method is shown in Algorithm 1.

Algorithm 1: Sensor-Cloud VM Selection Algorithm

- **Given:**

H Available Host List
VMsToMigrate VMs that are selected for migration

- **Algorithm:**

```
1: for each host h in H
2:     if isHostOverLoaded (LrAlgorithm, h)
3:         VMsToMigrate.add (getVMsToMigrateFromOverLoadedHost (MmtAlgorithm, h))
4: DoMigration (SensorCloudVMAllocationAlgorithm, VMsToMigrate)
5: VMsToMigrate.clear()
6: for each host h in H
7:     if isHostUnderLoaded (UnderLoadDetectionAlgorithm, h)
8:         VMsToMigrate.add (h.getVMs())
9: DoMigration (SensorCloudVMAllocationAlgorithm, VMsToMigrate)
```

3-2-2. Allocation of virtual machines

By selecting the appropriate virtual machine for migration, the destination server must be determined. To do this, the available resources of the selected destination server should meet the resource requirements of the virtual machine which is being migrated. Resource usage of virtual sensors depends on the type and amount of data collected by the associated physical sensor. The virtual sensors are classified based on their dominant usage of CPU, memory, and the network.

The main part of the proposed method for selecting the optimal destination considers two issues: the amount of traffic on the network caused by virtual sensor groups, and how to consolidate heterogeneous virtual sensors on a single server. In the following section each of these cases will be studied separately.

3-2-2-1. Destination host selection based on virtual sensor groups

The destination server should be selected so that the dependencies between dependent virtual sensors are considered and the traffic between them is reduced. Data communication between virtual sensors only observes in virtual sensor groups. If the virtual sensors of a group are far from each other, data transfer between them causes high traffic overhead on the network. Consequently, when placing virtual sensors of a group, the distance between them must be reduced as much as possible. To do this, an algorithm for detecting dependent virtual sensors is needed.

According to the method presented by H. T. Vu et al. [17], the traffic between virtual sensors is modeled by a complete graph. The virtual sensors are vertices and network communications are edges. Edge weights are the traffic weight between the virtual sensors. Then, lowest weight edges are removed recursively and this continues until dependent virtual sensor groups with the high amount of traffic remain as a set of isolated sub-graphs. From this graph a tree structure is created in which sibling nodes at lower levels are virtual sensors of a group (Figure 1). For selecting the destination server for migration, if the migrating virtual sensor has a sibling node in the tree, the location of the server is chosen such that the total distance of the server to the servers of dependent virtual sensors is minimum. In other words, the destination server must be the closest possible server to the servers of dependent virtual sensors.

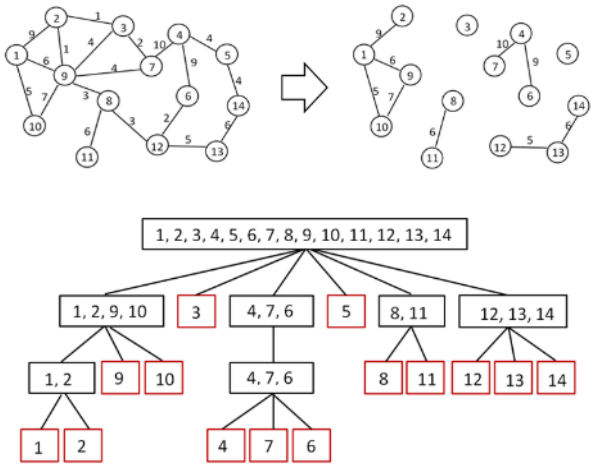


Fig 1. A conversion of graph to tree structure [17]

3-2-2-2. Destination host selection based on dominant resource consumption

As mentioned in Section 2, the consolidation of heterogeneous virtual machines on the same server has a high impact on performance and energy consumption. Replacement should be done such that the virtual sensors with the same dominant resource consumption are distributed on different servers as much as possible. In other words, these virtual sensors should not be placed on the same server. Consequently, conflicts and competitions for accessing the resources of servers will be reduced. By increasing the efficiency of virtual sensors, better service quality will be provided for applications. To achieve this, the destination server should be selected in such a way that the category of the migrating virtual machine's dominant consumption is different from the category of virtual machines' dominant consumption running on the destination server.

Moreover, for power management, the destination server should be selected such that the increase in energy consumption after the migration is minimum. The pseudo-code of the proposed allocation algorithm is represented in Algorithm 2.

Algorithm 2: Sensor-Cloud VM Allocation Algorithm

- **Given:**

H	Available Host List
V	migrating vm
allocatedHost	destination host for migrating vm

- **Algorithm:**

```
1:  for each host h in H
2:      if (h.consumptionType == v.consumptionType) then continue
3:      if ( h.isSuitableForVm(v) )
4:          if ( v.hasSibling() )
5:              if (h.TotalDistanceToSiblings( v.getSiblings() ) is minimum)
6:                  allocatedHost = h
7:          else
8:              if (h.EnergyIncreseAfterAllocation() is minimum)
9:                  allocatedHost = h
10: return h
```

4. Evaluation

For simulating the proposed algorithm and evaluating the energy consumption and traffic, the CloudSim toolkit [24] is used. It is an open source simulator for cloud computing environments, and it performs the modeling of virtual environments, on demand resource management and energy-aware simulations. Because of the absence of a suitable simulator for sensor-cloud networks, this tool is used.

The architecture considered for the data center is three-tier architecture. According to Cisco, today this architecture is mostly used in data centers in the cloud [25, 26]. Figure 2 illustrates this architecture. In the lowest layer of this architecture, the hosts are divided into partitions, and each partition is connected to an edge switch. Data communication between servers is via edge switches and data are not transferred to the higher layers. In the middle layer, each edge switch is connected to one or more aggregation switches. Then in the top layer, the core switches are connected to aggregation switches and also communicate with the rest of the data center. In the simulation of the proposed algorithm, this architecture is used for the data center of the cloud environment. The data center in the proposed data center includes 1 core switch, 3 aggregation switches, and 5 edge switches. Each edge switch is connected to 10 servers. Totally, the data center has 150 servers. The number of virtual machines equivalent to virtual sensors starts from 250 devices and in each round of simulation 10 devices are added to it. This amount goes up to 350 devices.

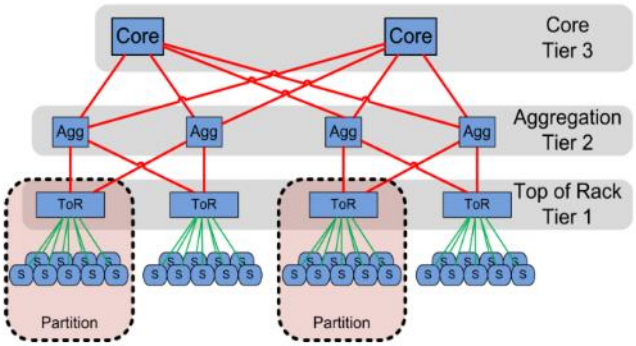


Fig 2. Three tier architecture for data centers [15]

The configuration of servers and virtual machines is according to the work of A. Beloglazov et al. [18]. The traffic in the data center is generated using FNSS tool. The run time of simulation is 20 minutes in each round. According to the generated traffic in the network, virtual machines are divided into three categories: CPU intensive, memory intensive, and network intensive. The servers are divided based on the number of virtual machines of each category running on them.

To select the appropriate virtual machine for migration, the algorithms presented by D. S. Dias et al. [15] are used. Since server selection and virtual sensors migration are done randomly in sensor-cloud networks, the random selection algorithm (LrRs) is used to simulate the migration in sensor-cloud networks [16]. Based on the results of the algorithms presented by D. S. Dias et al. [15], the LrMmt is nominated as the best algorithm for optimizing the energy consumption. This is combined with the proposed traffic-aware allocation method and is executed as the ScLrMmt algorithm. The evaluation parameters include: energy consumption, traffic costs, SLA violation, and the number of migrations.

4-1. Power consumption model

The main power consuming devices in data centers are processors, memory, hard disk, power generators and cooling systems. Recent studies [20] have shown that there is a linear relationship between the power consumption of servers and processor efficiency. Because of the complexity of power consumption modeling, especially in the new multi-core processors, instead of using an analytical model for the power consumption of the servers, the actual data released by the SPECpower benchmark is used for evaluating the power consumption level of servers.

The power consumption of the two servers, HP ProLiant G4 (http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00338.html) and HP ProLiant G5 (http://www.spec.org/power_ssj2008/results/res2011q1/power_ssj2008-20110124-00339.html), based on the percentage of CPU utilization is shown in the Table 1. These two types of servers are used in the simulation here.

Table 1. Power consumption of servers based on CPU utilization

CPU Utilization	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
HP ProLiant G4 Power Consumption (Watts)	86	89.4	92.6	96	99.5	102	106	108	112	114	117
HP ProLiant G4 Power Consumption (Watts)	93.7	97	101	105	110	116	121	125	129	133	135

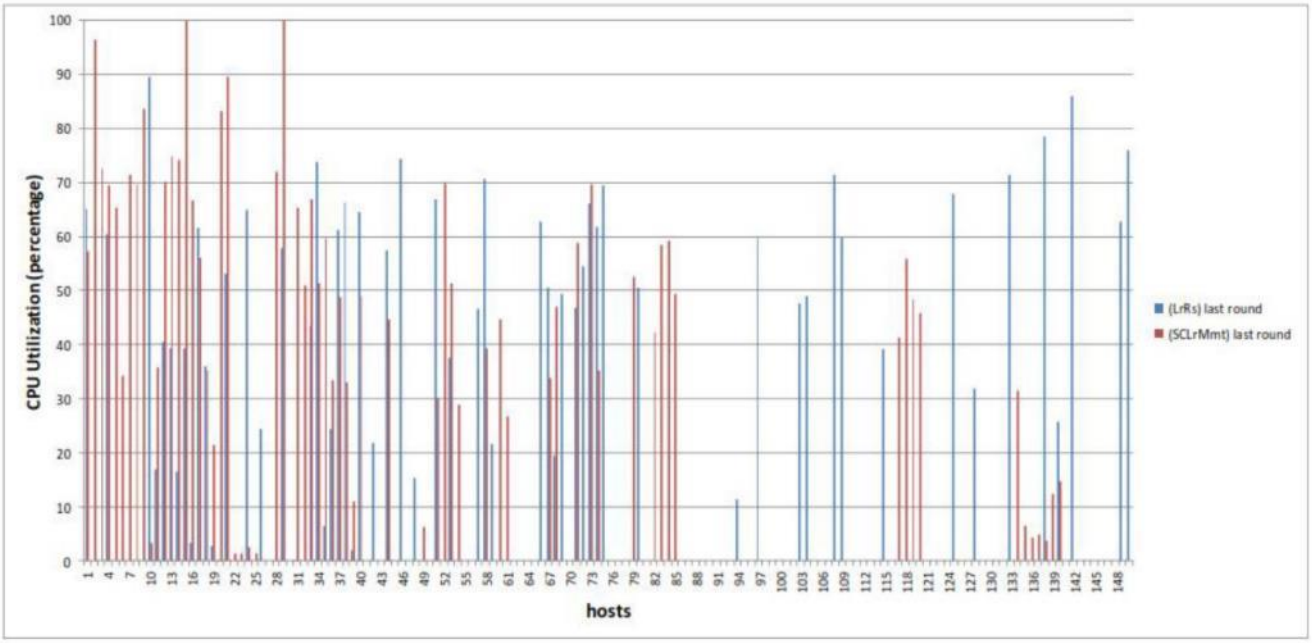


Fig 3. Average CPU utilization of servers

4-2. SLA Violation

Meeting QoS parameters is one of the most important issues in cloud environments. Requirements of quality of service are usually formulated as service level agreements and based on the characteristics like minimum throughput or maximum response time. Because in sensor-cloud networks, different applications use virtual sensors at the same time, SLA parameters must be defined independently from applications and their workloads. In assessment of the newly presented algorithm, SLA is calculated according to the P. Barham et al. proposed algorithm [18]. Thereby, the two parameters SLATAH and PDM are calculated first. Then, the following relationship is used to measure the service level agreement violation (SLAV) by combining these two parameters as follows:

SLAV=SLATAH . PDM

4-3. Simulation results

To evaluate the results of the proposed algorithm (SCLrMmt) and comparing it with the basic algorithm (LrRs), initially the impact of these algorithms on CPU utilization of cloud servers will be discussed. Among all the resource centers on a server, the CPU has a significant share of power consumption [18]. There is a direct relationship between CPU utilization and energy; i.e. a higher CPU utilization level causes more energy consumption. In addition, an increase in utilization leads to an increase in SLA violation since in higher utilization levels, virtual machine's access to the server processor will be restricted and competition for access will ensue. On the other hand, the power consumption of an idle server is about 70% of its power consumption in full usage state [27, 28]. Therefore, a suitable CPU utilization level is always desirable.

To check the CPU utilization of servers in both algorithms, simulations were run for 20 minutes with 150 servers and 350 virtual machines. The average of CPU utilization of servers during 12 rounds of simulation is illustrated in Figure 3. The proposed algorithm keeps the CPU utilization at a more appropriate level in comparison with the base algorithm. The average value of CPU utilization with the proposed algorithm was 27% while it was 25% for the base algorithm.

Figure 4 illustrates CPU utilization level at the end of the simulation. The average CPU utilization at this stage is about 46% in both algorithms. Although this amount is approximately equal in both algorithms, the proposed

algorithm has tried to change the number of idle servers to off situation in order to reduce energy consumption. In the proposed algorithm about 50 servers have switched to off status during the simulation. This amount is about 20 in the base algorithm.

With changing the status of the servers, the virtual machines are forced to migrate and the new destination is selected based on the SCLrMmt algorithm. Initially, as is shown in Figure 5, the number of migrations (~320) is almost the same for both algorithms.

By increasing the number of virtual machines from 320, the SCLrMmt algorithm tries to turn off the idle servers and reduce their energy consumption. As a result, the number of migrations goes up to 390. While in the LrRs algorithm about 340 migrations are performed.

The comparison of energy consumption in the two algorithms is illustrated in Figure 6. As the number of virtual machines on the servers increases during the simulation, the energy consumption of servers also increases. Although the rate of increase in energy is almost the same in both algorithms, energy consumption in the case of using the proposed algorithm is lower. The SCLrMmt algorithm has about 7% more energy savings in comparison to the LrRs algorithm. This reduction in energy consumption is due to the shutdown of the idle servers and hence the resulting CPU usage optimization. Also, the appropriate placement of virtual sensors affects proper energy consumption.

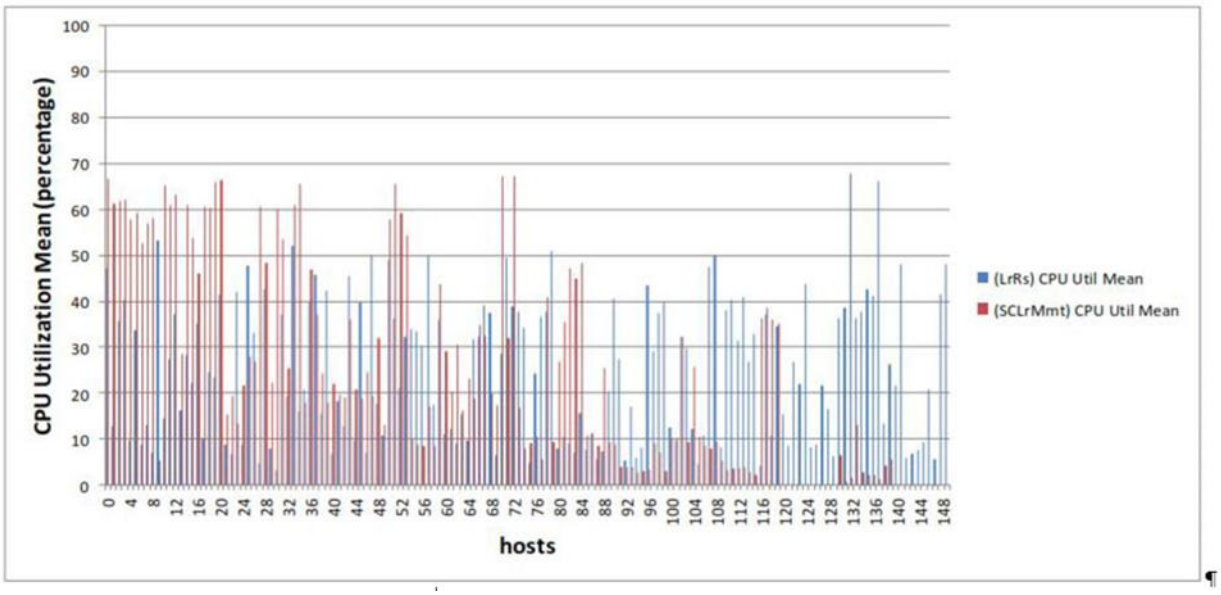


Fig 4. CPU utilization of servers at the end of simulation

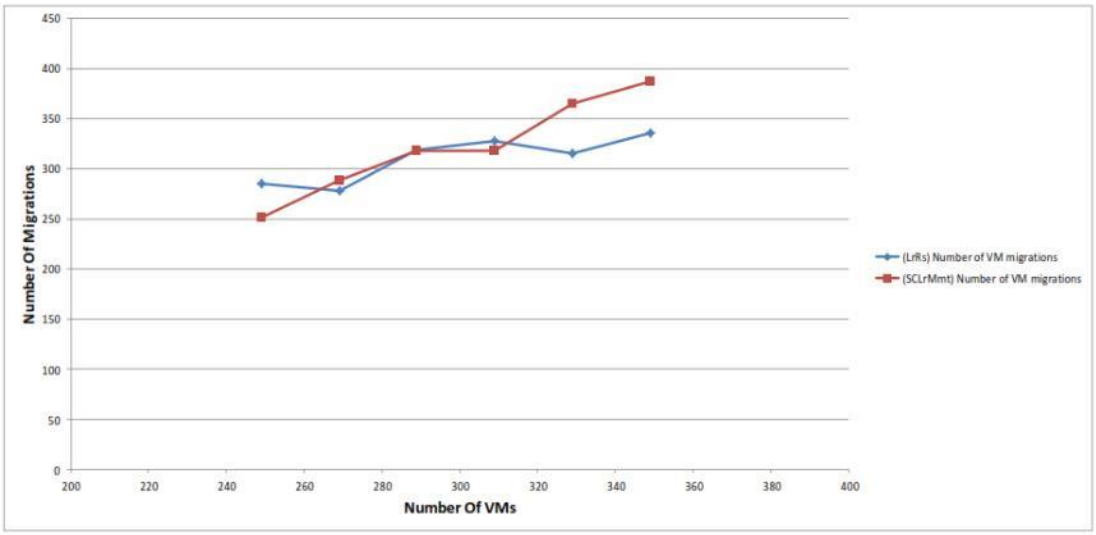


Fig 5. Number of migrations

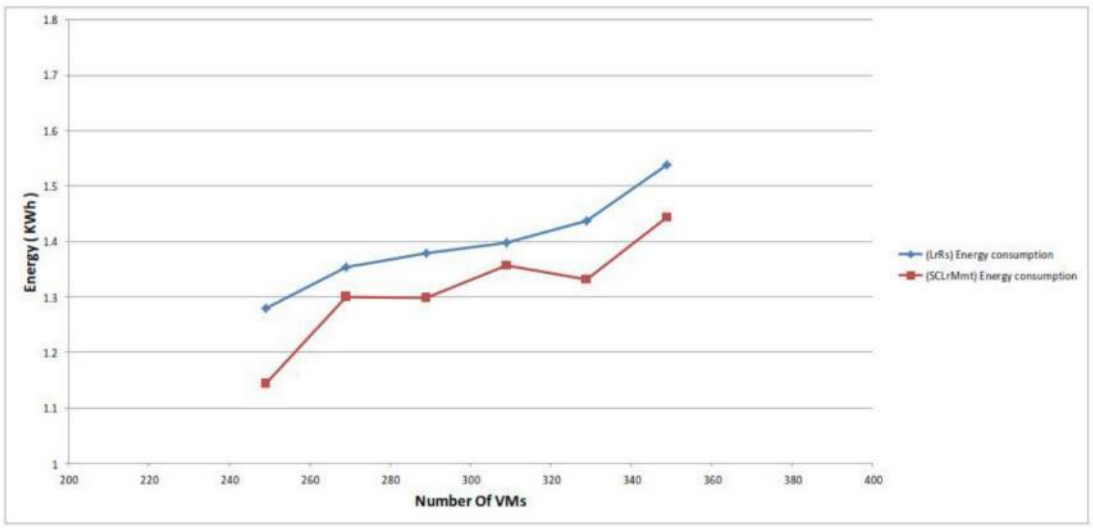


Fig 6. Energy consumption

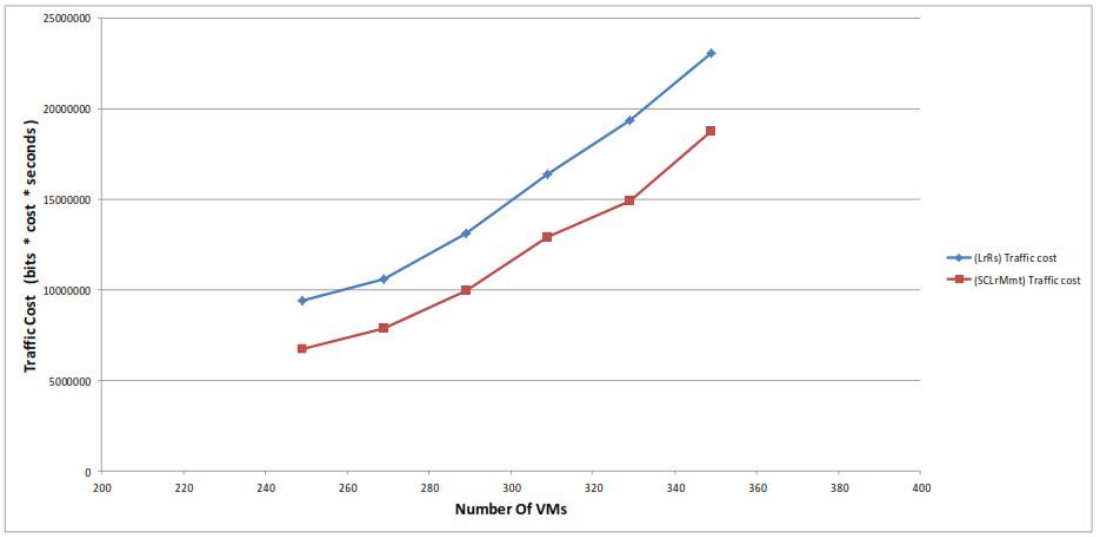


Fig 7. Traffic costs

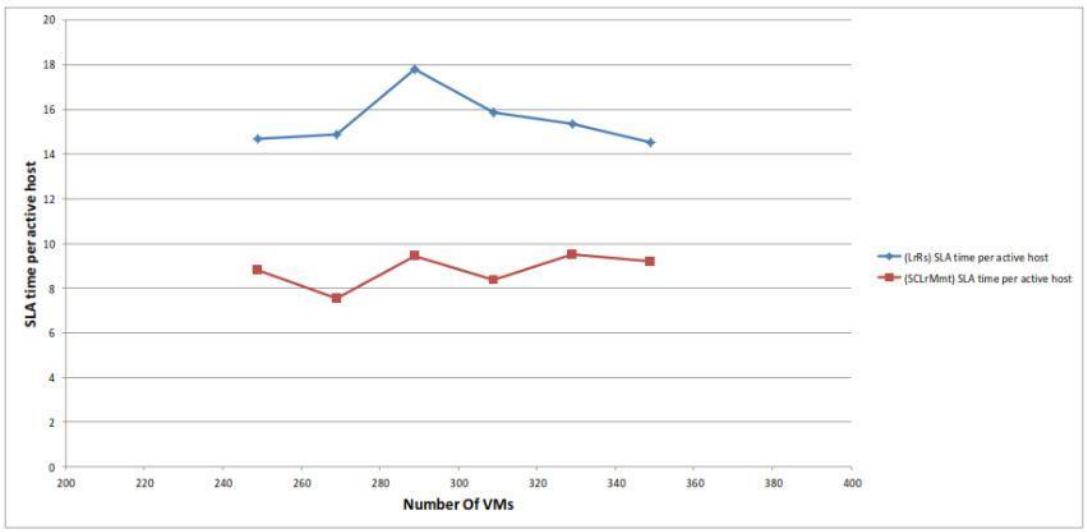


Fig 8. SLATAH comparison

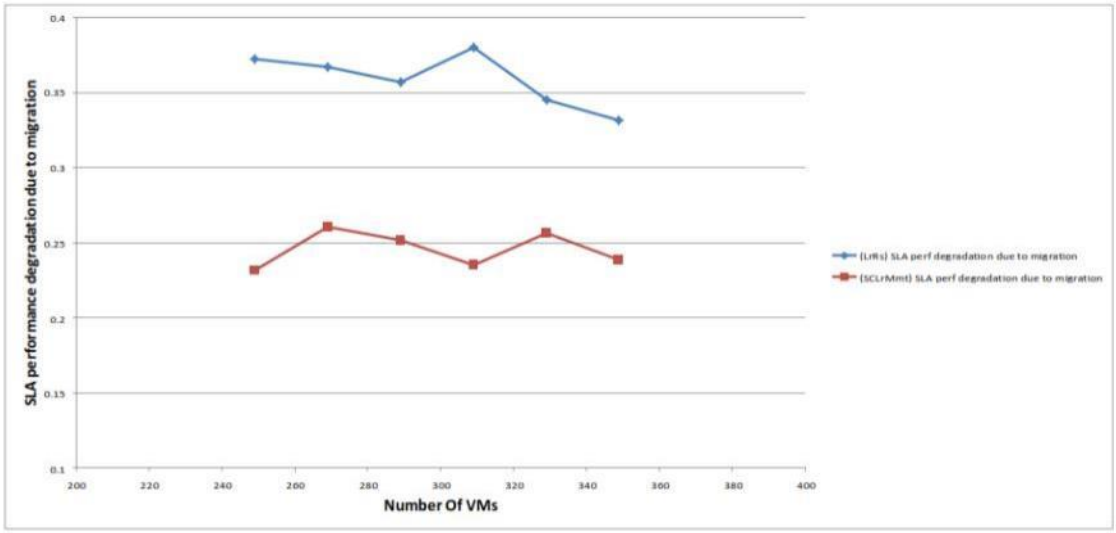


Fig 9. PDM comparison

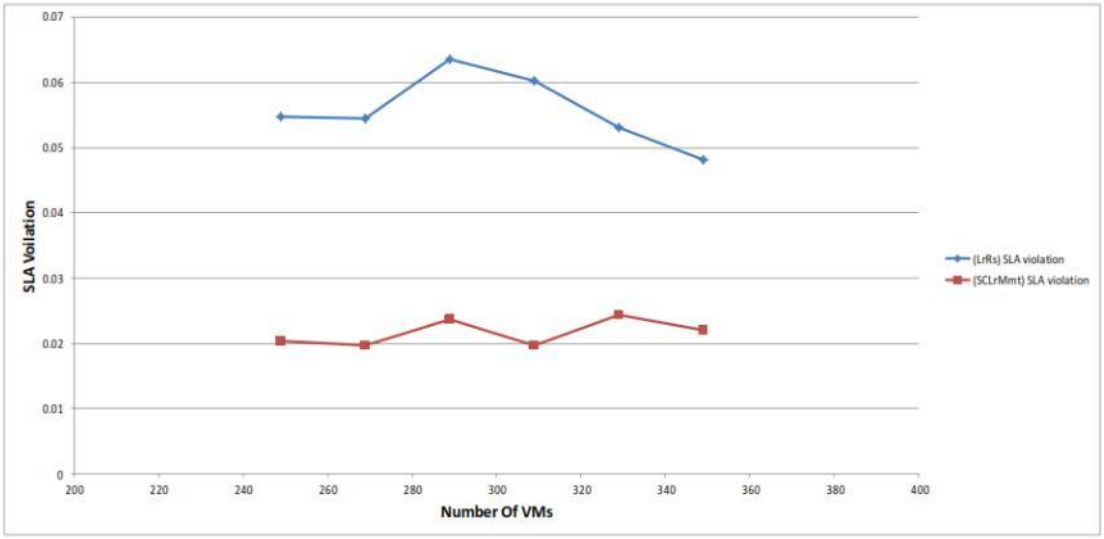


Fig 10. SLA violation

The traffic cost which is calculated based on the traffic between each pair of virtual sensors and the distance between them is compared for the algorithms in Figure 7. About 23% reduction in traffic cost is seen in the SCLrMmt algorithm. This is because the placement of virtual sensor groups are considered for choosing the destination of migration in order to minimizes the distance between the members of the groups. This type of placement results in reduction in traffic load and hence it lowers the cost.

The result of SLA violation for SCLrMmt and LrRs algorithm based on the two parameters SLATAH and PDM are illustrated separately in Figures 8 and 9. Subsequently, these two parameters are combined and SLA violation is calculated. Reduction in SLA violation is clearly visible in the proposed algorithm as shown in Figure 10.

5. Conclusion and future works

In this paper, a new approach for migration and reallocation of virtual sensors in sensor-cloud networks is proposed. By selecting the appropriate destination for migration, the proposed algorithm reduces the energy consumption in cloud servers. This method is based on minimization of the distance between the virtual sensor groups. It reduces the traffic overhead on the network and meets the real-time requirements of users. Reducing network traffic results in reducing violation of service level agreement. As a result, the desired quality of service is provided for applications. The results of the simulation of the proposed algorithm confirm improvements in energy efficiency, and decrease in traffic cost and SLA.

Since the main idea of the proposed algorithm for migration of virtual sensors is given from the migration of virtual machines in the cloud environment, there are similarities between them. Although in the new algorithm some special properties of sensor-cloud networks, such as the possibility of grouping and sharing virtual sensor groups are used, involving other parameters that are specific for sensor-cloud networks can clearly distinguish between cloud algorithms and sensor-cloud algorithms. Also, designing a tool that can simulate sensor-cloud networks influences the presentation of new algorithms. For example, just one virtual sensor is allocated to each virtual machine due to the lack of

an appropriate simulator for sensor-cloud networks, while allocation of virtual sensors to a virtual machine can be a new idea in designing new algorithms. In the proposed algorithm, the parameters that decide on the start time of a migration are the overhead on servers and CPU utilization. Other factors such as the amount of memory can also have an effect on the occurrence of migration.

Future work on the proposed algorithm is based on the number of user requests for using one special virtual sensor. If the number of users of a virtual sensor is too high, it may cause increased delay in the response time to them. This situation is not suitable for real-time applications. Another issue for future work is considering data stored in data centers. In the proposed algorithm, only real-time data are considered and there is no control over the data stored in data centers. Minimizing the distance between virtual sensors and databases can have a significant impact on network traffic.

Reference

[1] A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, and M. A. Hossain, "A Survey on Sensor-Cloud: Architecture, Applications, and Approaches", *International Journal of Distributed Sensor Networks*, vol. 2013, p. 18, Art. no. 917923, 2013.

[2] M. Körner, A. Stanik, O. Kao, M. Wallschläger, and S. Becker, "The ASCETiC Testbed - An Energy Efficient Cloud Computing Environment", *in Testbeds and Research Infrastructures for the Development of Networks and Communities: 11th International Conference*, TRIDENTCOM 2016, Hangzhou, China, June 14-15, 2016, Revised Selected Papers, S. Guo, G. Wei, Y. Xiang, X. Lin, and P. Lorenz, Eds. Cham: Springer International Publishing, pp. 93-102, 2016.

[3] Piraghaj, S. Fotuhi, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, A Survey and Taxonomy of Energy Efficient Resource Management Techniques in Platform as a Service Cloud. Handbook of Research on End-to-End Cloud Computing Architecture Design, 2016.

[4] S. Kabadayi, A. Pridgen, and C. Julien, "Virtual sensors: Abstracting data from physical sensors", *in In Proceedings of the International Symposium on on World of Wireless*, Mobile and Multimedia Networks, Buffalo-Niagara Falls, NY, 2006.

[5] S. Bose, N. Mukherjee, and S. Mistry, "Environment Monitoring in Smart Cities Using Virtual Sensors", *in 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 399-404, 2016.

[6] M. Yuriyama and T. Kushida, "Sensor-Cloud Infrastructure - Physical Sensor Management with Virtualized Sensors on Cloud Computing", *presented at the 13th International Conference on Network-Based Information Systems*, Takayama, 2010.

[7] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing", *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755-768, 2012.

[8] D. Huang, D. Yang, H. Zhang, and L. Wu, "Energy-aware virtual machine placement in data centers", *presented at the Global Communications Conference (GLOBECOM)*, Anaheim, CA, 2012.

[9] F. Song, D. Huang, H. zhou, and I. You, "application-aware virtual machine placement in data centers", *presented at the Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, 2012.

[10] C. Clark et al., "Live migration of virtual machines", *in Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*, vol. 2, pp. 273-286: USENIX Association, 2005.

[11] U. Deshpande and K. Keahey, "Traffic-sensitive Live Migration of Virtual Machines", *Future Generation Computer Systems*, 2016.

[12] P. Barham et al., "Xen and the Art of Virtualization", *ACM SIGOPS Operating Systems Review*, vol. 37, no. 5, pp. 164-177, 2003.

[13] V. Shrivastava, P. Zerfos, K. W. Lee, H. Jamjoom, Y. H. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers", *in INFOCOM*, Shanghai, 2011.

[14] T. Chen, X. Gao, and G. Chen, "Optimized Virtual Machine Placement with Traffic-Aware Balancing in Data Center Networks", *Scientific Programming*, 2016.

[15] D. S. Dias and M. K. Costa, "Online traffic-aware virtual machine placement in data center networks", *presented at the Global Information Infrastructure and Networking Symposium (GIIS)*, Choroni, 2012.

[16] J. Liu, J. Guo, and D. Ma, "Traffic Aware Virtual Machine Packing in Cloud Data Centers", *in 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity)*, IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS), pp. 256-261, 2016.

[17] H. T. Vu and S. Hwang, "A Traffic and Power-aware Algorithm for Virtual Machine Placement in Cloud Data Center", *International Journal of Grid & Distributed Computing*, vol. 7, no. 1, pp. Pages 21-32, 2014.

[18] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers", *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397-1420, 2012.

[19] M. F. H. Bhuiyan and C. Wang, "Energy-Efficient Virtual Machine Management in Heterogeneous Environment: Challenges, Approaches and

Opportunities", *presented at the International Conference on Systems, Man,* and Cybernetics (SMC), Manchester, 2013.

[20] M. Sharifi, H. Salimi, and M. Najafzadeh, "Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques", *The Journal of Supercomputing*, vol. 61, no. 1, pp. 46-66, 2012.

[21] X. Liu and L. Fan, "Priority-aware Gray-box Placement of Virtual Machines in Cloud Platforms", *presented at the arXiv preprint arXiv:1307*.6622, 2013.

[22] S. Madria, V. Kumar, and R. Dalvi, "Sensor cloud: A cloud of virtual sensors", *Software, IEEE,* vol. 31, no. 2, pp. 70-77, 2014.

[23] S. K. Madria, "Sensor Cloud: A Cloud of Sensor Networks", *2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW)*, 2016.

[24] R. Calheiros, R. Ranjan, A. Beloglazov, C. Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of Cloud computing environments and evaluation of resource provisioning algorithms", *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[25] "Cisco Data Center Infrastructure 2.5 Design Guide", May 2008.

[26] A. Gupta and N. Mukherjee, "Implementation of virtual sensors for building a sensor-cloud environment", *in 2016 8th International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1-8, 2016.

[27] R. Buyya, A. Beloglazov, and J. H. Abawajy, "Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges", CoRR, vol. abs/1006.0308, 2010.

[28] H. Goudarzi and M. Pedram, "Hierarchical SLA-Driven Resource Management for Peak Power-Aware and Energy-Efficient Operation of a Cloud Datacenter", *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 222-236, 2016.

Journal of Computer and Knowledge Engineering, Vol. 1, No. 2, 2017.
DOI: 10.22067/cke.v1i2.61798

Particle Filter based Target Tracking in Wireless Sensor Networks using Support Vector Machine

Ahmad Namazi Nik Abbas Ali Rezaee*

Abstract: Target tracking is estimating the state of moving targets using noisy measurements obtained at a single observation point or node. Particle filters or sequential Monte Carlo methods use a set of weighted state samples, called particles, to approximate the posterior probability distribution in a Bayesian setup. During the past few years, Particle Filters have become very popular because of their ability to process observations represented by nonlinear state-space models where the noise of the model can be non-Gaussian. There are many Particle Filter methods, and almost all of them are based on three operations: particle propagation, weight computation, and resampling. One of the main limitations of the previously proposed schemes is that their implementation in a wireless sensor network demands prohibitive communication capability since they assume that all the sensor observations are available to every processing node in the weight update step. In this paper, we use a machine learning technique called support vector machine to overcome this drawback and improve the energy consumption of sensors. Support Vector Machine (SVM) is a classifier which attempts to find a hyperplane that divides two classes with the largest margin. Given labeled training data, SVM outputs an optimal hyperplane which categorizes new examples. The training examples that are closest to the hyperplane are called support vectors. Using our approach, we could compress sensor observations and only support vectors will be communicated between neighbor sensors which lead to cost reduction in communication. We use LIBSVM library in our work and use MATLAB software to plot the results and compare the proposed protocol with CPF and DPF algorithms. Simulation results show significant reduction in the amount of data transmission over the network.

Keywords: Distributed Particle Filter; Support Vector Machines; Target Tracking; Wireless Sensor Networks**.**

1. Introduction

Target tracking is one of the most important applications of wireless sensor networks. Examples include security and surveillance [1], environmental monitoring [2] and tracking tasks [3]. Target tracking is the estimation of the current state and prediction of future states of a target based on measurements received from a sensor that is observing it. The limited on-board resources of the sensor node and the limited wireless bandwidth are the major constraints of performing target tracking in wireless sensor networks. In order to save resources, target tracking should be implemented in a distributed way. Distributed computation has found very successful applications in sensor networks, particularly when a powerful central unit is not available.

Before particle filtering methods became popular, the Kalman filter was the standard method for solving state space models [4]. The Kalman filter can be applied to optimally solve a linear Gaussian state space model. When linearity or Gaussian conditions do not hold, its variants, i.e. the extended Kalman filter and the unscented Kalman filter, can be used. However, for highly nonlinear and non-Gaussian problems they fail to provide a reasonable estimate.

Particle filtering techniques offer an alternative method. They work online to approximate the marginal distribution of the latent process as observations become available. Importance sampling is used at each point in time in order to approximate the distribution with a set of discrete values, known as particles, each with a corresponding weight. There are several papers and books which have presented detailed reviews of particle filters and their applications [5-12].

In this work we tackle the problem of implementing the DPF algorithm and make use of support vector machine – a well-known machine learning classification method – to compress measurements collected by processing nodes and thus reducing communication costs.

The rest of the paper is organized as follows. In Section 2, a brief review of prior related works on target tracking is presented. In Section 3 we introduce the problem of target tracking in the context of Bayesian filtering and describe the solution to the nonlinear filtering problem with a centralized PF. In Section 4 we provide a formal description of the DPF algorithm. Section 5 introduces support vector machines. In Section 6 we provide details of the proposed method. Simulation and experimental results are presented and discussed in Section 7 and, finally, Section 8 is devoted to conclusions.

2. Related Works

Target tracking has many real life applications such as battlefield surveillance, detection of illegal borders crossing, gas leakage, fire spread, and wildlife monitoring.

Various taxonomies of target tracking algorithms have been proposed in the literature and there is no standardized or predefined classification. Some works have studied tracking algorithms according to the security aspect [13] while others have considered energy efficiency [14], fault tolerance, mobility, accuracy, and so on [15].

A comparative study of target tracking with Kalman Filter, Extended Kalman Filter and Particle Filter using Received Signal Strength measurements has been reported in [16] and their simulation results show that PF has superior

Manuscript received Jaunary 14, 2017; accepted July 28, 2017.
A. Namazi Nik, Department of Information and Communication Technology, Payame Noor University, Tehran, Iran. namazi.a@pnurazavi.ac.ir.
*A. Ali Rezaee Department of Information and Communication Technology, Payame Noor University, Tehran, Iran. a_rezaee@pnu.ac.ir.

performance to the KF and EKF in terms of accuracy and root mean square error (RMSE).

The application of PFs in WSNs is challenging due to the limited resources of WSNs. Centralized particle filters (CPFs) have some problems such as consuming significant energy and vulnerability as a single point of failure. Distributed particle filters (DPFs) were studied as a response to these problems, in particular, to offload the computation from the central unit [17].

Particle filtering for target tracking in WSNs has already attracted some attention, including a body of work in distributed methods [18]. Its relation with agent networks has also been explored in [19].

In [20], a fully decentralized particle filtering algorithm for cooperative blind equalization is introduced. The technique is proper, in the sense that it does not make any approximations in the computation of the importance weights of the particles. However, the scheme is applicable only when the state signal is discrete, and would be infeasible in terms of computation and communication among nodes. In [21], the communication load is reduced using quantization and parametric approximations of densities. A similar parametric approach is applied in [18] to further simplify communications.

The work reported in [22] provides a generalized approach for approximating global likelihood through a consensus filter. It approximates log-likelihood by a polynomial function, and the sensors exchange only the coefficients of the polynomial function to compute global likelihood.

The authors in [23] proposed a distributed particle filtering algorithm with the objective of reducing the overhead data that is communicated among the sensors. In their algorithm, the sensors exchange information to collaboratively compute the global likelihood function that encompasses the contribution of the measurements towards building the global posterior density of the unknown location parameters. Each sensor uses its own measurement to compute its local likelihood function and approximates it using a Gaussian function. The sensors then propagate only the mean and covariance of their approximated likelihood functions to other sensors, thereby reducing the communication overhead. The global likelihood function is computed collaboratively from the parameters of the local likelihood functions using an average consensus filter or a forward-backward propagation information exchange strategy.

In [24] a distributed particle filter is designated and it is shown that the difference in accuracy of their proposed DPF and a centralized filter with the same total number of particles is less than 2 cm, while the DPF with four processing nodes is over four times faster than an equivalent centralized version. This equivalently means that the same performance can be obtained on less powerful hardware. The main limitation of that scheme is that every node performing a subset of the computations of the PF should have access to all the observations (i.e., all the measurements collected by the WSN at the current time step) in order to guarantee that the particle weights are proper and, therefore, the resulting estimators are consistent.

3. Nonlinear Filtering in State-Space System

3-1. Bayesian Filtering

Consider the Markov state-space random model with conditionally independent observations [25, 26] described by the triplet:

$$p(x_0), \quad p(x_t|x_{t-1}), \quad p(y_t|x_t), \quad t = 1, 2, \dots \tag{1}$$

We denote the states and the observations up to time t by $x_{0:t} \triangleq \{x_0, \dots, x_t\}$ and $y_{0:t} \triangleq \{y_0, \dots, y_t\}$, respectively. $p(x_0)$ is the prior probability density function (pdf) of the state, the transition density $p(x_t|x_{t-1})$ describes the (random) dynamics of the process x_t and the conditional pdf $p(y_t|x_t)$ describes how the observations are related to the state and it is usually referred to as the likelihood of x_t. The goal of a stochastic filtering algorithm is to recursively estimate the posterior distribution $p(x_t|y_{1:t})$, $t \geq 1$.

Suppose that the required pdf $p(x_{t-1}|y_{1:t-1})$ at time $t-1$ is available. The prediction stage obtains the prior pdf of the state at time t via:

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{2}$$

At time step t, an observation y_t becomes available, and it may be used to update the prior (update stage) via Bayes' rule:

$$p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1}) \tag{3}$$

Eqs. (2) and (3) form the basis for the optimal Bayesian solution [6]. If the system of Eq. (1) is linear and Gaussian then $p(x_t|y_{1:t})$ is Gaussian and can be obtained exactly using the Kalman filter algorithm [27]. If the state space is discrete and finite, exact solutions can also be computed [25]. However, if any of the pdf's in (1) is non-Gaussian, or the system is nonlinear, we have to resort to suboptimal algorithms in order to approximate the filter pdf $p(x_t|y_{1:t})$.

3-2. Particle Filtering

Particle Filters, also known as sequential Monte Carlo methods, are simulation based algorithms that yield estimates of the state based on a random point-mass (or "particle") representation of the probability measure with density $p(x_t|y_{1:t})$ [28-30]. Table 1 shows the standard particle filter algorithm. We refer to it as centralized in order to make explicit that it requires a central unit that collects all the observations together, generates all the particles and processes them together. The resampling step randomly eliminates samples with low importance weights and replicates samples with high importance weights in order to avoid the degeneracy of the importance weights over time [26, 31].

Table 1: The Centralized Particle Filter (CPF) algorithm

***Initialize**: At time $t = 0$*
For $m = 1, \dots, M$
sample $\boldsymbol{x}_0^{(m)}$ from prior $p(\boldsymbol{x}_0)$

***Recursive step**: for $t > 0$*
For $m = 1, \dots, M$

draw $\boldsymbol{x}_t^{(m)} \sim p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{(m)})$ and set $\boldsymbol{x}_{0:t}^{(m)} = \{\boldsymbol{x}_t^{(m)}, \boldsymbol{x}_{0:t-1}^{(m)}\}$
compute importance weights $w_t^{(m)*} = p(\boldsymbol{y}_t|\boldsymbol{x}_t^{(m)})$
Normalize weights $w_t^{(m)} = w_t^{(m)*} / \sum_{j=1}^{M} w_t^{(j)*}$
Resample the weighted sample $\{\boldsymbol{x}_{0:t}^{(m)}, w_t^{(m)}\}_{m=1}^{M}$ to obtain an unweighted sample $\{\boldsymbol{x}_{0:t}^{(m)}\}_{m=1}^{M}$

4. Distributed Particle Filtering

In this paper, we implement a distributed particle filter with nodes that can operate as processing elements (PEs) on a wireless sensor network. Each PE is a low-powered device that has to perform sensing, computation and radio communication tasks while running on batteries. A common assumption in other proposed schemes is that all observations can be readily made available to all PEs in the system [24, 32-33]. Such capacity cannot be taken for granted in a WSN, where the observations are collected locally by the nodes and communications are necessarily constrained because of energy consumption. This issue will be addressed in subsequent sections.

Assume we have N processing nodes in the network and each is capable of running a separate PF with K particles (we ignore any non-processing nodes for now since they do not run particle filters). The total number of particles distributed over the network is M=NK. In particular, after the completion of a full recursive step of the distributed PF at time t-1, the n-th PE should hold the set $\{x_{t-1}^{(n,k)}, w_{t-1}^{(n,k)*}, W_{t-1}^{(n)*}\}_{k=1,\dots,K}$, where $x_{t-1}^{(n,k)}$ is the k-th particle at the n-th PE, $w_{t-1}^{(n,k)*}$ is the corresponding non-normalized importance weight, and $W_{t-1}^{(n)*}$ is the non-normalized aggregated weight of PE n.

Each PF runs locally on a node involves the usual steps of drawing samples, computing weights and resampling. The generation of new particles, the update of the importance weights and the resampling step are taken strictly locally, without interaction between different nodes. To be specific, assume that the transition pdf of model (1) is used as an importance function and that the observation vector y_t is available at every node. Then, at the n-th PE, and for $k = 1, \dots, K$, $x_t^{(n,k)}$ is drawn from the pdf $p(x_t^{(n,k)}|x_{t-1}^{(n,k)})$, and the corresponding nonnormalized weight is computed as $w_t^{(n,k)*} = w_{t-1}^{(n,k)*} p(y_t|x_t^{(n,k)})$.

Hence, the information stored by the n-th node at this point becomes $\{x_t^{(n,k)}, w_t^{(n,k)*}\}_{k=1,\dots,K}$ and the aggregated weight is $W_t^{(n)*} = \sum_{k=1}^{K} w_t^{(n,k)*}$.

Next, a resampling step is taken locally by each PE. Assuming a multinomial resampling algorithm, we assign, for $k = 1, \dots, K$, $x_t^{(n,k)} = x_t^{(n,j)}$ with probability $w_t^{(n,j)}$ and $j \in \{1, \dots, K\}$, where $w_t^{(n,j)} = \frac{w_t^{(n,j)*}}{\sum_{l=1}^{K} w_t^{(n,l)*}}$, $j = 1, \dots, K$, are the locally normalized importance weights. After resampling, the particles at the n-th PE are equally weighted.

In the estimation step, we obtain local estimates of target position at any node as:

$$\hat{x}_t^n = E(x_t|y_{1:t}) = \int x_t p(x_t|y_{1:t})\, dx_t = \sum_{k=1}^{K} w_t^{(n,k)} x_t^{(n,k)} \quad (5)$$

where $w_t^{(n,k)} = w_t^{(n,k)*} / W_t^{(n)*}$, $k = 1, \dots, K$ are the locally normalized importance weights.

Global estimates can be easily computed by a linear combination of local estimates. In order to obtain a global estimate of target position, each node n in the network should transmit its local estimate $\hat{x}_t^n$ and its aggregated weight $W_t^{(n)*}$ to a prescribed node (working as a fusion center) where global estimates can be computed as:

$$\hat{x}_t^{MMSE} = \sum_{k=1}^{K} W_t^{(n)} \hat{x}_t^{(n)} \quad (6)$$

where $W_t^{(n)} = W_t^{(n)*} / \sum_{i=1}^{N} W_t^{(i)*}$ is the globally normalized aggregated weight of the n-th node.

5. Support Vector Machine

Support vector machines discriminate two classes by fitting an optimal linear separating hyperplane to the training samples of two classes in a multidimensional feature space. The optimization problem being solved aims to maximize the margins between the optimal linear separating hyperplane and the closest training samples which are called support vectors (Figure 1). In a linearly non-separable case, the input data are mapped into a high-dimensional space in which the new distribution of the samples enables the fitting of a linear hyperplane [34].

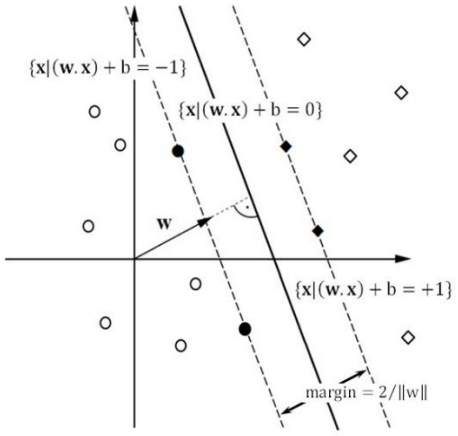


Fig 1. An example of classification of two classes by SVM. The support vectors are filled.

Assume some training data S which are a set of n points of the form:

$$S = \left\{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i \in \{+1, -1\}\right\} \quad i = 1, \dots, n \quad (7)$$

where $\mathbb{R}^d$ indicates the class to which point x_i belongs and each x_i is a d-dimensional real vector. The goal of SVM is to define a hyperplane which divides S, such that all the points with the same label are on the same side of the hyperplane while maximizing the distance between the two classes +1, -1 and the hyperplane. The boundary can be expressed as $w.x + b = 0$, where w is the normal vector to the hyperplane. The parameter $\frac{b}{\|w\|}$ determines the perpendicular distance from the hyperplane to the origin along the normal vector w and $\|w\|$ is the Euclidean norm of w. The data points nearest to the boundary are used to define the margins between the two classes and are known as support vectors. At the margins, where the support vectors are located, the equations for classes +1 and -1, respectively, are:

$$w.x + b = +1\,, \quad w.x + b = -1 \quad (7)$$

and the following decision function can be used to classify any data point in either class +1 or -1:

$$f(x) = sign(w.x + b) \quad (8)$$

The margin between the two classes is measured perpendicular to the hyperplane is $\frac{2}{\|w\|}$, so we want to minimize $\|w\|$. In a linearly separable case, the support vector machine looks for the separating hyperplane with the largest margin. Suppose that all the training data satisfy these constraints:

$$w.x_i + b \geq +1 \quad \forall\, x_i \; with \; y_i = +1 \quad (9)$$

$$w.x_i + b \leq -1 \quad \forall\, x_i \; with \; y_i = -1 \quad (10)$$

*T*hese can be combined into one inequality:

$$y_i(w.x_i + b) \geq 1 \quad i = 1, 2, \dots, N \quad (11)$$

where N is the number of training sets. According to [28] it is worth to use Lagrangian formulation of the problem. Thus, introducing Lagrange multipliers $\alpha_i \geq 0$, i = 1, 2 , … , N, one for each of the constraints in Eq. (9), we get the following Lagrangian:

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i (w.x_i + b) + \sum_{i=1}^{N} \alpha_i \quad (12)$$

We must now minimize Eq. (10) with respect to w and b, and maximize it with respect to α_i. Thus:

$$\frac{\partial}{\partial w} L(w, b, \alpha) = 0, \quad \frac{\partial}{\partial b} L(w, b, \alpha) = 0 \quad (13)$$

which leads to:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i, \quad \sum_{i=1}^{N} \alpha_i y_i = 0 \quad (14)$$

Substituting Eq. (12) into Eq. (10) yields the dual quadratic optimization problem:

Maximize

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j x_i . x_j \quad (15)$$

Subject to

$$\alpha_i \geq 0, \quad i = 1,2, \dots, N, \quad (16)$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad (17)$$

On substitution of Eq. (12) into the decision function (6) we obtain an expression which can be evaluated in terms of dot products between the pattern to be classified and the Support Vectors:

$$f(x) = sign(\sum_{i=1}^{N} \alpha_i y_i (x_i . x) + b) \quad (18)$$

The dot product can therefore be replaced with a nonlinear kernel function, thereby performing large margin separation in the feature-space of the kernel.

6. Using Support Vector Machine with Distributed Particle Filter

We use LIBSVM [35] in our work. LIBSVM is a library for Support Vector Machines and has gained wide popularity in machine learning and many other areas [36].

The Web address of the package is at http://www.csie.ntu.edu.tw/~cjlin/libsvm. Also, we use the MATLAB software to plot the results.

A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one "target value" (i.e. the class labels) and several "attributes" (i.e. the features or observed variables). The goal of SVM is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes. Our idea is to make use of support vector machine as a data classification technique in our work to reduce communications among the nodes.

As we mentioned in section 4 in the weight update step we assume that the observation vector y_t is available at every node which involves communications among the nodes. We use SVM to reduce these communications. SVMs only consider points near the margin (support vectors) instead of whole data points. According to our assumption, the observation coming from sensor j at time t, denoted $y_{j,t}$, is modeled as a binary observation. Then our SVM has two classes. Each sensor has two attributes which are equal to the coordinates of its position.

Scaling before applying SVM is very important. The main advantage of scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems. In [37] it is recommended to linearly scale each attribute to the range [-1, +1] or [0, 1]. We have to use the same method to scale both training and testing data. For example, suppose that we scaled the first attribute of training data from [-10, +10] to [-1, +1]. If the first attribute of testing data lies in the range [-11; +8], we must scale the testing data to [-1.1, +0.8]. There are four basic kernel functions in SVM, including linear, polynomial, radial basis function (RBF) and sigmoid. In our work we have used RBF kernel in the training step since it has fewer numerical difficulties and has better performance in nonlinear cases.

When the training is done, support vectors are generated. Once the support vectors are determined, the rest of the feature set can be discarded, since the support vectors contain all the necessary information for the classifier. We propagate observations corresponding to these support vectors ($\bar{y}_t$) rather than the whole y_t in the network. Then, in the weight update step of our distributed particle filter, every processing element can obtain observations of other sensors by running the final step of the SVM, namely prediction. On the other hand, in the prediction step of SVM, we obtain observation vector y_t from vector $\bar{y}_t$. Table 2 summarizes the DPF algorithm investigated in this paper.

Table 2. Distributed Particle Filter (DPF) algorithm

***Initialize**: At time $t = 0$, for $n = 1, \dots \dots N$*

Draw $\boldsymbol{x}_0^{(n,k)}$, for $k = 1, \dots, K$, from prior $p(\boldsymbol{x}_0)$

Assign $w_0^{(n,k)*} = \frac{1}{K}$ for all k, set $W_0^{(n)*} = 1$

Build the set $\{\boldsymbol{x}_0^{(n,k)}, w_0^{(n,k)*}, W_0^{(n)*}\}_{k=1}^{K}$

***Recursive step**: At time $t > 0$, start from the set $\{\boldsymbol{x}_{t-1}^{(n,k)}, w_{t-1}^{(n,k)*}, W_{t-1}^{(n)*}\}_{k=1}^{K}$. Then, for $n = 1, \dots, N$*

***Sampling**: Draw $\boldsymbol{x}_t^{(n,k)}$ from $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{(n,k)})$, for $k = 1, \dots, K$*

***Weight update**: $w_t^{(n,k)*} = w_{t-1}^{(n,k)*} p(\boldsymbol{y}_t|\boldsymbol{x}_t^{(n,k)})$*

***Estimation**: compute the desired output, such as the expected value*

***Resampling**: to obtain the set $\{\boldsymbol{x}_t^{(n,k)}, w_t^{(n,k)*}, W_t^{(n)*}\}_{k=1}^{K}$, where $w_t^{(n,k)*} = W_t^{(n)*}/K \quad for\ k = 1, \dots, K$*

7. Simulation and Experimental Results

The goal of our work is to implement a DPF for target tracking in a wireless sensor network and use SVM to compress measurements collected by these sensors. Our experimental scenario is shown in Figure 2. It is a room with 10 nodes (which are equipped with a light sensor) enclosing an area of 4×6 m^2 with a single source of natural light (a window). Modeling environment specifications and translating the disturbances caused by the target in the sensor readings into distance measurements are very complex. Then, instead we emphasize on obtaining binary observations: 1 if the target is in the detection zone and 0 otherwise.

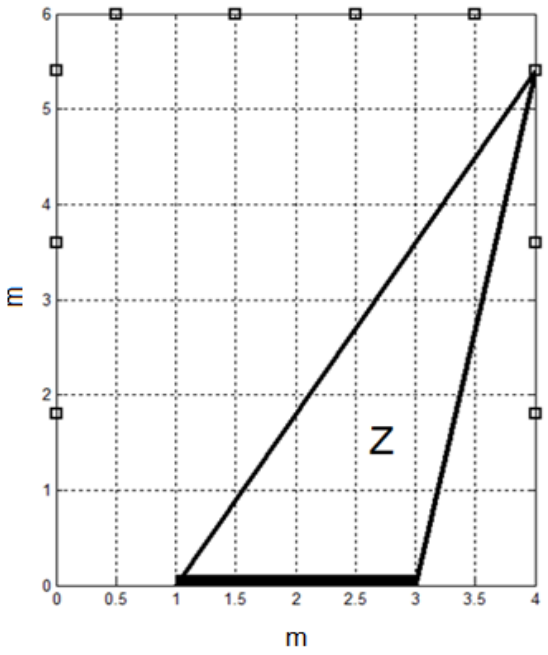


Fig. 2 Tracking scenario of 4×6 m^2. The thick line is the light source. There are 10 nodes equipped with light sensors around the edges, indicated by squares. The entry to the scenario lies at the bottom-right corner.

Table 3 displays values of the relevant simulation and algorithm parameters. The number of processing elements (N) is 4 in our experiments and we use N=1 as the equivalent to a centralized particle filter. Changing N affects other variables, such as the number of sensing-only elements (J-N) and the number of particles per PE (K=M/N). It does not matter which of the nodes are PEs and which are SEs, since we assume a fully connected network. Each node (either PE or SE) produces one binary observation every T_s second.

Figure 3 displays the empirical distribution of errors, and the average error, for 100 simulated paths. Figure 4 plots two selection of these paths along with the path estimated by our SVM-based DPF. The dissensions between true and estimated position tend to happen when the target moves between detection zones. Since the observations are binary and zone-based, rather than distance-based, there are gaps around the edges (see for example the final points in Figure 4). Accuracy also tends to be higher nearer the light source where more detection zones overlap.

Table 3. Simulation and algorithm parameters

Variable	Symbol	Value (unit)
Number of PEs	N	4
Number of nodes	J	10
Number of SEs		J-N
Total number of particles	M	100
Number of particles/PE	K	M/N
Number of timesteps	T	20 (s)
Sampling period	T_s	1 (s)

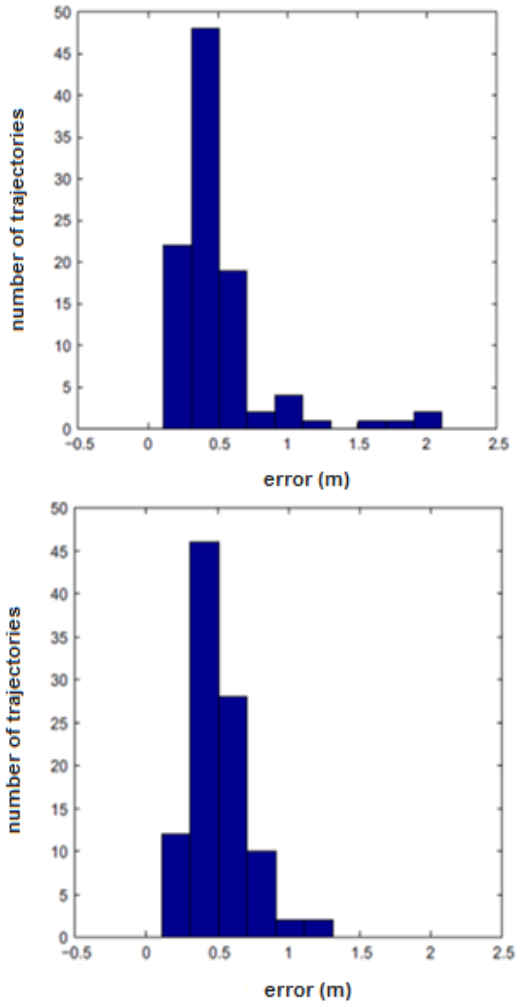
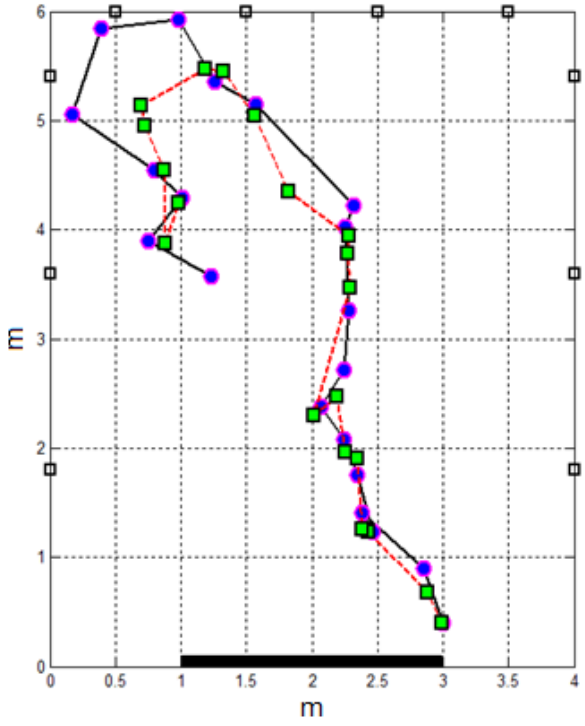


Fig 3. Histogram of position error in meters for both the centralized (up) and our distributed (down) versions of the particle filter over 100 simulated trajectories.



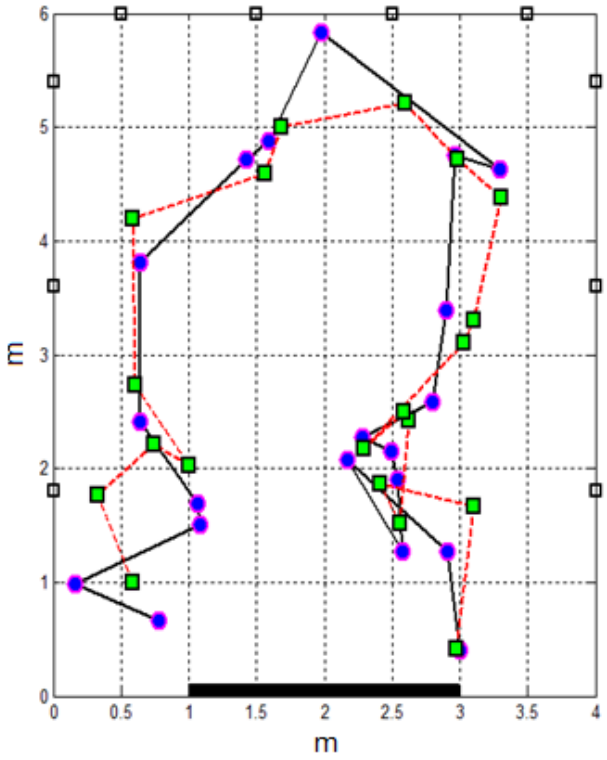


Fig 4. The simulated (black) path for two simulations, and the corresponding SVM-based DPF-estimated path (red); over T=20 time steps.

Figure 5 displays the amount of saving in the volume of propagating information for updating particle weights, using the proposed method, for 100 simulated paths. The horizontal axis shows the simulation run and the vertical axis shows the amount of propagating observations (in percent) on the network compared to the case when SVM is not used. The results show that using the proposed scheme, only %51.9 of sensor observations are propagated on the network, compared to the work done in [24], that leads to saving energy consumption of sensors.

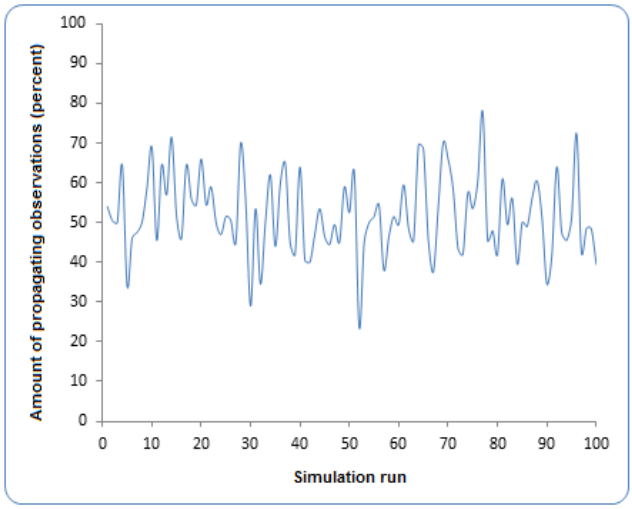


Fig 5. The amount of saving in the volume of propagating information for updating particle weights, using our proposed method, for 100 simulated paths.

8. Conclusion

In this paper, we have described the implementation of a distributed particle filter for target tracking in a wireless sensor network. One of the main limitations of similar works is the need to make all sensor observations available to every processing node. To overcome this limitation, we have used support vector machine to compress sensor observations. Simulation results show that the difference in accuracy of the proposed scheme and centralized particle filter and also distributed particle filter are insignificant, whereas by combining SVM with DPF we have reduced communications among the nodes around %48. Since SVMs only consider points near the margin (support vectors) instead of whole data points, they are suitable for data compression. SVMs can produce accurate and robust classification results on a sound theoretical basis, even when input data are non-monotone and non-linearly separable. The biggest limitation of the support vector approach lies in choice of the kernel function. In our work, we have used RBF kernel in the training step since it has fewer numerical difficulties and has better performance in nonlinear cases.

References

[1] E. Cayirci, H. Tezcan, Y. Dogan, and V. Coskun, "Wireless sensor networks for underwater survelliance systems", *Ad Hoc Networks*, vol. 4, pp. 431-446, 2006.

[2] S. Santini, B. Ostermaier, and A. Vitaletti, "First experiences using wireless sensor networks for noise pollution monitoring", *presented at the Proceedings of the workshop on Real-world wireless sensor networks*, Glasgow, Scotland, 2008.

[3] H.-W. Tsai, C.-P. Chu, and T.-S. Chen, "Mobile object tracking in wireless sensor networks", *Computer Communications*, vol. 30, pp. 1811-1825, 6/8/ 2007.

[4] A. Ribeiro, G. B. Giannakis, and S. I. Roumeliotis, "SOI-KF: Distributed Kalman Filtering With Low-Cost Communications Using the Sign of Innovations", *IEEE Transactions on Signal Processing*, vol. 54, pp. 4782-4795, 2006.

[5] A. Dhital, P. Closas, and C. Fernández-Prades, "Bayesian filtering for indoor localization and tracking in wireless sensor networks", *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, pp. 1-13, 2012.

[6] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking", *IEEE Transactions on Signal Processing*, vol. 50, pp. 174-188, 2002.

[7] P. M. Djuric, M. Vemula, and M. F. Bugallo, "Target Tracking by Particle Filtering in Binary Sensor Networks", *IEEE Transactions on Signal Processing*, vol. 56, pp. 2229-2238, 2008.

[8] Y. Huang, W. Liang, H.-b. Yu, and Y. Xiao, "Target tracking based on a distributed particle filter in underwater sensor networks", *Wireless Communications and Mobile Computing*, vol. 8, pp. 1023-1033, 2008.

[9] S. Sarkka, "Bayesian Filtering and Smoothing", *Cambridge University Press*, 2013.

[10] H. Q. Liu, H. C. So, F. K. W. Chan, and K. W. K. Lui, "Distributed particle filter for target tracking in sensor networks", *Progress In Electromagnetics Research C*, vol. 11, pp. 171-182, 2009.

[11] K. Achutegui, L. Martino, J. Rodas, C. J. Escudero, and J. Miguez, "A multi-model particle filtering algorithm for indoor tracking of mobile terminals using RSS data", *in Control Applications*, (CCA) & Intelligent Control, (ISIC), 2009 IEEE, 2009, pp. 1702-1707.

[12] O. Hlinka, F. Hlawatsch, and P. M. Djuric, "Distributed particle filtering in agent networks: A survey, classification, and comparison", *IEEE Signal Processing Magazine*, vol. 30, pp. 61-81, 2013.

[13] A. Oracevic and S. Ozdemir, "A survey of secure target tracking algorithms for wireless sensor networks", *in Proceedings of the World Congress on Computer Applications and Information Systems (WCCAIS '14),* pp. 1–6, IEEE, Hammamet, Tunisia, January 2014.

[14] O. Demigha, W.-K. Hidouci, and T. Ahmed, "On energy efficiency in collaborative target tracking in wireless sensor network: a review", *IEEE Communications Surveys and Tutorials*, vol. 15, no. 3, pp. 1210–1222, 2013.

[15] A. Ez-Zaidi, S. Rakrak, "A Comparative Study of Target Tracking Approaches in Wireless Sensor Networks", *Journal of Sensors*, vol. 2016, p. 11, 2016.

[16] M. W. Khan, N. Salman, A. Ali, A. M. Khan, and A. H. Kemp, "A comparative study of target tracking with Kalman filter, extended Kalman filter and particle filter using received signal strength measurements", *in Emerging Technologies (ICET),* 2015 International Conference on. IEEE, 2015.

[17] B. Jiang, B. Ravindran, "Completely Distributed Particle Filters for Target Tracking in Sensor Networks", *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, pp. 334-344, 2011.

[18] O. Hlinka, O. Sluciak, F. Hlawatsch, P. Djuric, M. Rupp, "Distributed Gaussian particle filtering using likelihood consensus", *in: International Conference on Acoustics*, Speech and Signal Processing, pp. 3756–3759, May 2011.

[19] O. Hlinka, F. Hlawatsch, P. Djuric, "Distributed particle filtering in agent networks", *IEEE Signal Process.* Mag. pp. 61-81 (January) (2013).

[20] Claudio J. Bordin, Marcelo G. S. Bruno, "Cooperative bling equalization of frequency-selective channels in sensor networks using decentralized particle filtering", *in: 42nd Asilomar Conference on Signals, Systems and Computers*, pp. 1198–1201, October 2008.

[21] Mark Coates, "Distributed particle filters for sensor networks", *in: The International Conference on Information Processing in Sensor Networks*, (IPSN),

pp. 99–107, April 2004.

[22] O. Hlinka, P. Djuric, F. Hlawatsch, "Consensus-based distributed Particle Filtering with distributed proposal adaptation", *IEEE Trans. Signal Process.*, vol 62, pp. 3029–3041, 2014.

[23] T. Ghirmai, "Distributed Particle Filter for Target Tracking: With Reduced Sensor Communications", Sensors, 16, 1454, 2016.

[24] J. Read, K. Achutegui, and J. Míguez, "A distributed particle filter for nonlinear tracking in wireless sensor networks", *Signal Processing*, vol. 98, pp. 121-134, 2014.

[25] A. Doucet, N. d. Freitas, and N. Gordon, "Sequential Monte Carlo Methods in Practice" Springer, 2001.

[26] B. Ristic, S. Arulampalam, and N. Gordon, "Beyond the Kalman Filter: Particle Filters for Tracking Applications", Artech House, 2004.

[27] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems", *Journal of Basic Engineering*, vol. 82, pp. 35-45, 1960.

[28] A. Bain and D. Crisan, "Fundamentals of Stochastic Filtering", Springer, 2009.

[29] O. Cappe, S. J. Godsill, and E. Moulines, "An Overview of Existing Methods and Recent Advances in Sequential Monte Carlo", *Proceedings of the IEEE*, vol. 95, pp. 899-924, 2007.

[30] Djuric, x, P. M., J. H. Kotecha, Z. Jianqui, H. Yufei, et al., "Particle filtering", *IEEE Signal Processing Magazine*, vol. 20, pp. 19-38, 2003.

[31] L. Tiancheng, M. Bolic, and P. M. Djuric, "Resampling Methods for Particle Filtering: Classification, implementation, and strategies", *IEEE Signal Processing Magazine*, vol. 32, pp. 70-86, 2015.

[32] M. Bolic, P. M. Djuric, and H. Sangjin, "Resampling algorithms and architectures for distributed particle filters", *IEEE Transactions on Signal Processing*, vol. 53, pp. 2442-2450, 2005.

[33] J. Míguez, "Analysis of parallelizable resampling algorithms for particle filtering", *Signal Processing*, vol. 87, pp. 3155-3174, 2007.

[34] C. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining and Knowledge Discovery*, vol. 2, pp. 121-167, 1998/06/01 1998.

[35] https://www.csie.ntu.edu.tw/~cjlin/libsvm.

[36] C.-c. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines", *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 2011.

[37] C.-w. Hsu, C.-c. Chang, and C.-j. Lin, "A practical guide to support vector classification", 2010.

Journal of Computer and Knowledge Engineering, Vol. 1, No.2, 2017.
DOI: 10.22067/cke.v1i2.62961

A Transmission Method to Guarantee QoS Parameters in Wireless Sensor Networks

Maryam Kordlar Gholamhossein Ekbatanifard
Ahad Jahangiry Ramin Ahmadi*

Abstract: Energy-efficient and reliable data transmission are crucial issues in WSNs due to the energy constraints and high packet loss rates, In order to increase network reliability and throughput, multipath forwarding is used in many applications. However because of using multiple paths between source and destination, the multipath forwarding mechanism increases network overhead. By splitting the original messages and forwarding each sub-packet through multipath routing protocols, energy consumption and network overhead can be reduced. This paper proposes a flexible recovery mechanism which increases reliability and minimizes both energy consumption and delay. The proposed scheme caches sub-packets at some special intermediate nodes and retransmits from these nodes instead of source node whenever it is necessary. Performance evaluations of sub-packet caching (SPC) demonstrate that using multipath forwarding with caching data can effectively increase reliability and reduce energy consumption and latency.

Keywords Wireless Sensor Networks, Energy-efficient Routing, Multipath, Reliable Transmission, loss Recovery

1. Introduction

With the development of miniaturized sensor nodes, wireless sensor networks (WSNs) [1] have become a promising technology that can play an essential role in many vital and critical fields. Sensors are responsible of forwarding significant data, immediately and efficiently to the base station for processing. Increasing reliability in WSNs is a challenging task which have made it the target of many studies. Because sensor nodes are powered by life-limited and irreplaceable batteries, minimizing the power usage of each node is important and must be considered while designing WSNs. In order to increase reliability, multipath forwarding is used in many applications. Single path forwarding enhances the choice of the same path that can cause dropping of the most used nodes. Inverse multipath routing allows forwarding packets through multiple paths between source and destination as a result of which multiple copies of data will be sent. Although it can remarkably enhance network overhead and consume more energy, reliability will be guaranteed. To overcome these inherent properties of many multipath routing protocols, the splitting approach can be used. Original messages can split in several sub-packets to be forwarded by multipath routing algorithm such that each node will forward only small sub-packets. The splitting procedure that is used in this paper is the Chinese Remainder Theorem (CRT) [4], which is characterized by a simple modular division between integers [9]. Meanwhile corrupted and lost packets are unavoidable facts in WSNs because of data transmissions over radio links. A packet loss can fall out due to wireless link errors or destroyed nodes due to environmental events or node crash down due to energy depletion and decreased network efficiency. This paper proposes a flexible recovery mechanism for lost packets which guarantee energy-efficient reliable data transmission. The proposed scheme caches sub-packets at some special intermediate nodes and retransmits from these nodes whenever it is necessary. Hop by hop protocol is an approach that caches data to every intermediate node. NBH [2] not only increases reliability, but it also increases network overhead. Inverse, End-to-End [3] protocol does not perform data caching. Therefore the network overhead will be balanced while the reliability decreases. Providing some reliability support at intermediate nodes is more energy-efficient. The proposed method which is called SPC fulfills this requirement by caching packets at some special intermediate nodes.

The remainder of this paper is organized as follows: Section II presents related work, subsequently section III presents the Chinese Remainder Theorem (CRT). We describe the proposed scheme in section IV. Section V details our simulation efforts and finally section VI concludes the paper.

2. Related Work

Reliable transmission is an important issue in extremely unreliable link environments of wireless sensor networks, which have become more difficult to attain due to the increased number of nodes. Recent studies have shown that in WSNs links are highly unreliable due to many factors such as interference, attenuation and fading [5-6]. Messages in the network are delivered according to their deadlines. Several routing protocols have been proposed to provide reliable data transmission. In [7] a Distributed and Reliable Data Transmission (DRDT) scheme with the objective of efficiently guaranteeing reliable data transmission is provided which increases reliability by cooperative retransmission task by helping nodes.

DRDT effectively reduces the number of retransmissions by using helping nodes. Selection of the helping node is based on the quality of the link to the receiver node. DRDT guarantees high energy-efficiency and effectively reduces

Manuscript received April 2, 2017; accepted November 12, 2017.
M. Kordlar Young Researchers Club, Tabriz Branch, Islamic Azad University, Tabriz, Iran.
Gh. Ekbatanifard, Department of Computer Engineering, Lahijan Branch, Islamic Azad University, Lahijan, Guilan, Iran.
A. Jahangiry, Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran.
* R. Ahmadi, Dept. of Computer Engineering, Rasht Branch, Islamic Azad University, Rasht, Guilan, Iran, Raminahmadi@iaurasht.ac.ir.

end-to-end transmission delays. Ganesan *et al*. reported that packet loss can occur over a short distance while the distribution of packet reception over an area is not uniform and low-power environments have quite asymmetric links [8]. The authors of [9] proposed a multi-path routing model for WSNs referred to as Multi-Constrained Multi-Path routing (MCMP) where packet delivery from nodes to the sink is achieved based on QoS constraints expressed in terms of reliability and delay. This model addresses the issue of multi-constrained QoS in wireless sensor networks taking into account the unpredictability of network topology and trying to minimize energy consumption. Couto *et al*. [10] have measurements for DSDV and show that with minimizing the hop count, the distance traveled maximizes by each hop and it increases the loss ratio. Another study proposes a packet delivery mechanism for energy aware called Multi Path and Multi-Speed Routing Protocol (MMSPEED), that overlays on the network layer and medium access control layer. This protocol addresses reliability dependency and presents multiple levels of delivery speed. In MMSPEED each node is aware of its neighbor nodes' geographical information within its radio range. Each node transmit data message to the closer neighbor node, so the data can be delivered to the sink without learning global information. Thus, MMSPEED provides multiple delivery speed, but it does keep an individual node's energy and it does not remove node's energy except for when it selects paths for forwarding to the sink. Therefore, many data packets are routed over the same routes. It is the opposite of load balancing in wireless sensor networks [11]. In [12] a Cluster-Based Forwarding (CBF) is proposed which guarantees reliability by using a cluster that consists of helper nodes with good link quality. The disadvantage of CBF arise when the location of the sink changes, so all the nodes should select helping nodes again. Therefore, CBF cannot fully guarantee the reliability of realistic wireless sensor networks. The Energy-Efficient Multiple Paths Routing Algorithm (EMRA) for WSNs is a routing algorithm that increases resilience to node failure. This algorithm is based on a data-centric and location-based approach to find discrete paths. This application is used to control overhead and energy consumption in the network. In EMRA, the source node floods data message to the sink, when after the sink node receives the data message it can find the main path. To find other secondary paths, the sink node tries its neighbor nodes by intermediate nodes. Thus, it reduces the delay to set up multiple paths [13]. Using caching in WSNs was first proposed in [14]. The authors of [15] proposed a multi-path-based distributed TCP caching algorithm, which uses redundant paths and hop-by-hop local retransmission to ensure reliable transport, on the basis of the original single-path-based algorithm. A new structure of caching is proposed in [16] which is called Active caching (AC). AC provides a tradeoff between end-to-end delays and memory requirements and more reliability.

3. The Chinese Reminder Theorem

In this section we will briefly review the Chinese Remainder Theorem. In the simplest case, this theorem can be described as follows [17]:

Given N primes $p_i > 1$, in which $i \epsilon \{1... N\}$, M will be the primes product, i.e. $M=\Pi p_i$. By assuming m as an original packet, then the set of integers $\{m_1, m_2 \ldots m_N\}$ will be sub-packets, considering the condition m <M, where m can be obtained from (1). Notice that in (2), q_i is c_i modular inverse.

$$m = (\sum_{i=1}^{N} c_i * m_i) \ mod \ M \quad (1)$$

$$c_i = Q_i * q_i \quad (2)$$

$$Q_i = \frac{M}{p_i} \quad (3)$$

$$m_i = mmodp_i \quad (4)$$

For example, consider a system that is m = 64 and we want sent message by the Chinese Remainder Theorem, instead of forwarding complete packet (m), knowing the set of primes $p_i=\{3, 5, 7\}$ that $i \epsilon \{1, 2, 3\}$can split it into three sub-packet by equation (3) and the sub-packets sent by the intermediate nodes to the sink.

Considering the above relations it can be proved that 7 bits are required to show this message, which cannot be considered as more than three bits for each sub-packet. For example in Fig.1, if B, C and D are the received message m from node A, each of them will execute the above method and calculated sub-packets by m_i equation with $i \epsilon \{1, 2, 3\}$ and transmit sub-packets to S node instead of m.

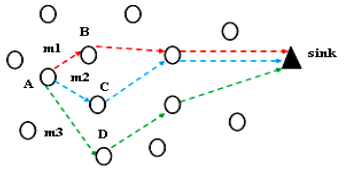


Fig. 1. Example of forwarding the main packet after division

$W_{CRT_{max}}$ Indicates the maximum number of CRT components where w is the number of bits in the original packet.

$$W_{CRT_{max}} = max(\lceil log\ (p_i)\ \rceil \quad (5)$$

In general, splitting packet to several sub-packets with a low number of bits will reduce energy consumption.

The maximum number of sub-packets bits and theoretical maximum energy reduction factor (*MERF*) can be obtained from the following equations:

$$MERF = \frac{(w - W_{CRT_{max}})}{w} \quad (6)$$

Selecting appropriate primes has the most important role in energy storage.

4. The Proposed Approach

Reducing delay, improving energy consumption and maximizing reliable data transmission are commonly discussed issues in WSNs. In many applications, multipath forwarding is used in order to increase both reliability and throughput. Hence, the delay and packet loss ratio will be extremely reduced because of using multiple paths in packet transmission. However, traffic congestion and network overhead increase because of sending several copies of a packet. By splitting the original messages into several sub-packets, each node will be responsible of forwarding a small sub-packet; consequently low power consumption of each node and network balancing will be achieved. The greedy forwarding selects a nearest neighbor node from the source node to the sink as the next forwarding node Storage data packets at some intermediate nodes between the source and

the destination for retransmitting the missing data packet can be an appropriate solution to reduce delay and increase reliability.

The queuing model of the sensor nodes of the proposed scheme is shown in Fig.2. The forwarding mechanism that is used to delivered packets to the destination is described in [18], which is the author's previous paper.

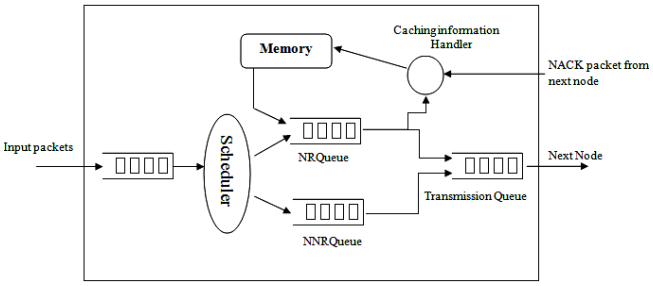


Fig 2. Queuing model of sensor nodes

In order to achieve a high delivery ratio, minimize network overhead and ensure high reliability, caching sub-packets along the multipath forwarding can be used. In the proposed scheme, the splitting procedure that is used to break up the original message is the Chinese Remainder Theorem (CRT). Two parameters play an essential role to detect whether caching is necessary or not. This method considers the reliability of the link rate as a threshold. Probability of Error can be calculated as follows [19]:]

$$POE = \frac{1}{2} * (1 - erf)\sqrt{\frac{E_b}{N_0}} \quad (7)$$

where *erf* is the error function, E_b is a bit of energy derived and N_o is the noise power spectral density.

Each node has the ability to count the number of NACK packets that receive from further nodes [20]. Probability of Caching (POC) is obtained by POE and NON [t-1] which refers to Number of NACK packets that node *n* receives in t-1.

Depending on the type of data that sensors should report to the sink, the Scheduler classifies received packets as need reliability (NR) packet or non-need reliability (NNR) packet and sends them to NRQueue or NNRQueue queues, respectively. By using the algorithm described in Fig.3, the caching information handler can decide whether caching is needed or not. Packet will send the memory of current node to cache for a special period of time if caching condition becomes true.

In WSNs, information that has been collected by the sensors must be delivered in a special time. Otherwise, data will be useless. Successfully forwarding in a packet timeline can be achieved in a balanced network. The proposed method minimizes packet overhead and ensures high throughput by multipath forwarding of sub-packets.

The following equation can be used to calculate the network load balancing when sub-packets are distributed to be forwarded through different multipath [21]:

$$\varphi(\bar{r}) = \frac{(\sum_{j=1}^{N} r_j p_j)^2}{N \sum_{j=1}^{N} (r_j p_j)^2} \quad (8)$$

where φ illustrates load balance ratio, the vector $\bar{r}$ denotes the traffic rates allocated to all available routes, r_j and p_j are respectively the traffic flow and the product of the path cost allocated to path j.

In the proposed scheme, the main packet will split into several sub-packets only in the source node. Equation (9) presents the transmission cost of each node. It reflects both communication overhead and energy-efficiency [7]:

$$TransmissionCost = \frac{T}{U.N} \quad (9)$$

where N represents the average number of hops in the paths, T shows the total bits each node transmits, and U denotes the number of useful bits that the sink received.

By using (8) the reduction of network overhead is obvious.

This paper is going to reduce energy consumption while minimizing the delay. To evaluate the node energy consumption we use following equation:

$$E_T = \frac{\sum_{i=1}^{M} (e_{i,int} - e_{i,res})}{M * N} \quad (10)$$

where M is the number of nodes, $e_{i,\ init}$ and $e_{i,\ res}$ are the initial and residual energy levels of node i, respectively and N is the number of data packets received by the sink.

Energy is a major factor which impacts the network life time. By reducing energy consumption and delay, the network lifetime will be prolonged. The following equation is used in order to obtain network lifetime:

$$NLT = \frac{\sum_{i=1}^{n} (e_{i,res} - e_{i,n})}{e_{int}} \quad (11)$$

where $e_{i,n}$ indicates the energy needed for packet forwarding of node i, and e_{int} represents the initial level of energy.

```
n=1        // current node in the path to destination
Loop n< N
(POE) n ←½ * (1-erf) √Eb/N0
(POC) n←(POE) n + NON [t-1] n
If (POC) n < θ1 and NON [t-1] n< θ2
Forward the packet normally and without caching
n←n+1
Return
Else if (POC) n > θ1 or NON [t-1] n> θ2
First cache the sub-packet and then forward to      next hop
n←n+1
Return
End loop
```

Fig 3. Caching algorithm at *n*-th node

5. Performance Evaluation

In this paper, a homogenous WSN has been considered. For the simulation, we assume 100 sensors that are randomly distributed through the sensing area of 100*100 m^2. Table 1 lists the main parameters of simulation.

We consider three different parameters to evaluate the performance of the proposed method: End-to-End packet delay, energy consumption and successfully delivered packets ratio.

The simulation has been done in two parts; splitting the main packet, respectively into two and three sub-packets. Fig. 4 and Fig. 5 show the performance comparison of AELAR, GEAR, and SPC. Due to the forwarding multipath (equal to the number of sub-packets) with different distances, the average distance will be lower than forwarding from a single path and also whenever the sub-packet is lost, the delay in SPC will be reduced and be lower than the other two protocols because of caching sub-packets in some intermediate nodes and retransmission from those nodes instead of source node as shown in Fig.4 (a) and Fig. 5 (a).

Table 1. Simulation Parameters

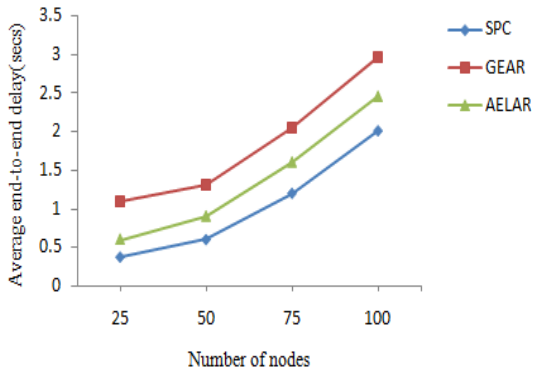
Parameter	Value
Number of nodes	100
Simulation area	100*100
Number of sink	1
Sensor distribution	Uniform random
Location of Sink	top left corner
Radio range	5m
MAC layer	IEEE 802.11
Bandwidth	200KB/S
Initial battery charge	3.3 Joule

To calculate the number of packets that reach the destination, the Data Delivery Ratio can be used [22] and it is express as:

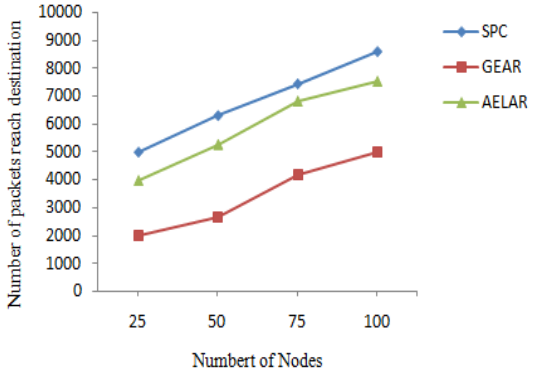
$$\text{Data Delivery Ratio} = \frac{\text{Succesfully delivered data}}{\text{Required Data}} \quad (12)$$

This equation demonstrates both the number of data packets that are sent by the source and the number of data packets received by the sink. In the ideal condition the result should be equal to 1.

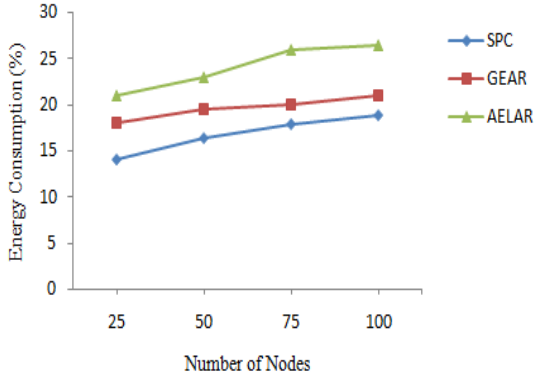
Due to caching in SPC scheme the percentage of packets which can successfully be delivered to the destination will increase. This fact is shown in Fig.4 (b) and Fig.5 (b). By getting closer to the ideal condition, high reliable forwarding will be obtained. By splitting the original message into sub-packets and using multipath for transmission, nodes use less energy while forwarding although the number of nodes that are involved in forwarding operation increased. This is because they are responsible for forwarding only a part of the message instead of the whole message, so energy consumption of the nodes will decrease. By using a CRT-based Packet Splitting Algorithm and splitting the original message into two and three sub-packets, respectively about 71% and 57% of the energy could be saved. The simulation results in Fig.4 (c) and Fig.5 (c) indicate this theme.



(a) Average end-to-end delay

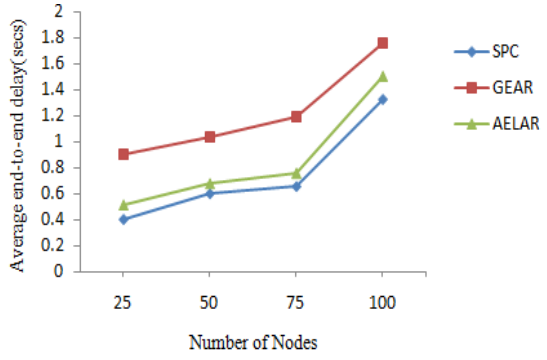


(b) Number of packet that reach destination

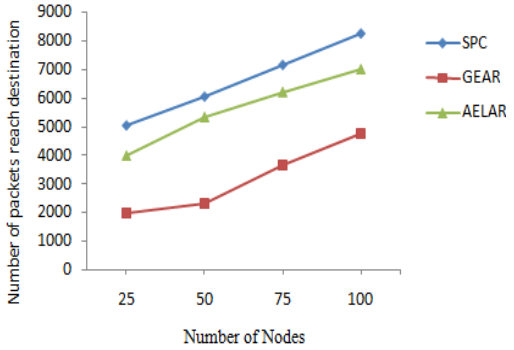


(c) Energy consumption

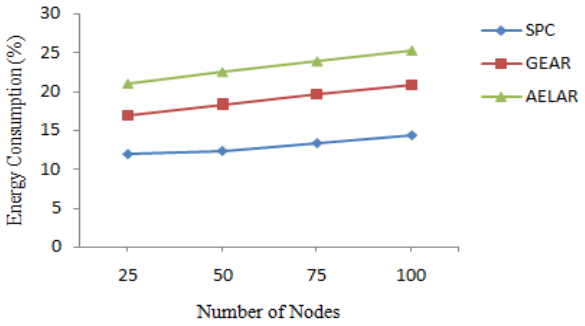
Fig 4. Simulation results for evaluating the performance of Splitting-packet cashing with 2 sub-packets (a, b, c)



(a) Average end-to-end delay



(b) Number of packet that reach destination



(c) Energy consumption

Fig 5. Simulation results for evaluating the performance of Splitting-packet cashing with 3 sub-packets (a, b, c)

6. Conclusion

This paper describes the splitting of original packets into several sub-packets based on the CRT algorithm, and uses the multipath to forward these sub-packets to the destination. A new Caching approach is proposed which considers two parameters; link error that is calculated as the probability of error and numbers of NACK packets that the current node has received from the next node in past times. The proposed scheme that is called SPC optimizes the tradeoff between the reliability and energy-efficiency, minimizes the network overhead, increases packet reception ratio and prolongs network lifetime.

References

[1] Akyildiz I, Su W, Sankarasubramaniam Y, Cayirci E. Wireless sensor networks: a survey. Computer Networks, 38:393–422, 2002.

[2] Yao-Nan Lien "Hop-by-Hop TCP for Sensor Networks", *International Journal of Computer Networks & Communications (IJCNC),* April, Vol.1, No.1, 2009.

[3] Paulo Rogério Pereira, António Grilo, Francisco Rocha, Mário Serafim Nunes, Augusto Casaca, Claude Chaudet, Peter Almström and Mikael Johansson," End-To-End Reliability in Wireless Sensor Networks: Survey And Research Challenges" EuroFGI Workshop on IP QoS and Traffic Control, P. Pereira (Ed.) Lisbon, Portugal, December 6-7, 2007

[4] J.-H. Hong, C.-H. Wu, C.-W. Wu "RSA Cryptosystem Based on the Chinese Remainder Theorem", *Proc. of Asia and South Pacific Design Automation Conference (ASP-DAC),* Yokohama, Japan, January 2001.

[5] Ganesan, D.; Krishnamachari, B.; Woo, A.; Culler, D.; Estrin, D.; Wicker, S. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks; Technical report; CS TR 02-0013; UCLA: Los Angeles, CA, USA, 2002.

[6] Zhao, J.; Govindan, R. Understanding Packet Delivery Performance in Dense Wireless Sensor Networks. *In Proceedings of ACM International Conference on Embedded Networked Sensor Systems, Los Angeles,* CA, USA, November, 2003; pp. 1–13.

[7] Jaewan Seo, Moonseong Kim, In Hur, Wook Choi and Hyunseung Choo "DRDT: Distributed and Reliable Data Transmission with Cooperative Nodes for Lossy Wireless Sensor Networks", 10, 2793-2811, Sensors 2010.

[8] Ganesan, D.; Krishnamachari, B.; Woo, A.; Culler, D.; Estrin, D.; Wicker, S. Complex Behavior at Scale: An Experimental Study of Low-Power Wireless Sensor Networks; Technical report; CS TR 02-0013; UCLA: Los Angeles, CA, USA, 2002.

[9] Huang, X., Fang, Y.: Multi constrained QoS

Multipath Routing in Wireless Sensor Networks. ACM Wireless Networks (WINET), 2007.

[10] Couto, D.S.J.D.; Aguayo, D.; Bicket, J.C.; Morris, R. A High-throughput Path Metric for Multi-hop Wireless Routing. *Wirel. Netw. 11*, 419–434, 2007.

[11] Emad Felemban, Student Member, Chang-Gun Lee, Member, and Eylem Ekici" MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks", *IEEE TRANSACTIONS ON MOBILE COMPUTING,* JUNE 2006 VOL. 5, NO. 6

[12] Cao, Q.; Abdelzaher, T.F.; He, T.; Kravets, R. Cluster-Based Forwarding for Reliable End-to-End Delivery in Wireless Sensor Networks. In *Proceedings of IEEE International Conference on Computer Communications*, Anchorage, AK, USA, May, pp. 1928–1936, 2007.

[13] C. Kang, X. Shangkon, Sh. Jinglun and W. Gang "An Energy-efficient Multiple Paths Routing Algorithm for Wireless Sensor Networks", *IEEE, ICCS*, 2008.

[14] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: a tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev,* vol. 36, no. SI, pp. 131 146, 2002.

[15] Yuhua Liu and Hao Huang Kaihua Xu "Multi-path-based Distributed TCP Caching for Wireless Sensor Networks", *IEEE DOI,* 2007.

[16] Dae-Young Kim and Jinsung Cho" Active Caching: A Transmission Method to Guarantee Desired Communication Reliability in Wireless Sensor Networks", *IEEE COMMUNICATIONS LETTERS,* JUNE, VOL. 13, NO. 6, 2009.

[17] G. Campobello, A. Leonardi, S. Palazzo "On the Use of Chinese Remainder Theorem for Energy Saving in Wireless Sensor Networks." *Proc. of IEEE International Conference on Communications (ICC 2008)*, Beijing, China, May 2008.

[18] Ali Ghaffari, Maryam Kordlar, Vida Aghakhanloyetakanloo, "Energy-efficient multipath data forwarding in wireless sensor network", *Australian Journal of Basic and Applied Sciences*, 5(8): 523-529, ISSN 1991-8178, 2011.

[19] M. R. Ebenezar Jebarani1and T. Jayanthy," An Analysis of Various Parameters in Wireless Sensor Networks Using Adaptive FEC Technique", *International Journal of Ad hoc,* Sensor & Ubiquitous Computing (IJASUC), September, Vol.1, No.3, 2010.

[20] Levente Buttyan, Ant´onio M. Grilo "A Secure Distributed Transport Protocol for Wireless Sensor Networks", *Australian Journal of Basic and Applied Sciences,* 5(8): 523-529, 2011.

[21] Ye Ming Luz and Vincent W. S. Wong, "An energy-efficient multipath routing protocol for wireless sensor networks", *International Journal of Communication Sytems,* 20:747–766, 2007.

[22] R Vidhyapriya, Dr P T Vanathi, "Energy Efficient Adaptive Multipath Routing for Wireless Sensor Networks", *IAENG International Journal of Computer Science,* 34:1, IJCS_34_1_8, 2006.

Journal of Computer and Knowledge Engineering, Vol. 1, No.2, 2017.
DOI: 10.22067/cke.v1i2.63668

A Novel Routing Algorithm for Mobile ad-hoc Networks Based on Q-learning and its Generalization to FSR Routing Protocol

Mahmoud Alilou Abdolreza Hatamlou*

Abstract: This study proposes a novel routing algorithm using Q-learning. Q-learning is a machine learning (artificial intelligence) algorithm using the reinforcement learning policy which can be used to solve problems for which there are different ways to reach their goal. The proposed algorithm, the Modified Q-learning routing algorithm (MQRA), has eliminated the episodes of Q-learning required to gradually learn in different stages and this has made it a rapid routing algorithm. MQRA can be used in various types of networks. This study uses MQRA in mobile ad-hoc networks, its generalization to fisheye state routing (FSR) (a routing algorithm) and its performance results are compared with the standard FSR. Experimental results confirm the applicability and potential of the proposed algorithm.

Keywords: Routing Algorithm; Mobile ad-hoc Networks; FSR Protocol; Reinforcement Learning; Routing in MANETs.

1. Introduction

In mobile ad-hoc networks (MANETs), nodes or workstations, trying to send information, are constantly moving and their neighbors are always changing. Thus, finding the current position of each node and the path to send the information packages has become one of the most important problems of such networks. Various routing algorithms use different methods to find a route and reinforce a particular component based on the policy used in sending information packages through the network. Thus, it is natural that concentrating on reinforcing one parameter would lead to distraction from the weakness of other ones. For instance if rapid delivery of the packages is important in a network, the routing algorithm loses more time in the routing phase, while finding the shortest path or it may have to select either an unsure short path or a safe long path to send the information packages. Another significant problem of mobile networks is the energy consumption of the nodes to process, store and send packages through the network. A light and intelligent algorithm can mitigate power consumption of the network and affect its lifetime with a given amount of energy. Making an algorithm more intelligent usually requires more information about the network, more computation makes the algorithm more time complex and usually it is not possible to make an algorithm both light and intelligent. Therefore, the proposed algorithm is presented to find the optimized route with the least possible amount of computation and in a shorter time than other similar algorithms [1, 2].

Q-learning Algorithm

- Input:State space S, Act.Space A
- Discount 7(0<=7<1);Learning a(0=<a<1)
- Outputs:Q
- Repeat{
  - S=get_current_word_state()
  - a=pick_next_action(Q.s)
  - (r.s)=act_int_world(a)
  - Q(s.a)=Q(s.a)+α*(r+7*max_$a'(Q(s'.a'))$-Q(s.a))
- } Until (bored)

Fig 1. The standard Q-learning algorithm

2. Literature Review

MQRA significantly changes Q-learning as shown in Fig. 1 and shows great performance while routing. Standard Q-learning executes stages to obtain a path between the origin and the destination which is the shortest path between those nodes. The proposed algorithm eliminates all stages to find this path as shown in Fig. 2 and finds all paths between the origin and the destination in a shorter time than the standard Q-learning instead of just one path.

```
// MQRA Algorithm
Repeat {
 for (i=0;i<dim_X;i++)
  for (j=0;j<dim_Y;j++)
   if(Route_table[i][j] != 1.0) {
    max_value = α * maximum_adjacent_node_value();
    Route_table[i][j] = max_value;
   } //End if
} Until (converge)
```

Fig 2. MQRA for a mesh network.

In the equation shown in Fig. 2, α is a variable which can vary in relation to bandwidths of the links of each node. Thus, instead of computing short paths by the number of steps, optimized paths are obtained by the amount of transferred data. However, for the simplicity of the example here, α is considered constant and equal to 0.98.

We describe the algorithm by an example of a given network. Take an 8x8 mesh, thus we have an 8x8 routing table and we assume that node (5, 5) is the destination and node (0, 1) is the origin. Therefore, the value of (5, 5) in the routing Table is 1 and the rest of the Table has zero values.

Manuscript received April 10, 2017; accepted November 27, 2017.
M. Alilou, Department of Computer Science, Khoy Branch, Islamic Azad University, Khoy, Iran.
*A. Hatamlou, Dept. of Computer Science, Salmas Branch, Islamic Azad University, Salmas, Iran. hatamlou@iaukhoy.ac.ir.

We start from slot 0 of the Table, move from left to right, top to bottom and set the value of each slot as α multiplied by the maximum value of its neighbors. If the neighbors with maximum values are all equal, one of them is selected by default.

Table 1. The initial routing table.

0	1	2	3	4	5	6	7	
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	3
0.00	0.00	0.00	0.00	0.00	0.98	0.96	0.94	4
0.00	0.00	0.00	0.00	0.98	1.0	0.98	0.96	5
0.00	0.00	0.00	0.00	0.96	0.98	0.96	0.94	6
0.00	0.00	0.00	0.00	0.94	0.96	0.94	0.92	7

In Table 1, first values of all slots equal to 0, except for the destination. After the first run of MQRA, this table is changed as shown in Fig. 3 and after about 8 iterations of this algorithm, the routing table is converged and all paths between the origin and the destination are obtained as shown in Fig. 4.

Table 2. The converged routing table.

	0	1	2	3	4	5	6	7
0	0.81	0.83	0.85	0.86	0.88	0.90	0.88	0.86
1	0.83	0.85	0.86	0.88	0.90	0.92	0.90	0.88
2	0.85	0.86	0.88	0.90	0.92	0.94	0.92	0.90
3	0.86	0.88	0.90	0.92	0.94	0.96	0.94	0.92
4	0.88	0.90	0.92	0.94	0.96	0.98	0.96	0.94
5	0.90	0.92	0.94	0.96	0.98	1.0	0.98	0.96
6	0.88	0.90	0.92	0.94	0.96	0.98	0.96	0.94
7	0.86	0.88	0.90	0.92	0.94	0.96	0.94	0.92

3. The FSR protocol

FSR is the reinforced protocol of GSR (both of which are based on the link state). Updating messages uses a significant amount of the bandwidth in GSR.

Fig. 3 presents an example of the fisheye boundary for node circles in red. This boundary is determined by the number of hops needed to reach a certain node. Not all updating messages of FSR contain all the information of the nodes. However, more information is provided about closer nodes than the farther ones. This decreases the size of updating message. The information of a node about its neighbors is updated frequently. However increasing the distance mitigates information validation. This process of dividing the network to different boundaries is performed for each node which means that there is no central node responsible for this division [4].

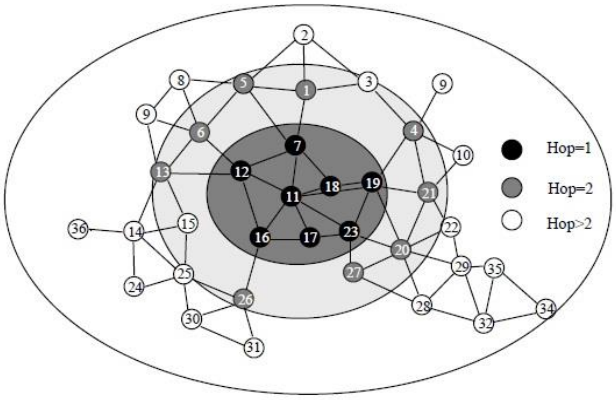


Fig 3. Boundary of the fisheye [3].

Despite the invalidity of information related to far neighbors, the routing procedure works correctly since approaching the destination increases the precision of information. This protocol is suitable for large-scale networks, since the protocol overhead is controlled.

The fisheye state routing protocol is table-driven (a proactive routing protocol). As it was mentioned, FSR is based on link state routing and it is able to provide the path information when needed. The link state package is exchanged periodically, not event-driven and the topology table is only sent to local neighbors instead of propagating in the entire network. The order of numbers is used to arrange the rows of the table, so that no row has the same number and thus routing is done with no cycles.

Updating messages of the nodes in smaller boundaries are more precise, since they send their routing tables more frequently; that is, nodes close to each other receive tables more frequently. However, the precision of farther nodes is mitigated, since it takes longer to exchange tables. Nonetheless, there is no need to find the path as done in demand based routing algorithms.

The fisheye boundary enables sending link state messages to nodes in different locations of the fisheye boundary in different time intervals. This leads to reducing the size of the link state packages.

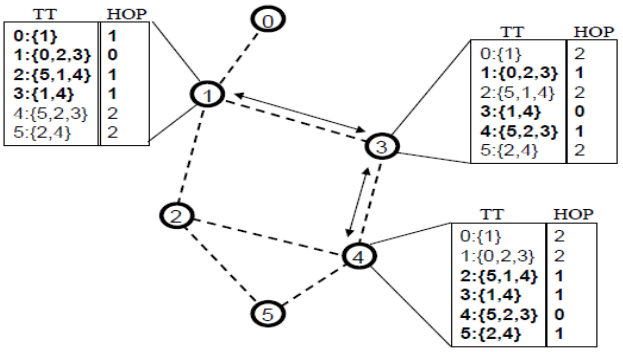


Fig 4. Reducing the message using fisheye [3, 5].

The highlighted row of table.12 is propagated more frequently to its neighbors since it has less hops. The TT column presents the neighbors.

Each node in FSR has the following information: The topology table, the link state list of neighbors, the routing table, Pros and Cons. The topology table is created using the

information from link state messages. Each node has one slot in this table (the entire topology map). Each slot consists of two parts: the link state information and the destination order number. The routing table is created according to the information of this table. Information related to distance is obtained after creating the routing table and its information is used to determine the fisheye boundary for a node.

The topology table has the following information in each row: destination address, destination order number and the link state list. While receiving a link state message, the receiving node registers or updates the sender in its neighbors list. If it receives nothing from its neighbor after a certain time out, the corresponding row is deleted from the neighbors list. Each node stores the link state and the last time stamp of its neighbors. The routing table provides the next hop information to send the package to its destination. The rows of this table are varied if the topology table is changed. The rows of the routing table present the destination address and the next hop address.

FSR is suitable for large mobile networks since it is not sensitive to link malfunction through control messages. The malfunction links are not considered in exchanging the next connection messages and that means link changes do not necessarily change the routing tables. FSR is a simple method due to using the shortest updated paths. It is also a robust method due to exchanging a part of the updated message with only its neighbors which reduces traffic.

It is easy to find the destination, since the topology map and a simple addressing scheme is used. The drawback of this protocol is the complicated storage of the routing table, the computation overhead and also its inability to provide security as other protocols do [4-7].

3-1. Generalizing MQRA to FSR

In MQRA generalized to FSR which we call "MQ-FSRA", each node sends its score equal to 1% of its value together with its ID to all direct neighbors. Each receiving node stores the ID, the score and sends a percentage of the received score, as well as the ID of the first node to its neighbors.

Consider two nodes labeled A and Z in Fig. 7. A node which has a score coefficient equal 1.0 sends its label and a percentage of its score to its direct neighbors. All neighbors repeat the same so that A is identified in the entire network. All nodes, e.g. Z, do the same to be identified in the network. Fig. 5 is designed assuming A as centroid and each node like A belongs to an area with its corresponding centroid. The areas shown represent the frequency of sending packages. This means that for instance A sends its information, including label, a percentage of its score and other necessary information, more frequently in a limited area highlighted in the Figure. This frequency is reduced for farther areas and information packages are sent less frequently.

To clarify this point, pay attention to the eighth node in Fig. 7. This node, as it was considered beforehand, has saved some coefficients to send packs to nodes A and Z. the closeness coefficient of node 8 to node A is 0.94. This means that in case node 8 is supposed to send a pack to A, it must be sent through a neighbor that has a higher closeness coefficient to A than itself.

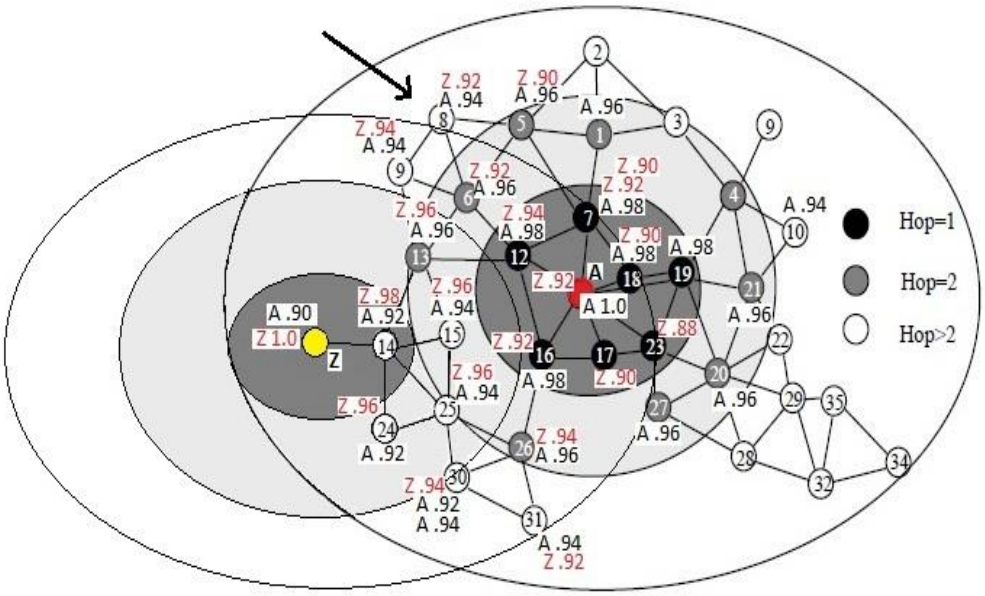


Fig 5. Routing in MQ-FSRA

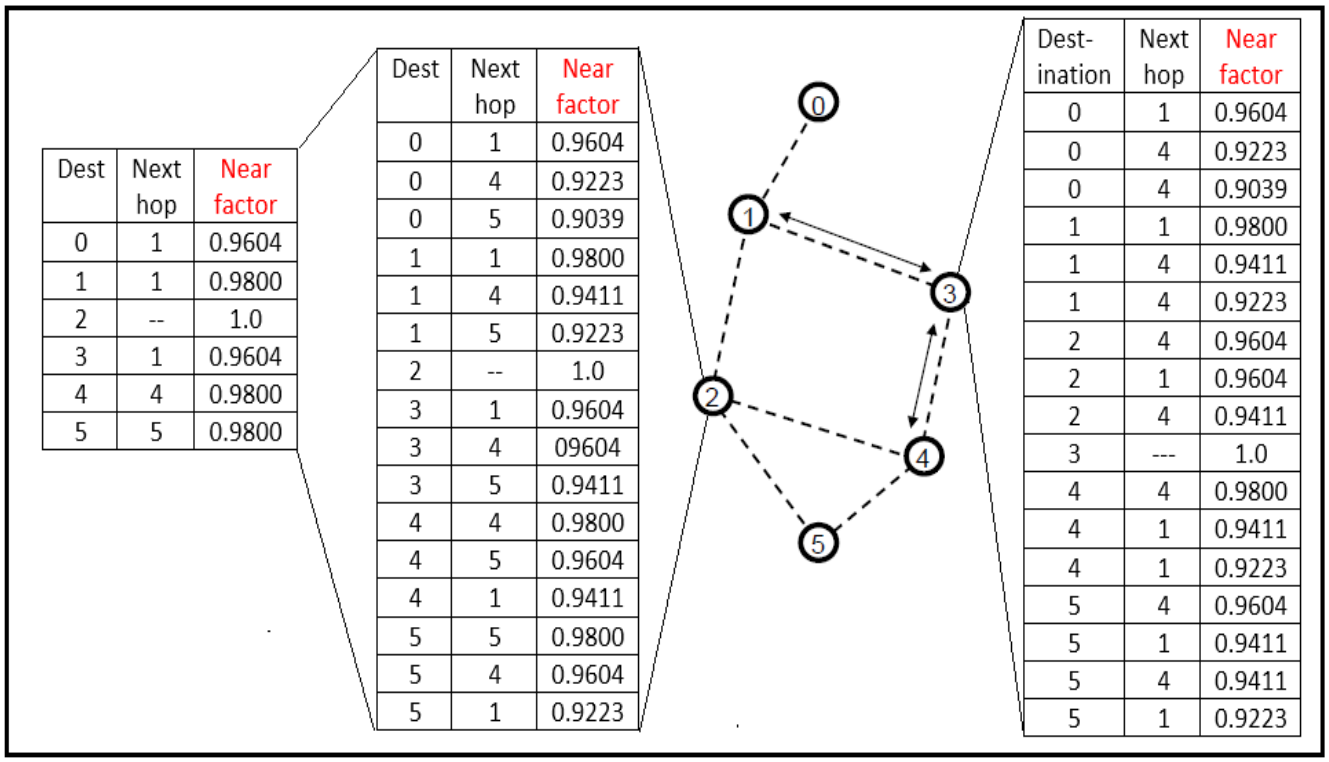


Fig 6. The navigation table structure in MQ-FSR.

In the example above, node 8 must choose node 6 or 5 in the next step to convey a pack to A, and if it wants to send a pack to node Z, the next step will be sending a pack to node 9. In this way, dispatch of any pack from any point in the net to the desired destination is possible through the shortest way.

With close attention to Fig. 5 we will notice that MQ-FSR easily supports Multi Pathing without calculating and saving any extra data in comparison with single pathing.

In order to dispatch any pack to destination, any node might simultaneously choose various neighboring nodes with higher closeness coefficients and choose one of the neighbors under the same circumstances. Choosing a neighbor can take place haphazardly or intelligently. The next step, for instance, can be based on the battery level in case some neighbors are under the same conditions. Namely, to choose the node among the neighbors that has a higher battery supply or to choose the neighboring node that has fewer tasks in buffer queue awaiting to be processed.

Pay attention to node 2 in this hypothetical net. As it can be seen in contrast with FSR list, the neighbors of other nodes are not saved in the table of this node. Moreover, instead of the number of paces, the closeness coefficient to the destination is mentioned. In this table, in case there are multiple routes to a destination. The longer ones with lower closeness coefficients can be eliminated and an optimum table can be produced by reduction of navigation table rows.

Therefore, less amount of information is propagated through the network. However, routing is performed correctly as mentioned before. This is so because as packages approach their destinations, the information related to the destination becomes more precise and the package is guided to its destination.

3-2. The Advantage of MQ-FSRA to FSR

As we can see in Fig. 6, FSR always propagates the network topology to direct neighbors of each node with different frequencies. This makes two problems arise which are not inherent to MQ-FSRA. First, each node must store its direct neighbors and a list of neighbors of other nodes. This makes each node identify its neighbors through sending and receiving packages and sending collected information to other nodes which consumes a significant amount of the energy of the network. Second, there is the problem of node dependencies; that is, nodes must try to send their packages by their neighbors whose information is propagated through the entire the network. This means an implicit dependency between nodes which makes the network update itself frequently. This is more intensified when velocities of nodes are great or nodes often fail. Obviously, none of these two problems exist in MQ-FSRA, since no list of neighbors is sent, there is less amount of information, there is no need to collect the information related to neighbors and there is no dependencies between neighbors. Wherever nodes are located in the network, they only have to send their information to one node which has a higher score.

4. Simulation Results

In the experiments that were conducted, nodes are mobile and the amount of their mobility is 0.5ms. The amount of energy is limited and the same for all nodes. The network space is 100x1000 and the number of nodes varies depending on the experiment; that is new nodes can join or leave the network during its lifetime. All facilities of the nodes are the same, wireless transmission is used for sending and receiving with a maximum bandwidth of 50Mbit/s and the radius of 30m.

In this section, we evaluate MQ-FSRA with important policies of routing protocol evaluations, i.e. average routing overhead and average package loss, and also compare the results with FSR. The following simulation results indicate that MQ-FSRA provides better results using these policies.

Routing overhead is the ratio of total number of sent control packages to total data packages received successfully at the destination. Figs. 7 and 8 present average routing overhead and average packet loss of FSR and MQ-FSRA. These diagrams consider the overhead amounts separately according to the number of current nodes of the network and the velocity of the nodes. Also average package lost is given based on node failures. As we can see in Figs. 7 and 8, MQ-FSRA has better performance compared with FSR, through reduction of navigation table rows.

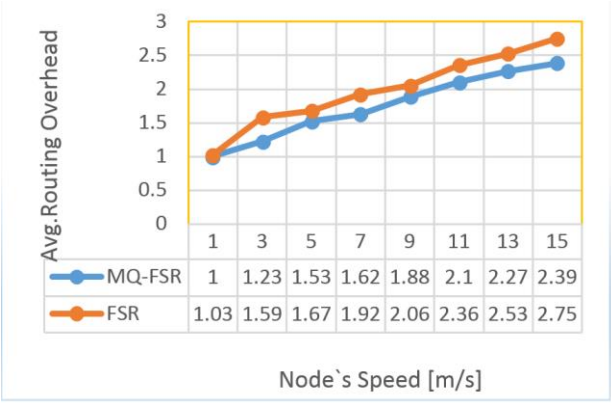


Fig 7. Average routing overhead according to cost functions with different number of nodes.

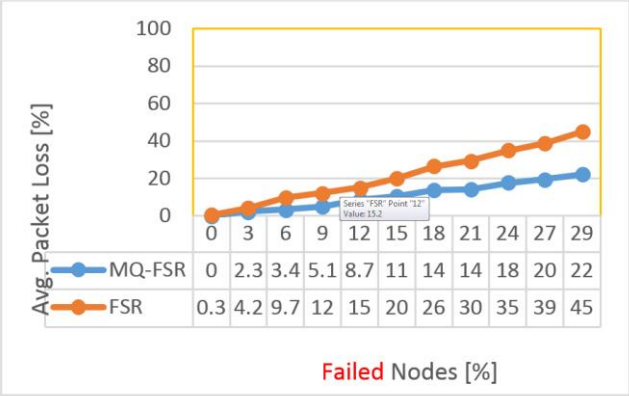


Fig 8. Average package loss according to the proportion of node failures.

5. Conclusions

As it was discussed in this study, the novel proposed algorithm called "MQRA" is a light and rapid algorithm, which can adapt to the environment and it can also be generalized to various protocols. Generalizing the proposed algorithm to FSR significantly reduces FSR computations and eliminates node dependencies. This leads to long-term updates of MQ-FSRA and imposes less overhead in the path finding phase and routing reconfiguration. Node independency reduces package loss due to node failures and path disconnection while sending the package. MQ-FSRA needs less amount of stored information to find a route compared to FSR since in contrast to FSR, MQ-FSRA does not need to store the information about direct and indirect neighbors. Therefore, it requires less memory and consumes less energy. Adding facilities like GPS to MQ-FSRA enables provision of new services which we intend to discuss in another paper. However, FSR does not predict utilizing such capabilities.

References

[1] R. S. Sutton and A. G. Barto, ―Reinforcement Learning: An Introduction,The MIT Press, 1998.

[2] A. F¨orster, ―Machine learning techniques applied to wireless ad-hoc networks: Guide and survey, *// in Proceedings of the 3rd International Conference on Intelligent Sensors,* Sensor Networks and Information Processing (ISSNIP), 2007.

[3] G. Pei, M. Gerla and T.-W. Chen, "Fisheye State Routing in Mobile Ad-Hoc Networks", *In Proceedings of the 2000 ICDCS Workshops,* Taipei, Taiwan, pp. D71-D78, Apr. 2000.

[4] L. Kleinrock and K. Stevens, "Fisheye: A Lenslike Computer Display Transformation", *Technical report, UCLA, Computer Science Department,* 1971.

[5] T.-W. Chen and M. Gerla, "Global State Routing: A New Routing Scheme for Ad-hoc Wireless Networks", *In Proceedings of IEEE ZCC'98,* Atlanta, GA, pp. 171-175, Jun. 1998.

[6] Gyanappa A. Walikar, Rajashekar C. Biradar , A survey on hybrid routing mechanisms in mobile ad hoc networks PP. 48-63. doi>10.1016/j.jnca.2016.10.014

[7] Fahimeh Dabaghi, Zeinab Movahedi, Rami Langar, A survey on green routing protocols using sleep-scheduling in wired networks, PP. 106-122. doi>10.1016/j.jnca.2016.10.005.

[8] Shafiee, Kaveh, and Victor Leung. Connectivity-aware minimum-delay geographic routing with vehicle tracking in VANETs. Ad Hoc Networks 9.2 (2011): 131-141.

[9] Ding, Zhiguo, and Kin K. Leung. Cross-layer routing using cooperative transmission in vehicular ad-hoc networks. Selected Areas in Communications, IEEE Journal on 29.3 (2011): 571-581.

[10] Sofra, Nikoletta, Athanasios Gkelias, and Kin K. Leung. Link residual-time estimation for VANET cross-layer design. Cross Layer Design, 2009. IWCLD'09. Second International Workshop on. IEEE, 2009.

[11] Al-Rabayah, Mohammad, and Robert Malaney, "A new hybrid location-based ad hoc routing protocol", *Global Telecommunications Conference (GLOBECOM 2010),* 2010 IEEE. IEEE, 2010.

[12] Al-Sultan, Saif, Ali H. Al-Bayatti, and Hussein Zedan, "Context-aware driver behavior detection system in intelligent transportation systems", *Vehicular Technology*, IEEE Transactions on 62.9 (2013): 4264-4275.

[13] Yang, Qing, et al. "ACAR: adaptive connectivity aware routing protocol for vehicular ad hoc networks", *Computer Communications and Networks*, 2008. ICCCN'08. Proceedings of 17th International Conference on. IEEE, 2008.

[14] Schaul, T. Bayer, J. D. Weirstra, Sun, T. "PyBrain", *Journal of Machine Learning Research*, vol 11, no.2, 2010.

[15] Kulkani, S., Rao, R., "Performance Optimization of Reinfrocement Learning Based Routing Algorithm Applied to Ad hoc Networks", *International Journal of Computer Networks and Communications*, vol. 2, pp. 46- 60, 2010.

Journal of Computer and Knowledge Engineering, Vol. 1, No.2, 2017.
DOI: 10.22067/cke.v1i2.66227

STAR: Improved Algorithm based on Sliding Window for Trust-Aware Routing in WSNs

Mouhebeh Sadat Katebi* Hassan Shakeri
Farzad Tashtarian

Abstract: *This research with attention to the establishment of trust in WSNs and with the goal of increment in energy supply and growth in malicious node detection accuracy by using of improved sliding window, is saving energy by using computation of the previous periods. Also, this research calculates the trust in aspect of transferring information, based on subjective logic model and incremented the detection rate of malicious node by proposing two algorithms for identifying these nodes. Then, this method increases the speed of routing. The results of simulation of STAR compared to EDTM (Jiang, et al., 2015) shows 11.99% increment in the residual energy of network and growth of 1.52% in detecting the accuracy of malicious nodes.*

Keywords: Trust - Wireless Sensor Network – Subjective Logic Model – Sliding Window – Confidence - Routing

1. Introduction

Since the wireless sensor networks spread everywhere and they are being targeted by many security attacks, it is necessary to provide and maintain security in these networks. Moreover, older security measures should be replaced by new ones. Trust establishment between nodes must be able to evaluate trust among all nodes because the survival of wireless sensor networks is dependent on the trust between the nodes and their participation.

On the other hand, algorithms of trust based systems, as computational loads go up, the nodes would experience lack of energy and life time reduction. Therefore, while designing trust based networks, we must consider the factors such as amount of energy, computational limits, and node and memory limitations in order to build a fully efficient network.

Can be expressed, in general, establishing trust in wireless sensor networks is down with two main objectives: to improve cooperation and increase security (Ishmanov, et al., 2015). Because of the collaboration between sensor nodes in wireless sensor networks, it is crucial to maintain the operation of a network. Also, the trust can be as an important component to gain confidence to obtain data in WSNs.

2. Material and methods

In this study, the trust computation system is implemented on WSNs using the improved sliding window and the Subjective Logic (SL) model. The manner of calculating the trust is based on three factors such as residual energy, node behavior and interval of sending and receiving. The residual energy factor, because of being considered as an important feature in node ability, runs a very important role in establishing trust. Node behavior would be determined based on its suspicious behavior or by node's failure to send (transfer) packets to the next nodes. Also, the interval of packet sending/receiving factor is considered for investigating the delay in receiving the packet. To detect the malicious nodes and prevent them to be selected in the routing process, two different algorithms were introduced with the goal of distinguishing two types of different behaviors. The first algorithm is to examine the tapering of the trust node and the second is considered the node cooperation in sending messages to the neighbor node.

This study implemented the trust computation system on WSNs by using the improved sliding window and SL model (Josang, et al., 2006). In this section, the concepts that are used in this research are defined.

2.1. Preliminary definitions

2.1.1. Trust

From the perspective of decision-making, trust means "the desire to confide". From this perspective, trust can be defined as follows:

Trust is the willingness to rely on something or someone in a certain situation which is characterized by a sense of relative security at the same time. Also, it will consider the possible implications (Gambetta, 1988).

This definition, in addition to the level of destination trustworthiness, considers the conditions that must be taken in decision-making. This includes numerous other options for cooperation, risk or profit and loss of interaction or lack of interaction.

2.1.2. Trust-aware routing

In recent years, some protocols have been proposed for trust-base routing. In this routing method, the focus is on the

Manuscript received ; July 20, 2017. Accepted October 30, 2017.
* M. S. Katebi, Dept. of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran, mhb.s.katebi@mshdiau.ac.ir.
H. Shakeri, Dept. of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran, shakeri@mshdiau.ac.ir.
F. Tashtarian, Dept. of Computer Engineering, Mashhad Branch, Islamic Azad University, Mashhad, Iran. mhb.s.katebi@mshdiau.ac.ir.

trustworthiness of nodes participating in the routing process and it tries to make the main goal which is the security of path by observing the certain constraints of any network such as energy constraint. On this basis and considering the fact that WSNs is one of the most popular networks for implementing the Smart City; trust-aware routing protocols will have a critical role in the security of this network.

2.1.3. Confidence

Believe to the truth of trust estimation.

The difference between trust and confidence: trust is taken into account as an agent, but confidence is about the value of trust evaluation.

2.1.4. Sliding window

The window that includes several slices which in any of it, in every slicing time, a value is placed. With arriving a new value, the oldest value will be removed from the window.

2.1.5. Piggybacking

It is a bidirectional data transmission in the network layer (in the OSI model). This approach is prompted by the data frames (that sent from receiver to sender) to add an acknowledgment that indicates that the data frame is received by the receiver successfully (ACK) and this means that instead of sending a separate ACK package, this package adheres over the data that are scheduled to be sent.

2.2. Related Work

Nowadays, there are different studies in trust-based systems domains in WSNs, especially in the trust base routing field. For encrypting communications between the two nodes and the routing packets in WSNs, in 2008, a dynamically and symmetric key distribution method was proposed by Lewis et al (Lewis, et al., 2008). In this method, each sensor node can distribute the common key. Each node can choose one of the neighboring nodes to distribute a key pair to communicate between two nodes. This selection is based on the local computing of obtained trust value from the requesting node. In this study, we present a trust-based routing method that each node employs this method in its routing. Use of direct and indirect trust, has a long history. A trust-aware routing that contains distributed trust model is based on direct and indirect trust information proposed in 2010. The innovation of this trust-aware routing algorithm is a defense against wide attacks using supervision method and awareness from energy. This also leads to better load balancing and more flexibility against attacks.

In 2012, for solving the loss of data integrity problem that was caused by invalid data injection by unauthorized nodes in the network, the researchers (Chakrabarti, et al., 2012) proposed a model that introduces a three layer architecture based on trust framework to also detect unauthorized nodes from authorized ones and to separate fake data from others. The base station keeps the sequence of nodes trust values and also the nodes in a cluster save a sequence of trust values of their cluster heads. According to the study, the trust value is affected by energy value, receiving data model (binary or possibility) and a difference in the received data from a node in comparison with the received data from neighbor nodes. With the aim of finding the trusted nodes and running the routing on these nodes, in 2014, the plan co-worker (Latha & Palanivel, 2014), introduced a secure routing algorithm that detected and ranked the trusted node by using the packet Message Authentication Code (MAC) model and it also gives a chance to the untrusted node to show its honesty. The model provides security features with high performance and a minimum of overload. In the first stage, the source node achieves the MAC value by the message secret code and sends it to its neighbor node. The neighbor node achieves the received message MAC value by the same key. Then both of MAC values would be compared with each other. If each of both values meet together, ACK message would be sent to the sender. If the sender did not receive the ACK message, its neighbor node would be placed in a different list as an untrusted node. In the second stage, the nodes that were detected as trusted are ranked based on the number of packets they send and also the quality of their behavior with other neighbors. In the third stage, as soon as the source node receives a signal, it starts the routing process for its packets using predetermined nodes, and then ranking would be done based on the second stage. In the fourth stage, the trustworthiness of the nodes that are kept in the list would be reevaluated. In another study, in 2014, the requirement for routing protocols based on trust was studied in research (Vasudha & Gajkumar Shah, 2014). In this study, the nodes that have low security are detected and then are equipped with the defense system by using clustering methods and path discovery algorithms that are based on trust.

In 2015, in a study of security threats and energy constraints in multi-hop WSNs, (Raza, et al., 2015) proposed an energy storage routing protocol that is safe and based on trust. This protocol supervised the trustworthiness and reputation of nodes and it maintains a history of interactions between nodes to determine the safety and trustworthiness of the paths. The protocol has three phases for detecting the neighbors, cluster, head selection and data sharing. In 2015, a trust framework was designed for secure routing in WSNs by (Hoceini, et al., 2015), that is based on network architecture structure. This approach can effectively reduce the costs of trust evaluations and guarantee the selection of most secure paths leading to the base station. The protocol evaluates the trustworthiness of sensor nodes according to acknowledges of the base station and the recommendations of the neighbor nodes. To protect WSNs against multi-hop path corrupter attackers, (Chavan, et al., 2015) a model that has a strong framework was implemented to trust-aware routing for WSNs, in 2015. This model provides an energy-efficient and trustworthiness path without synchrony and GIS. More importantly, this model has an efficient definition against attacks that produce fake ID. In 2015, in terms of energy saving and guaranteeing the secure data transmission for WSNs, a routing protocol based on a potential field and trust was proposed by (He & Zhao, 2015). For node and valid cluster head selection process, considered three factors such as residual energy, trust value and distance.

In 2015, Jiang et al. introduced a distributed trust model for WSNs by the acronym EDTM (Efficient Distributed Trust Model), that is a trust model which includes two components of one-hop trust model and multi-hop trust model, (Jiang, et al., 2015). In the one-hop model, if trust

value is completely achieved by direct experiences of node A with node B, this model is called direct trust, otherwise, a recommendation trust model is built. In the multi-hop model, when node A received a recommendation from other nodes for node B, then the indirect trust model is built. The direct trust between two neighbor nodes is calculated using Equation 1:

$$T_{n-direct} = W_{com}T_{com} + W_{ene}T_{ene} + W_{data}T_{data} \qquad (1)$$

where W_{cam} , W_{ene} and W_{data} are the weight of communication, energy and data trust, respectively.

Authors introduced Equations 2 and 3, to calculate the indirect trust in two steps of finding the multi-hop recommenders between two nodes and trust propagating.

$$T_{n-indirect}\binom{B}{C_1} = \begin{cases} T_{C_1} * T_{C_1}^{B}, & if\ T_{C_1}^{B} < 0.5 \\ 0.5 + (T_{c_1} - 0.5) * T_{c_1}^{B}, & else \end{cases} \qquad (2)$$

$$T_{n-indirect}\binom{B}{C_{i+1}} = \begin{cases} T_{C_{i+1}} * T_{n-indirect}\binom{B}{C_i}, & if\ T_{n-indirect}\binom{B}{C_i} < 0. \\ 0.5 + (T_{c_{i+1}} - 0.5) * T_{n-indirect}\binom{B}{C_i}, & else \end{cases} \qquad (3)$$

In 2016, researchers (Kaur, et al., 2016) suggested a trust-based key management routing framework in WSNs that creates a secure and trustworthiness path dependent on the current and past interactions. Then, the path is updated by the separation the malicious nodes or vulnerable nodes. In this model, the network parameters can be determined such as the network deployment area, the number of nodes, the rate of the malicious node in the network (if the effect of attacks and transmission range of a node are taken into account). Using a distributed trust model for the discovery and the separation of the misbehaving nodes, a secure and energy/trust-aware protocol was introduced by (Ahmed, et al., 2016) in 2016. This protocol uses a polymorphism routing method which considers trust level, residual energy and hop number of neighbor nodes, during the routing. This method not only guarantees data release through the trustworthiness nodes but also keeps the energy balance through trustworthiness nodes. This model contains four modules such as trust estimator, trust database, decision-making path and bootstrap path. The idea of integration of fault tolerance and secure routing for WSNs was present in 2016 by (D. Devanagavi, et al., 2016). The goal of this model is establishing a secure path from the source to the base station, even in the presence of malicious nodes. In this idea, the agent-base trust model is used. Data is also transferred from a safety path and without malicious or compromiser nodes to the base station. Also in 2016, (Salehi, et al., 2016) proposed a trust-base compromising routing protocol for WSNs. The protocol is used for direct relations between the sensor nodes and it benefits from a novel watchdog mechanism considering not only the forwarding behavior of the nodes but also the quality of the links between them. In a sophisticated algorithm, choose the next node for net hop, according to three criteria of link quality, geographic location and trust level.

3. Calculation of trust in the STAR

3.1. Network topology model

In this network, each node has a unique identification that cannot be assigned to the other. However, the nodes in this network are immobile and all of them are homologous in terms of storage capacity, initial energy, power supply and computing power. Also, each node holds a list of last neighbor nodes trust value that previously interacted with them, in a table with the name of "Information List" (IL). The structure of this table includes the sliding window improved plan. Also, in order to remove the malicious node from the routing process, information of the mentioned node was recorded in the "Malicious Node List" (MNL) which each node has it.

For the storage of trust, the values will be considered in the range of (0-15), because if values are stored as the binary form or stored between 0 and 4, it leads to reducing the trust accuracy and if its data type be as decimal, it finished to increase the computing space and subsequently led to reduction in the computation speed. Therefore, this choice has the advantage that it reduces the memory and communication overhead through the computation of trust in this range of storing.

3.2. Information List and present sliding window improved structure.

Each node needs appropriate space to store the information of nodes that interact with them to access them when required, the accession must be able to do search, insert and delete operations. Thus, the IL structure is introduced as below:

IL structure contains two priority levels (Fig 1.*a*). High priority is for nodes that have more interaction with the source node and so their values remain longer in the sliding window. Therefore, a low priority is assigned to other nodes. The node with low priority changes to high priority after it was used N_{using} times in the routing. The implementation of this list has a major advantage that there is not needed to broadcast the trust value from base station and the nodes will not be required to share the trust value. Use this list, as well, increase efficiency of system resource by reducing in network communication overhead.

In each node, IL includes the number of cells and the structure of each cell includes node identification number, sliding window that its width is L_{sw} and residual chance. (Fig 1.b)

The proposed structure of the sliding window also includes three items. The first item is computing the trust time, the second item is the trust value and the third item is the interaction number. The sliding window is organized in IL (Fig 1.b).

3.3. Computation of performance based on network trust in interactions

At the beginning, the IL of the sliding window is empty for all nodes and, will be recorded the destination node information in the source node IL with any interaction. When node A wants to interact with node B, if there is no previous data of node B in IL of node A, the required data is provided by recommender nodes and if the data of node B exist, it is

placed in proportional new location according to the current time. The scenario can be explained as follows:

1. Starter node (node A), determines the destination node (node B) that is going to interact with it.
2. Check the presence or absence of node B is addressed in the IL of node A.

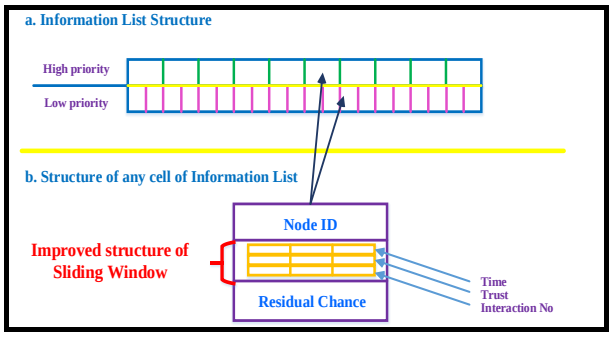


Fig 1. a. The IL structure, b. The structure of any cell of IL

There is the address of node B in IL of node A.

2.1.1. If the trust of node B, was unacceptable, node A passes up from interacting with node B and then terminates.

2.1.2. If the trust information retrieved from the list is acceptable, compare the time of the trust value recorded with the current time.

2.1.2.1. If this interval was acceptable, node A is going to interact with node B and inserts trust information of node B and the current time in the right location of IL and then terminates.

2.1.2.2. If this interval is not acceptable, go to the next step.

2.2. Address of node B does not exist in IL of node A or is not acceptable the interval then:

2.2.1. Node A sends a recommendation request package about declaration individual trust of each neighbor node to node B and asks their opinion about node B.

2.2.2. Any neighbor nodes send its direct or indirect opinion according to this scenario.

2.2.3. Node A, collects the received opinions. The opinions that have lower trust from threshold trust, are removed.

2.2.3.1. If the number of remover opinions is more than half plus one of total opinion, the final opinion of node A about node B, will be based on distrust to node B and terminate the process.

2.2.3.2. If the number of remover opinions is less than half plus one of the total opinion, node A arranges the gained opinions values base on trust value of its first next node. Then compute the weighted mean and then if the gained result is acceptable, the new trust of node B and its time, are stored in IL of node A. Therefore, node A will decide to interact with node B and choose the path that reports greatest trust value. If more than one path, declares the same trust value, the path of creating the interaction will be the path that has the minimum hop.

3.3.1. Methods of detection of malicious nodes

Based on the assumptions, there will be two kinds of malicious nodes: the node that its trust is reducing successively and the node that prevents from sending an information packet to the destination. Thus, two separate algorithms are introduced to detect malicious nodes:

3.3.1.1. Computation of Node Descending Trust (CNDT)

If there are $N_{use\text{-}chance}$ consecutive times, as much as 0.9% reduction in the latest trust (Tr_{new}) evaluation of a node than last trust value (Tr_{last}), the mentioned node has been identified as a suspicious node and is removed from IL and routing operations and their subsequent monitoring will be a duty of their neighbors. This is because it is believed that the trustworthiness of a node - that its trust value is reducing consecutively as much as 0.9 of last calculated trust value - is being questioned and needs supervision. To implement this approach, residual chance factor was used. In this way that by the initialization of residual chance for each node, as soon as the observation of condition (when trust value of Tr_{new} is reduced 0.9% than Tr_{last}), a unit of a residual chance factor is deducted and its value will be stored for future use. The Pseudo-code algorithm of CNDT can be seen in:

```
1  If N_rem-chance <> 0 then
2      If Tr_new >= Tr_last * 0.9 then
3          W_new= α
4          W_last = (1- α)
5      Else If Tr_new < Tr_last * 0.9 then
6          W_new= β
7          W_last = (1- β)
8          N_rem-chance = N_rem-chance – 1
9          Num_suspected++
10     Else
11         Alert ("The node is a ' Suspected Node ' and must be
           removed from the Info list and Routing process")
   Where:
       i.   α > (1- α)      AND
       ii.  (1- β) > β
```

Fig 2. CNDT algorithm pseudo-code

Here, the values of different W, are the weight of trust values or the impact factor of the history.

It should be pointed out that this algorithm, in addition, to identifying malicious nodes, provides the required weights for use in Equation 14, that will affect the history of the last trust value in the new value.

3.3.1.2. Computation of Node Association (CNA)

If the number of packets sent from a node such as A is greater or equal to $Percent_{packet\_sent}$ percentage of its neighbor nodes and the participation of a specific neighbor node like B exists in less than $Percent_{association}$ percentage of interactions of node A, then node B recognized as suspicious node, is used from chance and will be monitored. The Pseudo-code algorithm of CNA can be seen in Fig 3:

```
1  If (Num_packet_sent (A) >= (Num_neighbor (A) * Percent_packet_sent (A)) / 100)
2      If (Node_association (B) < (Num_packet_sent (A) * Percent_association (B)) / 100)
3          Using Chance (B)
4  Else
5          Alert ("The node is a ' Suspected Node ' and must be
           monitoring!")
   Where:
           Percent_packet_sent >> Percent_association
```

Fig 3. CNA algorithm pseudo-code

3.3.2. Monitoring phase

The scenario of monitoring phase is expressed as follows:

When a node detects that another node (such as X) has been suspicious, it informs its neighbors by sending a packet to them. Each node that was made aware of its malicious or suspected neighbor, after the time of tdet_mal from while node X detected as malicious or suspicious, then will be monitored the node X in a period of time, as follows:

1. If nodes A and C are as the neighbors of node X and are considered to be supervisor of X, then when node A wants to send data (including request Acknowledgement) to node C, then its data is sent via node X and waits for ACK message from node C (Fig 4-.a).
2. In this case, it is assumed that node X sends this packet to node C (Fig 4-.b). Also, node C which knows that node X is a malicious mode, sends an ACK message to senders of the packet which is received from node X (Fig 4-c) and then sends the received packet with node X information to the next node by the piggybacking method.
3. If after the t_{wait}, did not receive any ACK message from node C, that means that node X did not perform routing appropriately and sent the packet to its arbitrary node, such as node B (**Error! Reference source not found.**3.d). Therefore, node A will resend its data and information of node X also, by the piggybacking method.

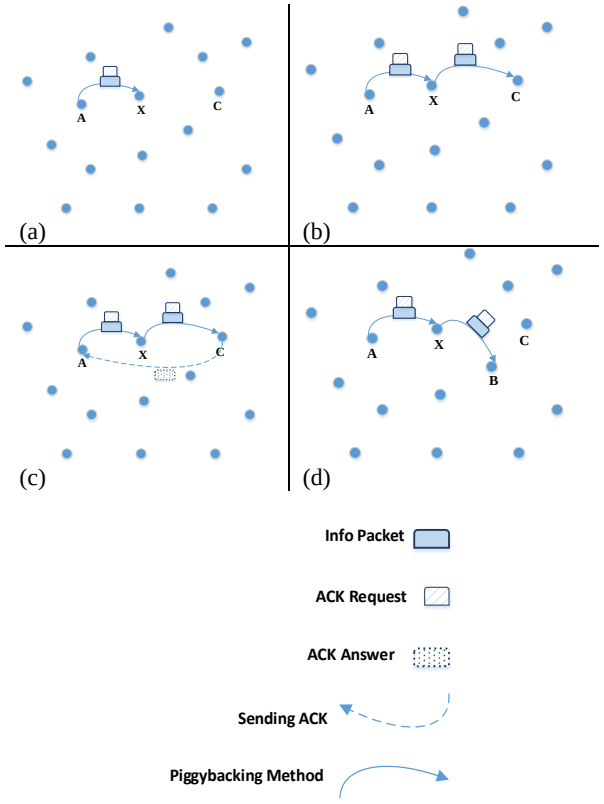


Fig 4. Steps of monitoring phase. a- Node A sends its packet via node X to node C. b- Node X sent the packet of node A to node C. c- Node C. When it received a packet of node A, it sends ACK message. d- Node X, doesn't send the packet to its destination.

Now we can consider a situation that malicious or suspect node X, wants to produce fake ACK and performing this action with the aim of deceiving the sender node (node A), certifies this node to arrive data to node C. Therefore, to ensure the arrival of the data packet to true destination, when a node A wants to send data to node X with the aim of monitoring it, node A informed the neighbors of node X. Neighbor node also, controls the arrival or non-arrival of the packet of node A to node C, through overhearing. Then node A after receiving the ACK, determines whether the ACK message is real or fake. Through voting results have been reported by neighbor nodes. The result of evaluation of of node X with gained information, including the amount of trust that has been evaluated and a number of times to monitoring, is stored in the base station to record the history. Each node that detects as malicious N_{det_mal} times, is removed from routing cycle and is not monitored never.

3.3.3. Distribution the information of malicious node between nodes and base station

The suggestion of distributing the information of malicious nodes will be based on this approach that each node in its interactions in several times, sends trust value of the node that was detected as a malicious node to base station via the piggybacking method. The base station also, in a certain interval, declares these values to neighbor nodes of the aforementioned node.

3.3.4. Method of interaction with new node and initialization of the trust value

The suggestion of how to deal with the node that has not recorded its interaction, would be considered a temporary trust based on its residual energy. This temporary value is the same initial value of trust for each node in starting the network. Then, the considered node will be tested on the verification test. If the result of the test is positive, interaction with this node will be started. Otherwise, the history of node evaluation will be stored as its trust value and it will be monitored by its neighbor nodes.

The verification test could be considered such that after the node A is sent its packet that includes of ACK request from node C, to node X (node X that is tested), after the appropriate time, the positive or negative result is determined through receiving or not receiving the ACK message from node C.

3.3.5. Calculation of energy parameters

The affecting factors in energy consumption, are the cost of sending and receiving the packet and passes up from the cost of energy for computation due to insignificance of the matter. To initialize the primary energy of any sensor node and energy consumption in receiving and sending the packet activities, the values that are proposed in (Zho, et al., 2015) have been used.

3.3.6. Decision and calculation of trust, based on subjective logic model

A node is computing the trust value of its neighbors by this method that the first search is the identification number (ID) of object node in its IL. If the search was successful, node A compares the last time of recorded trust value in its sliding window with the current time and if it was in interval

threshold, then its decisions are made based on trust value that is fetched and otherwise, node A computes the new trust value (Tr_{new}). It should be noted that the consideration of this case, has the important advantage that it will increase the speed of routing.

To calculate the amount of trust, the requester node use of these factors for evaluating the trust of the node considers: the amount of successful (N_S) and unsuccessful (N_{US}) interactions, the amount of residual energy (E_{rem}), the number of known as suspected node from evaluator node (N_{sus}), the number of monitored because of being known as malicious node from neighbor nodes (N_{mal}) and finally the interval of sending and receiving the packet by considering node (t_{sr}). To use any of these parameters first the amount of effect of any parameter must be investigated independently and ultimately, the resultant of all obtained relationships should be introduced as the final evaluated trust value. Subsequently, the trust value in the fields of energy, behavior and packet sending is computed.

The energy trust is calculated using Equation 4:

$$Tr_{energy} = E_{rem}/E_{ini} \tag{4}$$

Here, E_{ini} represents the value of initial energy of each node and Tr_{energy} illustrates the trust of the perspective of energy.

Due to Equation 5 of computing the trust in node behavior area, the effect of two parameters of N_{sus} and N_{mal} on trust value, fit into one equation:

$$Tr_{behaviour} = (1 - N_{sus}/N_{max-sus}) * (1 - N_{mal}/N_{max-mal}) \tag{5}$$

Where $N_{max\text{-}sus}$ is the maximum number of times that a node can monitor as a suspicious node and $N_{max\text{-}mal}$ is the maximum number of times that a node can monitor as a malicious node. $Tr_{behavior}$ illustrates the trust of the perspective of node behavior.

According to the Equation 5, it can be argued that if the outcome of $N_{sus}/N_{max-sus}$ or $N_{mal}/N_{max-mal}$ is zero, this means the complete trust in its area (The first fraction illustrates the suspicious and the second fraction expresses the malicious) and vice versa. If we have value one or close to one, this means subtracting the same amount of trust in the same field. The production of two parentheses also proves that if a node was detected as malicious node (that it's mean is $1 - N_{sus}/N_{max-sus} = 0$ or$1 - N_{mal}/N_{max-mal} = 0$), if one of the parenthesis being zero, the other will lose its cost; because it has been terminated its chance for malicious or suspicious and in each of these scenarios (malicious is known, or suspected), it has been established malicious of node.

To make-decision for the success or failure of sending a packet, the differences of the interval of sending and receiving of a packet must be calculated. But since it is inevitable that a packet might be missing in the networks, to consider the effects of this case, the calculation by (Faridi, et al., 2010) was used, first, the probability of packet loss in calculating the RTT was will be considered as follows:

When a node wants to send a packet and at the same time, another node or its neighbor nodes are also sending a packet, then the possibility of collision occurs. In this case, one or all of the packets are lost and the sender must resend its packet. What can occur in an information transmission, is the failure to access the channel, collision of packet or success in sending the packet.in (Faridi, et al., 2010), is considered as two modes for packet losing, one is the failure mode and another is the collision. So, based on the result of (Faridi, et al., 2010). The calculation of the probability that is the outcome of a packet is terminated to one of the modes of failure or collision obtained by Equations 6 and 7:

$$P_{FAIL} = \prod_{i=0}^{M}(1 - y_i) \tag{6}$$

$$P_{COL} = P_{co} \times (1 - P_{FAIL}) \tag{7}$$

In Equation 6, y is the possibility of access to the channel, M is number of channel detection states (CCA) andP_{FAIL}, is the possibility of failing in channel access and in Equation 7, P_{co}, is the probability of collision for each node and finally, P_{COL}, is the possibility of network collision.

Therefore, by applying the effect of the collision event, for obtaining a successful result or achieving an unsuccessful result in a particular interaction, Pseudo code of Fig 5 can be used:

```
1  If (RTT <= Tmax)
2      Status<- Successful
3  Else
4      Status<- Unsuccessful
```

Fig 5. Pseudocode of result of successful or unsuccessful in a sending

Here, RTT is the trip time (packet sending) and back (getting ACK) and T_{max} is the threshold value of RTT whose value is obtained for each node, by using Equation 8:

$$T_{max}(i) = Threshold_{RTT} + P_{Col}(i) \tag{8}$$

The concept of this equation is that for every node i can be added to the delay in receiving the packet, two values of RTT threshold and a measure of the probability of collision in packet sending for same node i.

By calculating each of subjective logic model parameters, the evaluated trust is computed based on SL model (Equations of 9, 10 and 11). S, F, b, d and u expressed success, failure, belief, disbelief and uncertainty, respectively:

$$b = (S/S + F + 1) \tag{9}$$

$$d = (F/S + F + 1) \tag{10}$$

$$u = (1/S + F + 1) \tag{11}$$

It is considerable that obtained value of u parameter is the confidence value that illustrated the amount of trust accuracy and for applying its effect on the trust value, the SL model is used.

By spotting the parameter "a" ($0 \leq a < 1$) as the base rate that is defined in the SL model, the expected value of trust in packet sending field, by Equation 12 is obtained:

$$Tr_{packet} = b + au \quad (12)$$

That the parameter "a" is usually considered as 0.5 and Tr_{packet} is expected value of trust in packet sending field.

Now, with having obtained trust values of each field (i.e. Equations of 4, 5 and 12), the final trust of a node is gained by adding and multiplying the weight of these trust values, according to Equation 13:

$$Tr_{new} = \alpha * Tr_{packet} + \beta * Tr_{energy} + \gamma * Tr_{behaviour} \quad (13)$$
$$Where \quad \alpha + \beta + \gamma = 1$$

In this equation, values of coefficient of α ، β and γ, are variable and show the amount of effect of each obtained trust. The value of Tr_{new} is the newest computed trust value of a certain node.

The method of entering and exiting trust value in sliding window and computing the impact factor in trust value can be calculated as defined in Equation 14:

$$w_i = \rho^{\varepsilon \Delta T_i} \quad (14)$$

And finally, the result of history (Tr_{last} is the result of trust value in three past interactions), is calculated according to Equation 15:

$$Tr_{last} = W_3 * Tr_3 + W_4 * Tr_4 + W_5 * Tr_5 \quad (15)$$

To save new trust in a sliding window by applying weighted (weights derived from algorithm Pseudo code CNDT), the new value of the last trust, Tr_{final}, is obtained from Equation 16:

$$Tr_{final} = W_{last} * Tr_{last} + W_{new} * Tr_{new} \quad (16)$$

4. Simulation results

The wireless sensor network of STAR has been 200 nodes that established in spatial of dimensions of 40 × 40 square meters. Each node has a different number of neighbors and has radio radius of eight meters. The parameters that using in this network and values of trust computation parameters is observed in **Error! Not a valid bookmark self-reference.**

Table 1. parameter of implementation

Row	Parameter name	Parameter value	explanation
1	IniSensorEnergy	10000	Amount of initial energy of each node
2	TransEnergy	0.144	Amount of energy consumption for packet sending
3	RecEnergy	0.0576	Amount of energy consumption for packet receiving
4	EnergyThreshold	6000	Acceptable threshold for residual energy
5	BaseRateSL	0.5	Base rate in SL model
6	α	0.25	Ratio of expected value of packet sending in computing the new trust value
7	β	0.55	Ratio of trust in energy field in computing the new trust value
8	δ	0.2	Ratio of trust in node behavior in computing the new trust value
9	θ	0.7	Weight of new trust value in state : $TR_{new} >= TR_{last} * 0.9$
10	ω	0.3	Weight of new trust value in state : $TR_{new} < TR_{last} * 0.9$
11	TrustThreshold	0.4	Acceptable threshold for trust value
12	MaliciousPercent	5 – 50	Total percentage of number of malicious node in network
13	PacketSentPercent	80	Percentage of number of sent packet by a certain node
14	AssociationPercent	3	Percentage of association of a node in sending the packet of neighbor nodes
15	RTTThreshold	5	Acceptable threshold for RTT
16	TotalChance	3	Total number of chances for each node
17	TotalSusPermission	3	Total number of allowed times for detecting a node as suspected
18	ε	0.01	Power parameter in Equation 14
19	ρ	0.9	Ratio of impact factor in Equation 14

4.1. Comparative evaluation of STAR with EDTM

In this section, the result of comparing the residual energy and detection percentage of the malicious node in STAR, with EDTM (Efficient Distributed Trust Model) will be analyzed that was proposed by (Jiang, et al., 2015).

4.1.1. Evaluation and analysis the residual energy of total of network

For achieving the residual energy of total of network, after each time determining the percentage of the number of malicious node and termination of simulation, the percentage of residual energy of total network is computed. The result of comparing the simulation of STAR and EDTM, presented in the graph of Fig. 6.

As the same time, STAR, is shown by having 11.99% increase compared to EDTM, it can better preserve the residual energy of network with a final value of the energy 867.9 MJ while, the residual energy of network in EDTM, is 760 MJ. This difference occurred due to storing the energy of sensor nodes by not forwarding the packet to malicious nodes that are placed in their neighborhood. Because after the simulation of malicious nodes detection algorithms, and again, because of the report of the detector node to base station, the neighbor nodes of the malicious node via base station declaration, are kept from interaction with malicious node without any energy, time and relation consumption. The providence in interaction creation, is useful in broadcasting a packet by a node to its neighbor nodes, because the malicious node exists in MNL of the sender node and then does not send any packet to it. Therefore, this led to more storing the energy of sender node. Also, using improved sliding window with impact factor, helps each node to use the latest trust values in sliding window instead of creating new interactions, the last trust value is used in sliding window structure and benefits from the previous actions in the detection and evaluation of that node, without any need to renewed consumption of energy.

4.1.2. Evaluation and analysis of the detection accuracy of malicious nodes

In the simulation, MaliciousPercent parameter grows from 5% to 50%, that in each increasing the malicious nodes number, the simulation is done and the amount of malicious nodes detection is evaluated. The result of this evaluation illustrates an increase in detection of the precision of STAR than EDTM model. The result is shown in graph of Fig. 7.

This comparison demonstrates that STAR increased the ability of malicious node detection to 94.02% by 1.52% increasing than EDTM design. However, the detection accuracy of EDTM model is 92.5%. But as you can see in the above figure, in STAR, the amount of detection accuracy from 30% damage to the next, has downfall by more coefficient than EDTM. This is because that in simulation and with the action of malicious nodes algorithms specially, and also, because of that some nodes it is placed near the more malicious nodes and is forced to use from residual neighbor nodes in their packet sending and some of these neighbor node after do not have enough energy for sending packet and then are detected as malicious nodes step-by-step, so if real malicious node is close to these nodes and that they are identifiable only through this nodes, then the suspicious nodes would remain hidden.

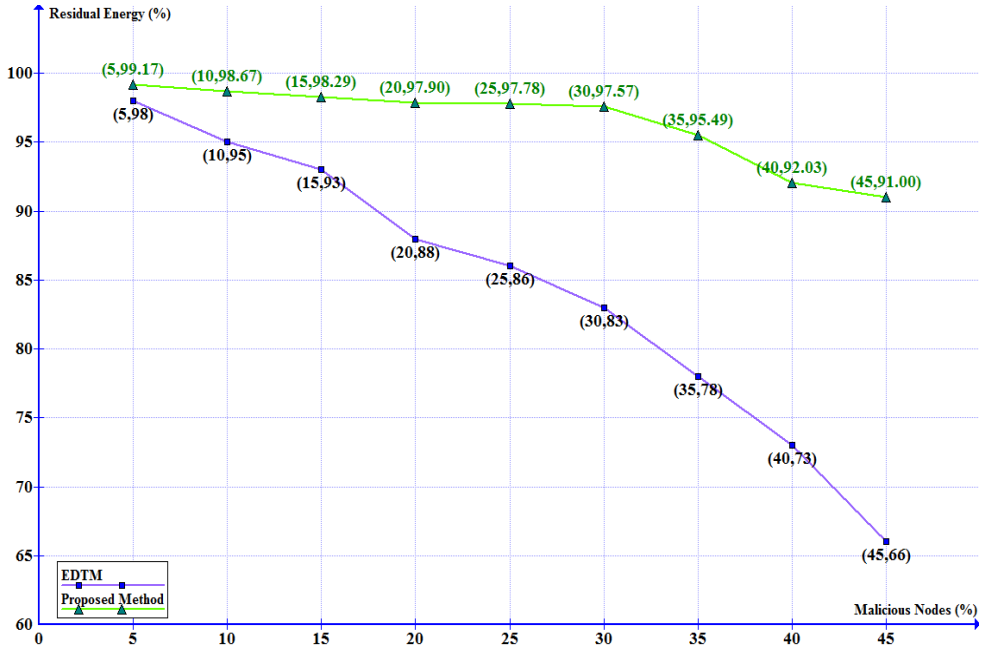


Fig. 6. comparison of residual energy

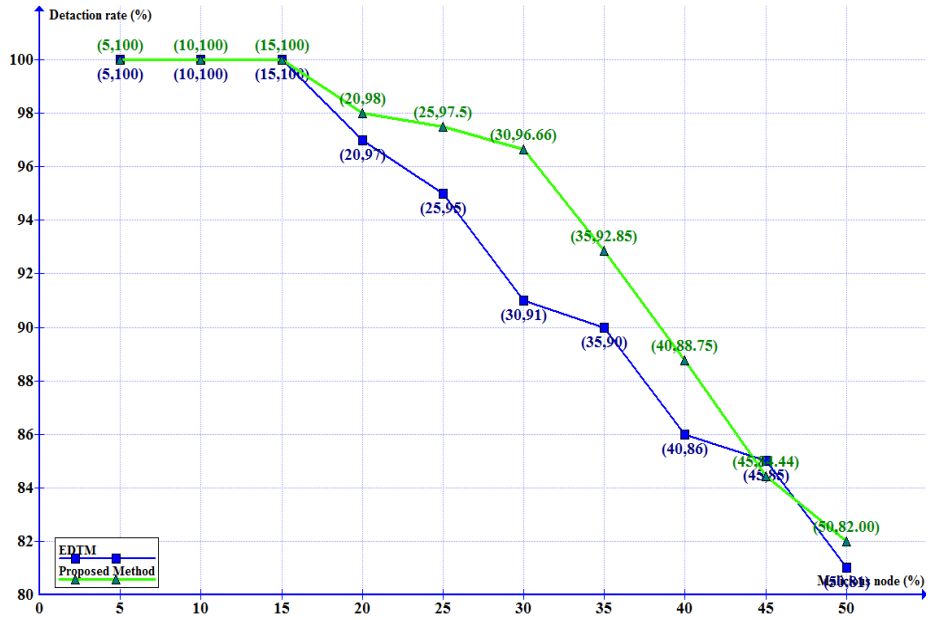


Fig. 7. Comparison of the detection accuracy of malicious nodes

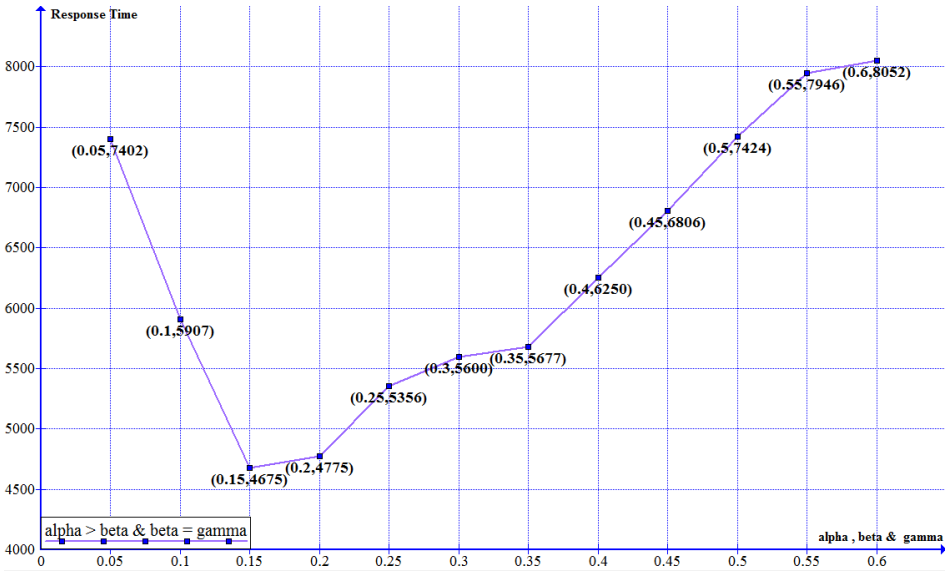


Fig. 8. Discussion of optimal value of α and response time

4.2. Analysis of the performance of STAR

In this section, the parameters of the simulation are discussed and based on the obtained curves of several runs of simulation, the optimal value is achieved.

4.2.1. Evaluation the effect of trust computation coefficients on network performance

In this evaluation, the purpose of trust computation coefficients is the coefficients used in Equation 13 that are recognized in simulation by the names of α, β and δ. So, for presentation of the amount of access speed of malicious nodes detection – that is one of goals of this study – for different values of the coefficients and the malicious percent equal to 5%, the graphs of the Figure 8 Figure 1 to are provided that express the amount of access speed of malicious nodes detection based on different values of coefficients of α, β and δ. In all of the graphs, the numerical value of obtained points response time in each change of α, β and δ values, is the yield of result mean of simulation run in three times. Other parameters are initialized according to the Tabel 1..

Remind that α is the TR_{packet} coefficient in Equation 13. In the graph, the below situations for α, β and δ are considered.

$alpha > beta \ \& \ gamma$

$beta = gamma$

Since that sum of three coefficients is equal to one, each point in the graph of Fig 8. The α, β and δ values of Table 2 follow.

The graph of Fig 8. and Table 2 shows that optimal value for α coefficient is placed in range of (0.15, 0.25). The

optimal point is explanatory that the increment in expected value of packet sending, from out of this range, led to an increment in response time and then this event is accented that two coefficients of β and δ are also important,.

The same process is running for β and δ coefficients in graphs of Fig. 9 Fig. 9and a similar value of Table 2.
The state of coefficients in Fig. 9. is as follow:
$beta > alpha \,\&\, gamma$
$alpha = gamma$
and in
Fig. 10, is as follow:
$gamma > beta \,\&\, alpha$
$alpha = beta$

It is pointed out that the β in Equation 13 the TR_{energy} coefficient exists.
As the same shown in Fig. 9 Fig. 9the optimal value of β is in the range of (0.45, 0.55). This graph shows that although the response time is decreasing by increasing the energy trust coefficient value to 0.5, but after this amount and further increasing the energy trust coefficient, the response time will be increased. That means that although residual energy amount is the more effective factor in response time, but at the same time, the amount of trust in packet sending and in node behavior in the interaction with neighbors, has its appropriate value, too. The graph of

Fig. 10, shows the discussion of δ that is $TR_{behavior}$ coefficient. The optimized value of δ is in the range of (0.15, 0.25).

Table 2. α, β and δ values in discussion the optimal value of α

Point Coordinates	α	β	δ	Response Time
0.05 , 7402	0.05	0.475	0.475	7402
0.1 , 5907	0.1	0.45	0.45	5907
0.15 , 4675	0.15	0.425	0.425	4675
0.2 , 4775	0.2	0.4	0.4	4775
0.25 , 5376	0.25	0.375	0.375	5356
0.3 , 5600	0.3	0.35	0.35	5600
0.35 , 5677	0.35	0.325	0.325	5677
0.4 , 6250	0.4	0.3	0.3	6250
0.45 , 6806	0.45	0.275	0.275	6806
0.5 , 7424	0.5	0.25	0.25	7424
0.55 , 7946	0.55	0.225	0.225	7946
0.6 , 8052	0.6	0.15	0.15	8052

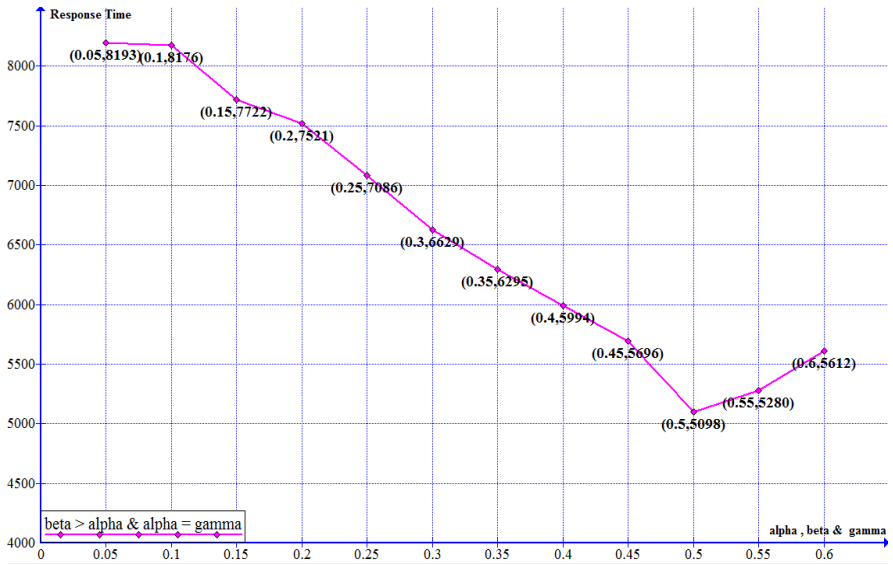


Fig. 9. Discussion of optimal value of β and response time

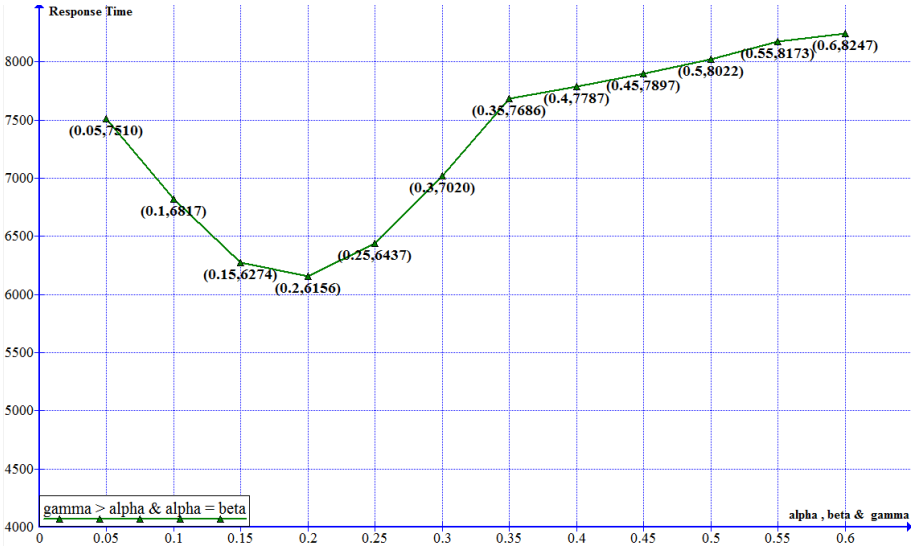


Fig. 10. Discussion of optimal value of δ and response time

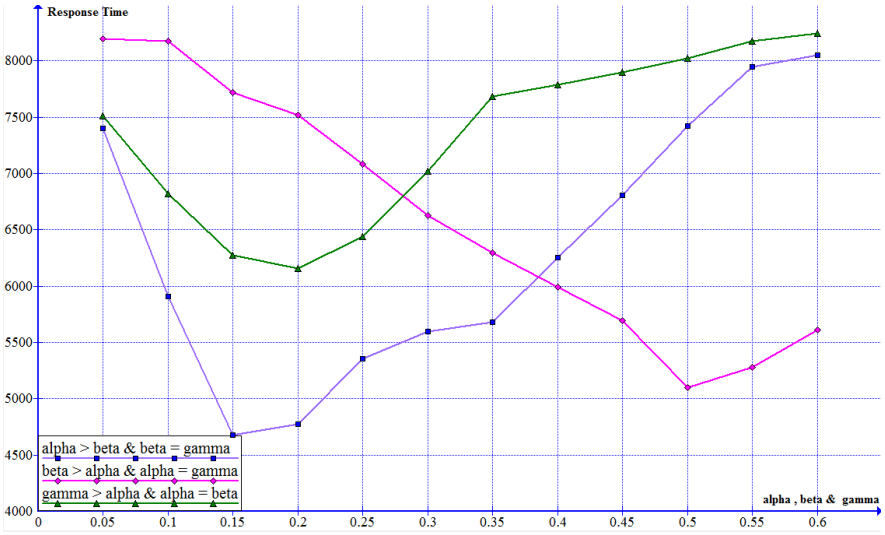


Fig 11. Comparison of response time in α, β and δ

For comparison and designing how to select the coefficients values of α, β and δ, all three graphs are plotted in Fig. 11.

Based on the chart, we can say according to the proof of the significance and the effect of energy in decreasing the response time, the criterion of choosing α, β and δ values is based on priorities of energy, expected value of packet sending and node behavior. The obtained value for β coefficient (0.55), reflects the fact that the residual energy of a sensor node in the trustworthiness of the same node, has more importance than the node behavior between neighbor nodes. On the other hand, it seems reasonable that if a subject node is going to choose a node for its interactions, between two sensor nodes that the first node has high trustworthiness in packet sending (TR_{packet}) or in behavior ($TR_{behavior}$) and has minimum of energy, and against the trustworthiness of packet sending or behavior of second node was lower than the first node but its residual energy was greater, the starter node will choose the second node, because the probability of reaching the packet. By the node that has more energy than a node with lower energy, is much higher. Also, according to this graph and with comparing the minimum point on all three graphs, about the δ curve that has upper minimum point than the α and β curves, we can say that the impact of node behavior in STAR, than the impact of energy and the expected value of packet sending, is lower. In other words, the high important of energy and expected value of packet sending in routing process is reflected.

4.2.2. Evaluation of the accuracy of STAR

In this section, the accuracy of STAR is evaluated based on Precision, Recall and F_1-measure factors.

The Precision factor (Equation 17) shows the accuracy

rate among the predicted data.

$$Precision = TP/(TP + FP) \quad (17)$$

The Recall factor (Equation 18) shows the ratio of predicted data to a total number of expected data for prediction.

$$Recall = TP/(TP + FN) \quad (18)$$

Also, F_1-measure (Equation 19), is the weighted mean between Precision and Recall.

$$F_1 = 2 * ((Precision * Recall)/(Precision + Recall)) \quad (19)$$

Therefore, it can be plotted in

Table 3 that shows Precision, Recall and F_1-measure values in malicious nodes detection for different percentages of malicious nodes (Malicious Percent).

Table 3. Accuracy of STAR between predicted data

MaliciousPercent	Precision	Recall	F_1-measure
55	1	1	1
10%	1	1	1
15%	0.88	1	0.93
20%	0.84	0.97	0.9
25%	0.81	0.98	0.88
30%	0.8	0.96	0.87
35%	0.78	0.92	0.84
40%	0.78	0.92	0.84
45%	0.78	0.86	0.81
50%	0.74	0.82	0.77

According to the Precision values in

Table 3 that illustrate the obtained accuracy, it is observed that although with the increment in the malicious rate of nodes, the detection accuracy has a descending flow, but achieving the values more than 0.7 show the detection with appropriate accuracy. Also, Recall values are explanatory that even by an increment in malicious nodes percent, the STAR has a very good ability for detection of the really malicious nodes. Similarly, the F_1-measure factor emphasizes on verification the resulting values of two criteria of Precision and Recall.

5. Conclusion and future works

The result of this research has shown that by using the STAR it can be performed up to very substantial saving in energy consumption and at the same time, adding to previous achievements, the accuracy of detection of malicious nodes is increased. Increased accuracy in malicious nodes detecting is a factor for the speed growth of routing. This is because if detection accuracy is higher, the participating malicious or suspicious nodes in selecting and applying in the path of packet sending is decreased and then the packet will arrive earlier and more securely.

In the future we can regulate the chance factor for each node based on function of neighbor opinions until the number of chance for a node that has more trust value with view of its neighbors can be much more than a node that does not have this advantage, because the factor is an important agent in the postponement of detection a node as malicious node. So the node that cannot have trust value from views of its neighbors, has less chance of having the opportunity and will be introduced earlier as a malicious node. In addition to the above, in order to reduce the number of nodes that are known to malicious node wrongly, the base station can evaluate the received information and if it confirms the existence of malicious behavior, votes not to trust reported node and then introduces this node to the network as malicious. We can also present the strategies for resisting the STAR against the attacks, especially on-off attacks and bad mouthing attack in future.

References

[1] Ahmed, A., Abu Bakr, K. & Channa, M. I., "A Secure Routing Protocol with Trust and Energy Awareness for Wireless Sensor Network", *Mobile Networks and Applications*, 21(2), pp. 272-285, (2016).

[2] Chakrabarti, A., Parekh, V. & Ruia, A., "A Trust Based Routing Scheme for Wireless Sensor Networks", Bangalore, India, Springer Berlin Heidelberg, pp. 159-169, (2012).

[3] Chavan, P. A., Aher, R. D., Khairnar, K. V. & Sonawane, H. D., "Enhanced Trust Aware Routing Framework against Sinkhole Attacks in Wireless Sensor Networks", *Engineering and Technical Research (IJETR)*, 3(1), pp. 22-25, (2015).

[4] D. Devanagavi, G., Nalini, N. & C.Biradar, R., "Secured routing in wireless sensor networks using fault-free and trusted nodes", *Communacation Systems*, 29(1), pp. 170-193, (2016).

[5] Faridi, A., Dohler, M. & Grieco, L. A., Comprehensive Evaluation of the IEEE 802.15.4 MAC Layer Performance With Retransmissions. IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, 59(8), pp. 3917-3932, (2010).

[6] Gambetta, D., "Can we trust trust", *Trust: Making and breaking cooperative relations.* pp. 213-237, (1988).

[7] He, S. & Zhao, H., "Trust and potential field-based routing protocol for wireless sensor networks", *Ningbo, Signal Processing, Communications and Computing (ICSPCC)*, (2015).

[8] Hoceini, O., Talbi, S. & Aoudjit, R., "Trust framework for a secured routing in wireless sensor network", *Electronic Journal of Information Technology*, 8(8), pp. 58-65, (2015).

[9] Ishmanov, F., Won Kim, S. & Yeob Nam, S., A "Robust Trust Establishment Scheme for Wireless Sensor Networks", *Sensors*, Volume 15, pp. 7040-7061, (2015).

[10] Jiang, J. et al., "An Efficient Distributed Trust Model for Wireless Sensor Networks", *PARALLEL AND DISTRIBUTED SYSTEMS*, 26(5), pp. 1228 - 1237, (2015).

[11] Josang, A., Gray, L. & Kinateder, M., "Simplification and Analysis of Transitive Trust Networks", *Web Intelligence and Agent Systems Journal*, 4(2), pp. 139 - 161, (2006).

[12] Kaur, J., Sandeep, S. G. & Balwinder, S. D., "Secure Trust Based Key Management Routing Framework for Wireless Sensor Networks", *Journal of Engineering*, pp. 1-9, (2016).

[13] Latha, D. & Palanivel, K., "Secure Routing Through Trusted Nodes in Wirels Sensor Networks – A Survey", *Advanced Research in Computer Engineering & Technology (IJARCET)*, 3(11), pp. 3792-3799, (2014).

[14] Lewis, N., Foukia, N. & G. Govan, D., "Using trust for key distribution and route selection in Wireless Sensor Networks", *Salvador*, Bahia, s.n, (2008).

[15] McKnight, D. H. & Chervany, N. L., "The Meanings of Trust", s.l.: University of Minnesota, *Management Information Systems Reseach Center*, (1996).

[16] Olmedilla, D., Rana, O. F., Matthews, B. & Nejdl, W., "Security and Trust Issues in Semantic Grids", Dagstuhl, Germany, *Internationales Begegnungs- und Forschungszentrum Informatik (IBFI)*, (2006).

[17] Raza, S., Haider, W. & Durrani, N. M., "Trust Based Energy Preserving Routing Protocol in Multi-hop WSN. In: Networked Systems", *s.l.:Springer International Publishing*, pp. 518-523, (2015).

[18] Salehi, M., Boukerche, A. & Darehshoorzadeh, A., "Towards a novel trust-based opportunistic routing protocol for wireless networks", *Wireless Netw*, 22(3), p. 927–943, (2016).

[19] Vasudha, N. & Gajkumar Shah, P., "Requisite Trust Based Routing Protocol for WSN", *International Journal* of *Innovative Research in Advanced Engineering (IJIRAE)*, 01(05), pp. 138-142, (2014).

[20] Zahariadis, T. et al., "Implementing a Trust-Aware Routing Protocol in Wireless Sensor Nodes", *London*, s.n, (2010).

[21] Zho, C. et al., "Toward Offering More Useful Data Reliably to Mobile Cloud From Wireless Sensor Network", *IEEE Transactions on Emerging Topics in Computing*, 3(1), pp. 84-94, (2015).

Journal of Computer and Knowledge Engineering, VOL. 1, NO.2, 2017.
DOI: 10.22067/cke.v1i2.69769

Modeling Intra-label Dynamics and Analyzing the Role of Blank in Connectionist Temporal Classification

Ashkan Sadeghi Lotfabadi

Kamaledin Ghiasi-Shirazi Ahad Harati*

Abstract: The goal of many tasks in the realm of sequence processing is to map a sequence of input data to a sequence of output labels. Long short-term memory (LSTM), a type of recurrent neural network (RNN), equipped with connectionist temporal classification (CTC) has been proved to be one of the most suitable tools for such tasks. With the aid of CTC, the existence of per-frame labeled sequences are no longer necessary and it suffices to only knowing the sequence of labels. However, in CTC, only a single state is assigned to each label and consequently, LSTM would not learn the intra-label relationships. In this paper, we propose to remedy this weakness by increasing the number of states assigned to each label and actively modeling such intra-label transitions. On the other hand, the output of a CTC network usually corresponds to the set of all possible labels along with a blank. One of the uses of blank is in the recognition of multiple consecutive identical labels. Assigning more than one state to each label, we can also decode consecutive identical labels without resorting to the blank. We investigated the effect of increasing the number of sub-labels with/without blank on the recognition rate of the system. We performed experiments on two printed and handwritten Arabic datasets. Our experiments showed that while on simple tasks a model without blank may converge faster, on real-world complex datasets use of blank significantly improves the results.

Keywords: Connectionist Temporal Classification; Handwriting Recognition; Recurrent Neural Networks; Multidimensional Long Short Term Memory; Blank.

1. Introduction

Labeling unsegmented sequences is one of the most significant and common problems in the field of artificial intelligence. Handwriting, speech and gesture recognition are examples of this problem. Some solutions to this problem have been given by probabilistic graphical models like HMM (Hidden Markov Model). However, HMMs are generative models whereas labeling a sequence is a dicriminative task.

On the other hand, RNNs (Recurrent Neural Networks) can be trained discriminatively and have a strong structure which learns the data and time dependencies. However, RNNs need pre-segmented data for training. A traditional solution has been to combine HMMs with RNNs. As we mentioned above, HMMs are generative models and they are not the best choice for a discriminative task like sequence labeling. Another solution is CTC which is a newer framework than HMMs. We can consider the output of RNN as a probability distribution on all the possible label sequences and then we get an objective function to maximize the true labeling probability [1].

One of the main issues confronted by RNNs is the vanishing/exploding gradient problem[2,3]. LSTM (Long Short Term Memory) [4,5] was designed to solve this problem. Moreover, BLSTM (Bidirectional LSTM) [6,7] and MDLSTM (Multidimensional LSTM) [8] are two other generalizations of LSTM and are proposed to learn bidirectional and multidimensional contexts, respectively. Since LSTM is a kind of RNN, it is possible to combine CTC with LSTM [1, 8-15].

Woellmer [10,16] proposed to combine DBNs (Dynamic Bayesian Network) and CTC to learn more complex relations like finding keywords in speech or text. In addition, there are some generalizations for CTC like ECTC (Extended CTC) [15] which consider a consistency to evaluate frame-to-frame visual similarities in CTC. Another generalization is HCTC (Hierarchical CTC) [17] which is composed of several layers of CTC in which each layer has a special task to learn a specific context of a sequence. In addition to RNNs, CTC can be combined with graphical models like LDCRF (Latent-Dynamic Conditional Random Field) [18].

One problem with CTC is that it leaves the task of learning the dynamics within each label to the underneath RNN. In this paper (This paper is the extension of our previous paper [19] A. S. Lotfabadi, K. Ghiasi-Shirazi, and A. Harati, "Modeling intra-label dynamics in connectionist temporal classification," in 2017 7th International Conference on Computer and Knowledge Engineering (ICCKE), 2017, pp. 367-371.), we propose to model each label as a sequence of hidden internal labels and show how these internal labels can be learned. We postulate that by splitting each label to several hidden sub-labels, the task of the underneath RNN will become simpler and the overall accuracy increases. The output of the underneath network consists of all possible labels plus blank. The blank plays two roles in the network. Firstly, it allows recognition of consecutive identical labels and secondly, it frees the network from predicting the label of a sub-sequence until it has gathered enough evidence. Nevertheless, we will show that by considering several states for each label, the network can recognize contingent identical labels without using blank.

The organization of the paper is as follows: Section 0 introduces CTC, its mathematical formulation and its training algorithm. In Section 0, we introduce the Multi-state CTC

Manuscript received: October 25, 2017. Accepted December 22, 2017.
A. Sadeghi Lotfabadi, Department of Computer Engineering Ferdowsi University of Mashhad, Iran, a.sadeghilotfabadi@mail.um.ac.ir.
K. Ghiasi-Shirazi, epartment of Computer Engineering Ferdowsi University of Mashhad, Iran, k.ghiasi@um.ac.ir.
*A. Harati, Department of Computer Engineering Ferdowsi University of Mashhad, Iran. k.ghiasi@um.ac.ir.

(M-CTC) in which we propose splitting each label into multiple states/sub-labels. In section IV, we investigate the role of blank in standard CTC and the proposed M-CTC. In Section V we report our experiments on M-CTC with/without blank. We conclude the paper in Section VI.

2. Connectionist Temporal Classification (CTC)

CTC was proposed in 2006 by A. Graves [1] to train RNNs on unsegmented sequences. Prior to CTC, training RNNs on sequential data required the label to be specified for every frame of the input sequence. CTC revolutionized this process by making training possible when only a label sequence was given for the whole input sequence, without knowing the alignment between input and label sequences [20].

Assume that the number of outputs in the underneath RNN for each frame is equal to the number of labels plus one (for blank, or no label). A softmax layer normalizes these outputs before the last hidden layer sends them to CTC:

$$y_k^t = \frac{e^{a_k^t}}{\sum_{k'} e^{a_{k'}^t}} \tag{1}$$

where a_k^t is the kth output of the RNN in frame t and y_k^t is the normalized output. To obtain the probability of path (π), we use equation (2):

$$p(\pi \mid \boldsymbol{x}) = \prod_{t=1}^{T} p(\pi_t \mid \boldsymbol{x}) = \prod_{t=1}^{T} y_{\pi_t}^t \tag{2}$$

Paths are mapped to a label sequence by function F. This function removes the same consecutive labels and blank. For example, $F(a-ab-) = F(-aa--abb) = aab$ in which- means blank. Therefore, the probability of a special labeling like l is the sum of the probabilities of all paths which are mapped by F to l as shown in (3):

$$p(\boldsymbol{l} \mid \boldsymbol{x}) = \sum_{\pi \in F^{-1}(\boldsymbol{l})} p(\pi \mid \boldsymbol{x}) \tag{3}$$

The number of related paths to a specific labeling grows exponentially with respect to the length of the input sequence. However, it can be solved by dynamic programming and the algorithm is similar to the forward-backward in HMM [1]. Fig. 1 illustrates the CTC structure for all paths which map to the labeling ‘SUN’.

Loss function L(s) in CTC is the negative log probability of correctly labeling of all the training examples.

$$L(S) = -\ln \prod_{(\boldsymbol{x},\boldsymbol{z}) \in S} p(\boldsymbol{z} \mid \boldsymbol{x}) = -\sum_{(\boldsymbol{x},\boldsymbol{z}) \in S} \ln p(\boldsymbol{z} \mid \boldsymbol{x}) \tag{4}$$

The derivation of the loss function with respect to the networks parameters is done by using the backpropagation through time algorithm. So, it is possible to train the network by any gradient-based nonlinear optimization algorithm [20].

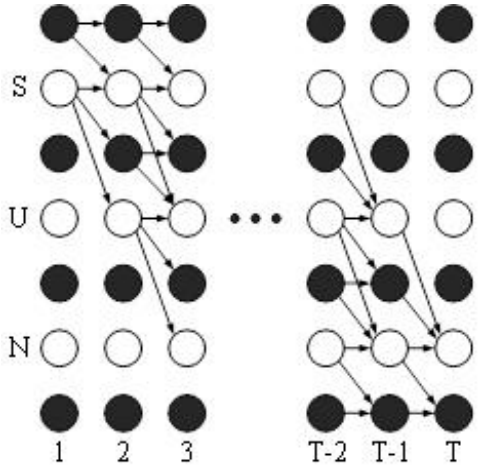


Fig. 1. CTC structure for ‘SUN’. Black and white units show *blank* and labels, respectively. The allowed paths which lead to ‘SUN’ labeling are determined by arrows.

3. Multi-state CTC (M-CTC)

Learning temporal context plays an important role in sequence labeling. So, the network should be able to maximally learn relations between frames. As we mentioned earlier, RNNs have the vanishing/explosion gradient problem. Therefore, we use LSTM which solves this problem. In most sequences the relations between frames are bidirectional. It means that future information is as important as past information. Thus, it is better to use networks which use both past and future information. Accordingly, we choose MDLSTM [8] which is a generalization of LSTM since MDLSTM can learn long-range dependencies in all spatio-temporal dimensions.

As we noted before, CTC was proposed to help RNNs (which is MDLSTM in our case) in training with unsegmented data. Considering Fig. 1, it is supposed that the MDLSTM determines the probability of each label at time t and the labels are given in the exact order from the left side which in this example are English alphabets. CTC considers 1 state for each label. So, the error is calculated by the CTC for each label at time t. Therefore, MDLSTM just learns the extrinsic dynamics between the labels. However, if we consider n state for each label in the CTC, the network not only learns extrinsic dynamics between the labels, but also learns intrinsic dynamics of labels and the relations of internal components of each label (which from now on we refer to as sub-labels). This increase in the number of states leads to better and more detailed learning of MDLSTM. An example of considering 2 states for each label class is illustrated in Fig. 2.

However, an equal number of sub-labels for each label may not be the best choice. It is better to determine the number of sub-labels according to the data length of that label the class. It means that a long data set needs more sub-labels than a short data set. Fig. 3 provides more details about this idea which illustrates 3 handwriting words from the IFN/ENIT [21] dataset. In this example, the number of sub-labels is determined based on average label length. By the above explanation, the number of sub-labels for letters ‘ث’, ‘ر’, ‘ا’ and ‘م’ are 5, 4, 2 and 5, respectively.

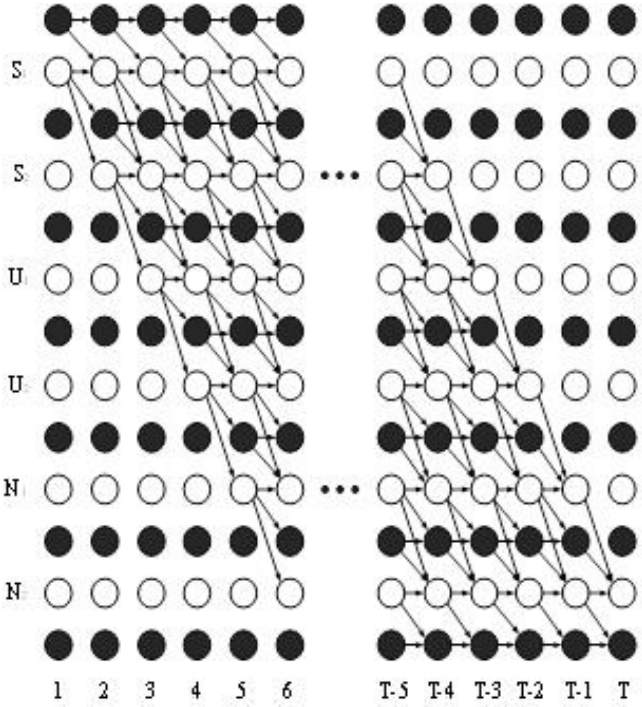


Fig. 2. CTC structure after considering 2 state for each label class

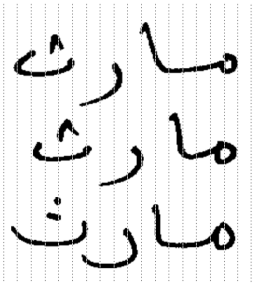


Fig. 3. Determining number of sub-labels according to label length. The gaps between vertical lines correspond to the frames.

4. Role of Blank

The output of the network beneath CTC consists of all possible labels plus the blank. Blank plays two roles in the network. Firstly, it prevents deletion of identical consecutive labels. For example, without blank CTC would recognize the word 'accuracy' as 'acuracy', recognizing the two consecutive letters 'c' as one. Secondly, blank frees the network from the oblige of predicting a label at each frame. For example, in the task of speech recognition, silences and short pauses between the utterance of phonemes can be recognized as blank (see page 64 of [20]). In the following, we will investigate the role of the blank when one models each label by multiple states in CTC.

According to Eq. (3), the probability of a labeling for the whole sequence is obtained by summing up the probabilities of all paths leading to that labeling. A path is an assignment of labels (possibly blank) to each frame. We define the function F from paths to sequences of labels as a mapping that removes blanks and repetitive labels. For example, paths 'oo-f-fff', '--off-f---', and '---o-ff--fffff-' are mapped to the sequence label 'off'. Please note that in these examples, frames which are labeled 'f' form two groups which are separated by one or multiple blanks. Having only a single group of labels 'f', the path would have been mapped to the label sequence 'of'. In fact, one of the reasons for provisioning the blank in CTC was to avoid misrecognition of words with consecutive repetitive words. Fig. 4 shows the structure of a CTC for recognizing the word 'off'. As it can be seen, there is no link between the two instances of label 'f' and all paths should pass through the intermediate blank label.

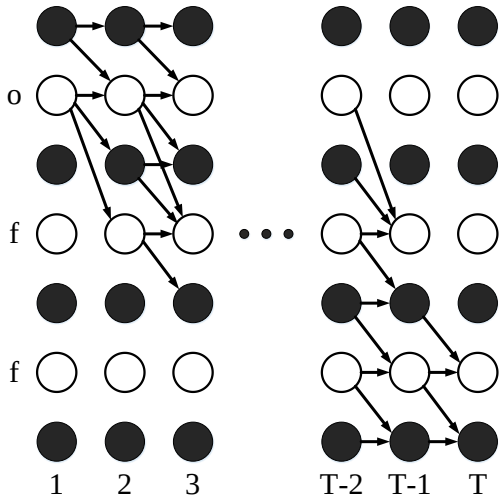


Fig. 4. CTC structure for 'off'. There is no link between the two instances of label 'f' and all paths should pass through the intermediate blank label.

In Fig. 4, which exemplifies the structure of standard CTC, each label is modeled by exactly one state (shown as nodes in the Figure). In this paper, we propose to model each label with several states/sub-labels. Having multiple states for each label automatically guarantees that consecutive labels would be correctly recognized, eliminating one of the reasons behind considering the special label blank. Assume that we have modeled each label with two states and have eliminated blank from CTC, arriving at the CTC structure is shown in Fig. 5. Now, to map a sequence of frames to a label, it is necessary to see both of its sub-labels in order, which guarantees correct recognition of consecutive repetitive labels. For example, paths '$o_1o_1o_2f_1f_2f_2f_2f_1f_2$' and '$o_1o_1o_2f_1f_1f_2f_1f_2f_2$' both yield the sequence label '$o_1o_2f_1f_2f_1f_2$'. Modeling each label with multiple sub-labels eliminated the first reason for the existence of the blank. But, what about the second reason which was to free the network from predicting a (non-blank) label at each frame? In the following, we perform experiments with/without blank to answer this question.

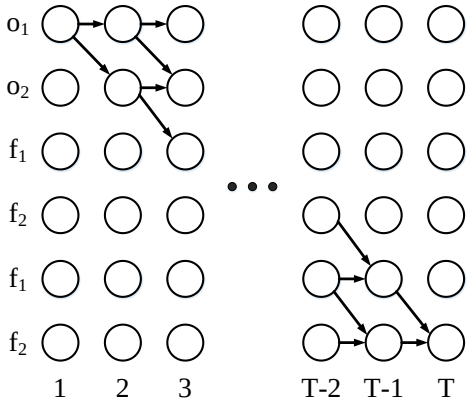


Fig. 5. CTC structure for 'off' by considering 2 state for each label and removing blank.

5. Experiments and Results

In this section, we explain the experiments used to evaluate the performance of the proposed method. The goal of these experiments is to investigate the role of the number of sub-labels and inclusion/exclusion of blank in the recognition accuracy. To evaluate the standard CTC, we use RNNLIB [22] which has been developed by inventors of CTC themselves. The neural network at the bottom of CTC consists of three hidden layers each one having four parallel LSTM layers. Each LSTM layer consists of several LSTM cells, the number of which is identified by "Hidden Size" in the tables of our experiments. The input data of each hidden layer is obtained by moving a window on the output neurons of the preceding layer in four directions. The sizes of these windows are designated by "Input Block" and "Hidden Block" in input and hidden layers, respectively. We also use some subsampling hidden layers whose sizes are identified by "Subsampling Size". In all cases, we use the gradient descent training algorithm.

5-1. Printed Digits Dataset

To evaluate the proposed method in the task of discovering the true sub-labels of each label, we generated an artificial dataset in which each label actually encompasses two sub-labels. In this dataset, we have 10 different labels corresponding to digit sequences '01', '03', '11', '12', '21', '31', '33', '41', '43', and '52', each label being made from two digits. The dataset contains 1000 samples of sequences of 3 labels (6 digits). Fig. 6 shows a sample image from this dataset. We use 800 samples for training, 100 for validation, and the remaining 100 for testing.

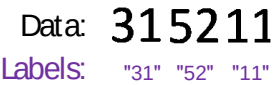


Fig. 6. An example of printed digits dataset with true labels.

We performed 7 different experiments on this dataset. The parameters of four of these experiments which were performed with blank are shown in Table 1 and are titled from A to D. There are three other experiments which are identical to the experiments B, C, and D, except that now the blank is removed. The reason that experiment A has no counterpart with blank removed is that in experiment A the number of sub-labels is equal to one and, according to the explanations of Section 4, the blank is essential to the recognition of consecutive repetitive labels. The number of states for each label in the experiments B, C, and D was chosen as 2, 4, and 10 respectively. The other fact important to note is the difference between the values of "Input Block" and "Hidden Block" in experiment D. The reason behind this difference is that since the number of sub-labels is very high, the length of the input data to CTC should be long enough to visit all the sub-labels. Therefore, the length of the input data to CTC in experiment D should be slightly more than that of the other three experiments. In fact, we believe that one of the reasons that in our experiments M-CTC has been shown to be superior to CTC, in contrast to the experiments reported in [23], is the adjustment of these parameters.

Table 1. Parameters For Printed Digits Experiments

Experiment	A	B	C	D
Hidden Size	2, 10, 50	2, 10, 50	2, 10, 50	2, 10, 50
Subsample Size	6, 20	6, 20	6, 20	6, 20
Hidden Block	3x4, 2x4	3x4, 2x4	3x4, 2x4	2x4, 2x4
Input Block	3x4	3x4	3x4	2x4
Learn Rate	1e-4	1e-4	1e-4	1e-4
momentum	0.9	0.9	0.9	0.9
optimizer	Steepest ascent	Steepest ascent	Steepest ascent	Steepest ascent
Number of Sub-labels	1	2	4	10

Because of the simplicity of this dataset, in all experiments, the accuracy of 100% was obtained. For this reason, we compare method based on their speed at reaching a solution. Explicitly, we compare the error of different methods at the end of a certain epoch. Table 2 illustrates the results of our experiments. This Table has been filled based on the error obtained at epoch 25. As stated previously, experiment A has no "without-blank" counterpart. The experiments show that increasing the number of sub-labels has decreased the error, obtaining the best results in the experiment D. The other important observation is the effect of removing blank. The results show that the accuracy of experiments without blank is superior to those with a blank. This shows that in this simple example (in which data for each label are artificially generated by concatenating data of two sub-labels), not only the blank label does not improve the results, but it also has increased the time of getting error zero. An example of the output of a model with 4 sub-labels and without blank is shown in Fig. 7. In this Figure, two consecutive instances of label '52' are recognized without any problem.

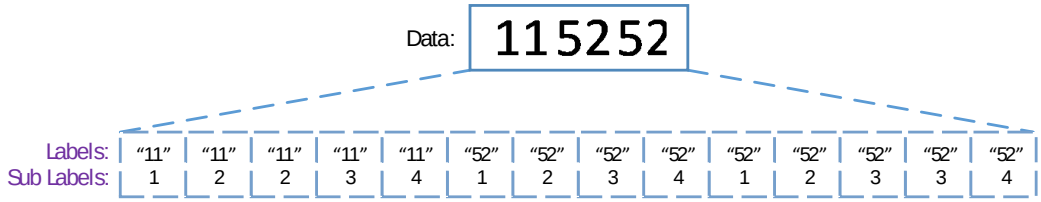


Fig. 7. MDLSTM output for a data from printed digits. In this model blank is removed and each label has 4 sub-labels. Each square shows a frame and includes the labels and sub-labels.

Table 2. Results For Printed Digits

Experiment	With blank						Without blank					
	CTC Error			Label Error %			CTC Error			Label Error %		
	Train	Val	Test	Train	Val	Test	Train	Val	Test	Train	Val	Test
A	7.52	7.60	7.67	87.63	88.94	87.90	-	-	-	-	-	-
B	12.15	11.75	11.79	74.57	73.72	72.65	0.29	0.30	0.31	1.79	2.12	1.68
C	1.00	0.84	0.82	0.28	0.22	0.13	0.33	0.36	0.35	0.06	0.05	0.05
D	0.19	0.19	0.19	0.00	0.00	0.00	0.21	0.21	0.20	0.00	0.00	0.00

5-2. IFN/ENIT Dataset

This dataset consists of 32249 samples from the handwritten images of 937 towns in Tunisia. Each sample in this dataset has a label sequence, each label being chosen from 120 possible choices of letters of alphabet, digits, and punctuation signs. This dataset has five segments named 'a' to 'e'. We perform our experiments on segment 'a'. We first randomly select a subset of segment 'a' consisting of 1000 samples and perform our experiments on this subset. Then we repeat the experiments on the whole segment 'a'. A sample of data of this segment is shown in Fig. 5.

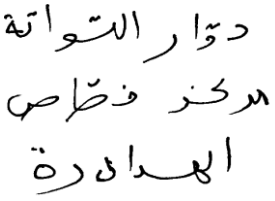


Fig. 8. Some sample handwritten words from the INF/ENIT dataset.

We perform four experiments on the 1000-sample data and the whole samples of segment 'a'. Details of the three experiments A, B, and C (in which the blank label is present) are shown in Table 3. The first three experiments are named A, B, and C and the fourth experiment is similar to B with the difference that the blank label is removed. The reason that experiments A and C are not repeated without blank is that in these experiments some of the labels have only one state in CTC. In experiment C, the number of sub-labels ranges between 1 and 3 in proportion to the average length of data of that label. The values of "Input Block" and "Hidden Block" differ between the three experiments and is chosen in a way that ensures that there are enough data for all sub-labels of the input data. This dataset is much harder than the printed digits dataset and so the number of cells in LSTM, i.e. the value of "Hidden Size", is chosen in a way that gives the best results for each experiment. Since by increasing the number of sub-labels the network should learn more details about data, the number of LSTM cells should increase accordingly. In all experiments, we use the "early stopping" method to avoid overfitting. If the error rate on validation set does not decrease for 40 consecutive epochs, we stop training and return the network with the lowest validation error.

Table 3. Parameters For IFN/ENIT Experiments

Experiment	A	B	C
Hidden Size	2, 10, 50	4, 20, 80	4, 15, 64
Subsample Size	6, 20	6, 20	6, 20
Hidden Block	3x4, 2x4	2x4, 2x4	2x3, 1x3
Input Block	3x4	2x4	3x3
Learn Rate	1e-4	1e-4	1e-4
momentum	0.9	0.9	0.9
optimizer	Steepest Descent	Steepest Descent	Steepest Descent
Number of Sub labels	1	2	1 or 2 or 3

5-3. 1000 sample subset of IFN/ENIT Dataset

The 1000 samples of this dataset have been chosen randomly from segment 'a' of the IFN/ENIT dataset. We use 800 samples for training, 100 for validation, and 100 for testing. The results of the experiments on this dataset are shown in Table 4. By comparing experiments A and B one can observe that increasing the number of sub-labels from one to two has increased recognition accuracy. In addition, by comparing experiments B and C one can see that the appropriate choice of the number of sub-labels has improved the results. Now,

we investigate the role of the blank label by considering experiment B with and without blank. It can be seen that the results obtained with blank are much higher than those obtained without it. Another important observation is that without blank, the training error is much higher (in addition to the test error). This shows that removal of blank leads to a learning machine with much lower capacity (possibly due to optimization issues). Therefore, we designed another experiment in which we removed the stopping condition and allowed the network to obtain a much lower error on the training set. However, the results obtained by this method did not differ considerably from those obtained by early stopping.

5-4. Segment 'a' of the IFN/ENIT Dataset

Segment 'a' of the IFN/ENIT dataset consists of 6537 samples from which 5702 are used for training, 426 for validation, and 409 for testing. The results are given in Table 5. By comparing this Table with Table 4, we see that by increasing the number of training samples the model is much better optimized and much higher accuracy results have been obtained over the validation and test sets. In addition, similar to the previous experiments, we have obtained better results in experiment B in comparison with experiment A, because of using two sub-labels for each label. We have achieved the best results in experiment C in which the number of sub-labels is chosen in proportion to the average length of each label. By comparing experiment B in the two cases with and without blank we see that the results with blank are much better than those without blank. This shows that the use of blank in datasets with complex data leads to improved recognition accuracy.

5-5. Analyzing the effect of blank

In the previous sections, we reported the results of our experiments on the printed digits and IFN/ENIT datasets. The first dataset was very simple as every sample contained three labels with equal lengths and widths. We observed that the use of blank deteriorated the speed of obtaining a solution. In contrast, in experiments that did not use blank, much better results had been obtained in early epochs. Because of the simplicity of this dataset, all experiments ended with 100% accuracy. However, the IFN/ENIT dataset was much more complex: having labels with different average lengths, and 12 times more labels. For example, Fig. 3 shows different styles of letters which differ in length and writing style. Because of this complexity, the network should have a flexible structure which can learn all states of each label. The results showed that the use of blank has a huge effect on improving the recognition accuracy. It can be deduced that even when the number of states is more than one, use of blank improved the recognition results on real-world datasets. After doing this work, we found that similar observations have been done in [23], confirming the superiority of RNN-CTC models using blank over those that do not. In contrast to the results reported in [23], we found that the combination of increasing the number of states per label and using blank gives the best results. This difference in observations may be due to the fact that we have modified the structure of the beneath neural network appropriately to cope with the increase in the number of states in M-CTC.

Table 4. Results For 1000 Data From IFN/ENIT

Experiment	With blank						Without blank					
	CTC Error			Label Error %			CTC Error			Label Error %		
	Train	Val	Test	Train	Val	Test	Train	Val	Test	Train	Val	Test
A	0.53	17.45	11.93	0.22	40.22	41.35	-	-	-	-	-	-
B	0.57	28.27	19.56	0.03	33.77	33.30	76.69	95.60	83.88	68.71	76.26	81.75
C	0.96	25.89	18.85	0.51	31.64	30.25	-	-	-	-	-	-

Table 5. Results For set 'a' of IFN/ENIT

Experiment	With blank						Without blank					
	CTC Error			Label Error %			CTC Error			Label Error %		
	Train	Val	Test	Train	Val	Test	Train	Val	Test	Train	Val	Test
A	0.41	6.69	5.46	0.49	19.45	17.98	-	-	-	-	-	-
B	0.33	9.43	8.18	0.07	13.28	12.85	25.33	36.99	37.73	29.13	46.24	46.28
C	0.32	8.36	7.53	0.15	12.66	10.86	-	-	-	-	-	-

5-6. Conclusions and future works

In this paper, we extended CTC to model and learn intra-label relations. We achieved this goal by increasing the number of states in CTC for each label. In other words, we considered several states/sub-labels for each label. Experimental results showed that the proposed extension improves the recognition accuracy. In addition, we studied the role of blank. We showed that by increasing the number of states, models without blank can learn repetitive consecutive labels. Our experiments showed that while on simple tasks a model without blank may converge faster, on real-world complex datasets use of blank significantly improves the results. Although we restrict the model to see the sub-labels in true order during training, there is not a similar obligation during test time. In other words, it is possible during test time that the model predict the second sub-label without seeing the first sub-label. In future, we want to analyze the impact of this discrepancy at training and testing time on the recognition accuracy.

References

[1] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks", *in Proceedings of the 23rd international conference on Machine learning*, pp. 369-376, 2006.

[2] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult", *IEEE transactions on neural networks*, vol. 5, pp. 157-166, 1994.

[3] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies", *ed: A field guide to dynamical recurrent neural networks. IEEE Press*, 2001.

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory", *Neural computation*, vol. 9, pp. 1735-1780, 1997.

[5] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks", *Journal of machine learning research*, vol. 3, pp. 115-143, 2002.

[6] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM networks for improved phoneme classification and recognition", *in International Conference on Artificial Neural Networks*, pp. 799-804, 2005.

[7] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures", *Neural Networks*, vol. 18, pp. 602-610, 2005.

[8] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks", *in Advances in neural information processing systems*, pp. 545-552, 2009.

[9] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition", *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, pp. 855-868, 2009.

[10] M. Wöllmer, F. Eyben, B. Schuller, and G. Rigoll, "Spoken term detection with connectionist temporal classification: a novel hybrid ctc-dbn decoder", *in 2010 IEEE International Conference on Acoustics*, Speech and Signal Processing, pp. 5274-5277, 2010.

[11] M. Wöllmer, B. Schuller, and G. Rigoll, "Probabilistic ASR feature extraction applying context-sensitive connectionist temporal classification networks", *in 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 7125-7129, 2013.

[12] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks", *in 2013 IEEE international conference on acoustics*, speech and signal processing, pp. 6645-6649, 2013.

[13] M. Wöllmer, F. Weninger, J. Geiger, B. Schuller, and G. Rigoll, "Noise robust ASR in reverberated multisource environments applying convolutive NMF and Long Short-Term Memory", *Computer Speech & Language*, vol. 27, pp. 780-797, 2013.

[14] A. Graves and N. Jaitly, "Towards End-To-End Speech Recognition with Recurrent Neural Networks", *in ICML*, pp. 1764-1772, 2014.

[15] D.-A. Huang, L. Fei-Fei, and J. C. Niebles, "Connectionist Temporal Modeling for Weakly Supervised Action Labeling", arXiv preprint arXiv:1607.08584, 2016.

[16] M. Woellmer, B. Schuller, and G. Rigoll, "Keyword spotting exploiting long short-term memory", *Speech Communication*, vol. 55, pp. 252-265, 2013.

[17] S. Fernández, A. Graves, and J. Schmidhuber, "Sequence Labelling in Structured Domains with Hierarchical Recurrent Neural Networks", *in IJCAI*, pp. 774-779, 2007.

[18] A. A. Atashin, K. Ghiasi-Shirazi, and A. Harati, "Training LDCRF model on unsegmented sequences using Connectionist Temporal Classification", *arXiv preprint arXiv:1606.08051*, 2016.

[19] A. S. Lotfabadi, K. Ghiasi-Shirazi, and A. Harati, "Modeling intra-label dynamics in connectionist temporal classification", *in 2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 367-371, 2017.

[20] A. Graves, "Neural Networks," in Supervised Sequence Labelling with Recurrent Neural Networks, ed: Springer, pp. 15-35. , 2012

[21] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze, and H. Amiri, "IFN/ENIT-database of handwritten

Arabic words", *in Proc. of CIFED*, pp. 127-136, 2002.

[22] A. Graves, "RNNLIB: A recurrent neural network library for sequence learning problems", [OL][2015–07-10], 2013.

[23] T. Bluche, H. Ney, J. Louradour, and C. Kermorvant, "Framewise and CTC training of Neural Networks for handwriting recognition", *in Document Analysis and Recognition (ICDAR), 2015 13th International Conference on*, pp. 81-85. , 2015.

Table of Contents: