IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 27, 2018, accepted January 8, 2019, date of publication January 23, 2019, date of current version February 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2893501

Network Experience Scheduling and Routing Approach for Big Data Transmission in the Internet of Things

FADI AL-TURJMAN[1], (Member, IEEE), LEONARDO MOSTARDA[2], (Member, IEEE), ENVER EVER[3], (Member, IEEE), AHMED DARWISH[3], AND NAZIHA SHEKH KHALIL[3]

[1]Computer Engineering Department, Antalya Bilim University, 290 Antalya, Turkey
[2]Computer Science Department, Camerino University, 62032 Camerino, Italy
[3]Computer Engineering, Middle East Technical University, Northern Cyprus Campus, Güzelyurt, Mersin 10, Turkey

Corresponding author: Leonardo Mostarda (leonardo.mostarda@unicam.it)

ABSTRACT The recent developments in the Internet of Things related technologies have caused a shift towards smart applications such as smart cities, smart homes, smart education systems, e-health, and online applications to run businesses. These, in turn, have introduced significant additional loads to the existing network infrastructures. In addition, these applications use big data and require relatively short response times. In this paper, we are introducing a new scheduling and routing approach to enhance the end user experience, and utilize the network resources by providing improved transmission speed for the big data applications. The approach considers the source and destination requirements in terms of data size, expected delay, link load, and link capacity. Extensive simulations are performed, and the results obtained show the efficiency of our approach against other competitive approaches in terms of in-network delay, network throughput, and dropped packets.

INDEX TERMS Data delivery, Internet of Things (IoT), wireless networks, routing, big data.

I. INTRODUCTION

In Internet of Things (IoT), the load caused by the data of heterogeneous nature is getting larger dramatically, due to high demand from various networks, such as wireless sensor networks (WSNs), data centers, cellular networks, personal and environmental monitoring devices which are also connected to the cloud [1]. According to the Cisco Visual Networking Index (VNI) [2], [3], there is a dramatic increase in mobile data traffic, which goes up to 18 times over the past five years. The volume of the traffic is expected to reach seven times as much as in 2016 by 2021. Moreover, the number of mobile devices is expected to become 11.6 billion by 2021.

Currently, IoT technology provides solutions to various systems. D2D communications can solve the problem of Vehicle-to-Vehicle (V2V) communications in Intelligent Traffic Systems (ITSs) for traffic control and safety applications [4]–[6]. Narrowband IoT (NB-IoT) is another low power cellular technology which provides a way to connect different devices that require a limited amount of data over a long period. NB-IoT has been used in various scenarios such as intelligent parking, intelligent meter reading, and smart hospitals [7], [8]. Moreover, health monitoring, agriculture, smart city, and smart industry are other applications where IoT plays a significant role to improve the quality of life of people [9].

The breakthroughs in IoT applications have been followed by significant investments to the field. European Union (EU) has invested more than 100 million Euros in a series of projects through Seventh EU Framework Programme (FP7 for R&D), and these projects will be actively deployed in smart grid, intelligent transportation, smart cities, etc. Similarly, South Korea invested 27.8 million U.S. dollars in IoT fundamental technology development, IoT testbed advancement, and IoT standardization [10].

The IoT based applications are expected to be rapidly deployed in various areas, and the additional load caused by communicating billions of nodes can significantly affect the user experience. Many studies in the existing literature investigate efficient ways to transmit data back and forth between devices in the network. Different distributed and centralized (cloud-based) approaches are introduced to solve the problems caused by the additional traffic,

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

where some other studies focus on security and privacy related issues [11]–[14]. In the centralized (cloud-based) approaches, the network information is used to enhance the transmission quality between the devices, since using the network information can enhance the routing of the data in the network. In this paper as well, we introduce a centralized scheduling and routing algorithm to enhance the end user experience.

In this paper, IoT network nodes are classified as source node, intermediate node, and destination node. The source nodes generate the data that needs to be transmitted, the intermediate node describes the node that data pass through to reach the destination, and the destination node is the node that will receive and execute the data generated in the source node. On the other hand, the cloud server is a server that can reach all nodes; it can know the status of the nodes in case there is a failure in one of the nodes in the network. The server is used to assign a path for the packets from the source node to the destination node.

Different centralized approaches focus on the data size, data type, and the applications used for the data to determine the path for a packet from the source to the destination [1], [3], [15]–[22]. However, these approaches did not consider the network experience for that type of packets and the size packets. In this paper, we are introducing a novel scheduling and routing approach that uses dynamic programming to determine the estimated best path that the packet can take according to its type, size, and the previous network experience for similar packets and the expected performance of the network. The contributions of this paper can be summarized as follows:

1. A new centralized routing and scheduling algorithm is proposed.
2. The new approach is evaluated through extensive simulations with another centralized routing and scheduling algorithms, and the results show that it performs better than existing ones in terms of a variety of quality of service (QoS) measures.

The paper has been organized as follows. In Section II, we provide a brief review of the related work. Section III contains the assumptions, and system models for this paper. In Section IV, we identify and introduce our approach. In Section V, the details of the simulation, and the results used to evaluate the performance of the proposed approaches are discussed. In Section VI a brief summary and final remarks are presented.

II. RELATED WORKS

When application areas of IoT such as smart cities, smart transportation systems, cellular networks, and wireless sensor networks (WSN), are considered, one can easily see that many of them make use of wireless communications/transmissions quite extensively. It is possible to specify two main approaches that are focusing on improving the bandwidth usage and time delay in data delivery, which are distributed, and centralized approaches [1].

A. DISTRIBUTED APPROACHES

The Distributed approach is mainly implemented through two popular ad-hoc routing protocols. The first one, DSR stands for dynamic source routing. It performs route discovery and route maintenance following a reactive approach [23]. In route discovery, it discovers the route from source to destination. If a node finds more than one way to reach the same destination, it stores that information. In case one route is broken then it can use one of the already stored alternative routes. The major problem with this approach that it does not perform well in high mobility, and high scale networks, as packet loss is high due to time spent to discover routes, where it checks all possible routes (broadcast the packet to all neighbors to check if there is a route or not). AODV stands for Ad Hoc on Demand Distance Vector. In this approach, if source node wants to communicate with the destination node, it will broadcast the RREQ (route request) to neighboring node and wait for the RREP (route reply) for a specific time period. Unlike the DSR, there is no need to update each and every routing table in the network. Only active nodes are required to be (up to date). If any link fails in the route, then all other active nodes will be updated, about the fact that this link is no more available and remove this link from the routing table. The problem with this approach is that it consumes extra bandwidth, comes with lower data rate, high error rate, cannot adapt to dynamic topologies and performs bad in cases of high scale networks especially in terms of energy efficiency [19], [23]. Nevertheless, different variations of ADOV have been developed in the recent years. These Developed versions show high resilience against DoS attacks, using certain protocols such as PARSER [24]. However these variations rely on highly intelligent nodes compared with previous versions which in turn introduce higher demands in terms of computing power.

B. CENTRALIZED APPROACHES

These Distributed approaches has significant drawbacks such as high rate of packet loss, bad performance in high scale networks, lower data rate and high error rate, which makes the centralized network approach more preferable. In centralized network approach, all users connect to a central server, which is the acting agent for all communications. This server would store both the communications and the user account information. There are different types of centralized approaches, some require the packets to pass through the centralized node (sink node), where the cloud computing [13], [14] enables users of sensor networks to fully utilize wireless technologies in storing, sharing and retrieving the collected data by sensors anytime and anywhere. However, the support of the cloud may have some limitations due to security and privacy related issues. Other approaches manage the flow of the packet in the network. CAR Context-aware routing (CAR), is a new developed centralized approach that uses the cloud as an extra level of data-request processing to improve network performance in terms of data delivery [1]. It considers source

and destination requirements in terms of data size, delay, link capacity, and available applications on the operating devices as well. This approach is unlike other approaches where intermediate nodes are randomly selected regardless of the application. In the central server (in the cloud) the shortest/fastest path is calculated, which is defined as the minimum number of hops (intermediate nodes) and maximum capacity, supporting the requester application. Once the link is established, a peer-to-peer communication model is applied to maintain the lowest delay and most efficient bandwidth utilization between two end users (clients) of the link. The source node requests a path from the centralized server (possible path, according to the application needed), then choose the best (shortest) path. The cloud server sends path for the packet from source to destination according to the available paths, data size, applications needed. The intermediate nodes check the neighbor node to send the packet if a failure occurs, it requests a new path from the cloud server. The destination node checks the packet and the route between it and the source. The shortest path algorithm used to calculate the path is Dijkstra's Algorithm. The contest-aware approach outperformed CUR (context-unaware routing) regarding average data request in-queue delay, in-network delay, and drop rate [1].

Centralized channel assignment approach, is developed for the Multi-Channel wireless mesh networks (WMN). In this approach, a channel allocation algorithm is introduced, and the routes are assigned for the packets through one of the following:

- shortest path routing and
- randomized multi-path routing

Channel assignment depends on the expected load on each virtual link, which depends on routing. Given a set of communicating node pairs, the expected traffic between them, and the virtual link capacities, the routing algorithm determines the route through the network for each communicating node pair. The resulting routes populate the routing tables of all the nodes. In [21], it is shown that multi-channel affects the overall time of a packet in the system significantly. In this study as well, we use the Randomized multi-path routing, but it is modified so that the random assignment process is performed from among the possible shortest paths.

Please note that none of the proposed approaches introduce a comprehensive framework that uses the previous network experience to enhance the end user experience. To the best of our knowledge, in this paper, a new network experience scheduling and a routing approach is introduced that uses previous network experience for similar packets type, and packet size, to determine the estimated best path for that packet to be transmitted from source node to the destination node, taking into consideration the network performance for the first time.

III. SYSTEM MODEL AND ASSUMPTIONS

In this section, we explain our assumptions and our system models. The network under study is assumed to be single channel rather than multi-channel, to be able to compare the efficiency of routing approaches. In Table 1 abbreviations used in this study are introduced. Nodes describe the devices that generate, receive, or transmit the data that needs to be delivered. Packet stands for the data that is going to be transmitted from source $node_s$ to destination $node_d$. All the nodes within radius r can communicate with each other $\{node_1, \ldots, node_m\}$. All of the nodes in the network are connected to the *server*. Each $node_i$ has a buffer of size q. $node_i$ cannot generate a new p if the buffer is full and currently executing p_j. Each p_j has its own service time, depending on the packet type, and packet size. All nodes in the network $\{node_1, \ldots, node_n\}$ share the same functionality, to measure the performance of different proposed routing approaches.

TABLE 1. Notations.

Symbol	Quantity
$\boldsymbol{node_i}$	Current / intermediate node
$\boldsymbol{node_s}$	Source node magnetic induction
$\boldsymbol{node_d}$	Destination Node
$\boldsymbol{qMax}$	Maximum buffer size
$\boldsymbol{p}$	Packet
$\boldsymbol{server}$	Cloud Server: can reach all node within radius
$\boldsymbol{r}$	specific magnetization
$\boldsymbol{n}$	Number of nodes in the network
$\boldsymbol{m}$	Number of neighbor nodes
Ψ	Overall delay moment
$\varphi\,(\boldsymbol{node_i}, \boldsymbol{node_j})$	The delay between two consecutive nodes

A. NETWORK AND TRAFFIC MODELS

The nodes represent the devices that generate, transmit, and receive the packets. The nodes are connected through nondirected links [25]. The nodes use IEEE 802.11a/g for transmission, with data rate of 54Mbps [26]. $node_i$ can communicate with all its neighbors $\{node_1, \ldots, node_m\}$ within radius r, as illustrated in FIGURE 1. The radius varies from one node to another, and the nodes are distributed uniformly. At the center, the *server* is located. All nodes in the network $\{node_1, \ldots, node_n\}$ can communicate with the server at any time, requesting a path for packet p_j from $node_s$ to $node_d$. In case there is a failure in $node_i$ (or $node_i$ is out of server coverage) all paths, that include $node_i$ are going to be updated accordingly. In case there is an update in the position of one of the nodes $node_i$, paths are going to be updated accordingly. The links between the nodes are updated from the server, divided into 4 categories, Blocked links, High load links, Normal load links, and low load links.

B. NODE MODEL

For each node, there is a queue (buffer) that contains, the packets $\{p_1, \ldots, p_{qMax}\}$ that are generated from the node, or transmitted to the node. The queue (buffer) follows First in First out (FIFO) scheduling, so the priority for the packets are according to the arrival time of the packet p_j at $node_i$. For each p_j there is a service time that depends on the packet type, and

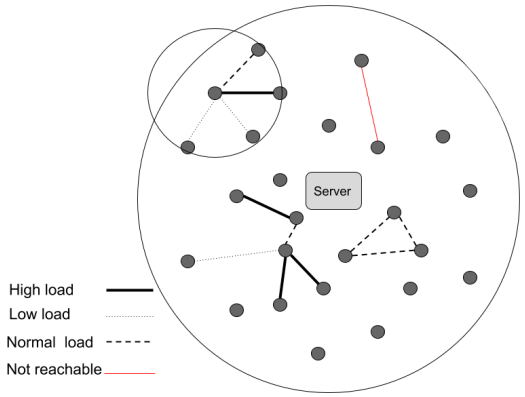

FIGURE 1. Illustration of a centralized network, with path weight estimation.

packet size assuming all nodes share the same performance (computational power is 6.75KB/ms for each packet in the network). Once there is a packet in the queue, the node $node_i$ checks if the packet has a path or not, or its destination node is itself. In case of no path to the destination, the node $node_i$ request a path from the server to the destination $node_d$. In case the following node in the path is not reachable the packet will be pushed back into the queue (only 5 times, then the packet will be discarded). Once a packet is generated, the destination node $node_d$ expectes the packet from the source node $node_s$. If one of the expected packets p_j did not come within limited time, $node_d$ will request the packet again to be sent from $node_s$.

C. CLOUD SERVER MODEL

The server model, differs from one algorithm to another, as it is the critical phase. Each $node_s$ requests a path from the *server*. According to the proposed algorithm the *server* assigns the path for the packet p from $node_s$ to $node_d$. In case there is a failure in one of the nodes $node_i$, or the node is out of the range of its current neighbors $\{node_1, \ldots, node_m\}$, the paths that include $node_i$ as intermediate node are regenerated, corresponding to the new position of $node_i$.

D. MODELS USED FOR EVALUATION

In this paper, we consider End-to-End time as the overall delay time for a packet in the system. We consider the delay definition which depends on the number of hops (intermediate nodes) as $\boldsymbol{\varphi}$ ($\boldsymbol{node_i}$, $\boldsymbol{node_j}$) [27], and queueing time $\boldsymbol{\Sigma\Psi(node_i)}$. $\boldsymbol{\varphi}$ represents the delay between two nodes and $\boldsymbol{\varphi}$ represents the queuing time that packet $\boldsymbol{p_i}$ took in the queue of the $\boldsymbol{node_i}$.

The throughput is considered as the number of packets successfully transmitted from the source $\boldsymbol{node_s}$ to the destination $\boldsymbol{node_d}$ [27].

Path elimination model is a part of the network experience process. The server use path elimination method to eliminate certain paths that are expected to cause a drop for the packet, or a very high delay time, which exceeds the packet life time.

Finally, expected performance method is used in our proposed approach. It records the requests from the source nodes $node_s$ into memory and according to the previous performance of the network to the requested type of packet. After the elimination process, the server uses the expected performance of the network to choose the most suitable path for the given packet p_j, according to the scheduling of the possible paths to $node_d$.

IV. NETWORK EXPERIENCE APPROACH

In Network experience approach, the packet passes through two main phases, 1) the node phase, 2) server phase. In the node phase, the packet can be in source node $node_s$, where there is no path identified to the required destination. In the intermediate node $node_i$, the node checks the path of the packet, and transmit the packet p to the following node in the packet path, if and only if, the following node is reachable. The packet is executed in the destination node $node_d$. During the execution of a packet p_j, $node_d$ cannot transmit packets to the neighbours $\{node_1, \ldots, node_m\}$, however, it can receive packets, and those packets are added to the $node_d$ queue (buffer). The packet p_j will be dropped in the case that the queue is full. The packet time limit, identify the maximum time the $node_d$ waits to receive p_j from $node_s$, before sending a request message to $node_s$, to send the packet p_j again. Once a packet is received by $node_d$, the node removes it from the expected packets list, and checks if the packet was delivered before or not to execute it.

Once the *server* receives a request for a path from $node_s$ to $node_d$, the *server* relies on the previous network experience for similar packets using expected performance model, and the current network estimated load to eliminate the paths using path elimination model, that might cause a drop for a packet, or the time delay for that packet p_j to reach $node_d$ exceeds the packet time limit. The threshold, for the path elimination, differs from one packet to another, as it depends on the network experience for that type of packets and current load on the system. The *server* estimates the delay in one path, according to previous requests, and estimates service time for packet p_j, according to previous experience for the similar packet type, and packet size.

Three different routing approaches are proposed in this study which use the network experience to determine the path for p_j, from $node_s$ to $node_d$. The first routing approach is minimum load approach, which chooses the path that has the minimum load to transmit the packet from $node_s$ to $node_d$, and the chosen path will be recorded for the following requests with similar packets, to update network experience data. The second approach is the shortest path approach. The approach chooses from the possible paths (with a low percentage of packet drop, and acceptable time delay), the shortest to send the packet from $node_s$ to $node_d$, and the data is recorded for the next request accordingly. The third approach is random assignment approach, where after eliminating the paths that have a high percentage of packet drop, or have a

high time delay, among the remaining paths one of them is chosen randomly following uniform distribution.

TABLE 2. Variables and functions.

Name	Definition
currentID	The node ID
setupMessage	Path for the packet assigned by the server
q	Queue (node buffer)
p	Packet
message	setupMessage that is received by the server
currentTime	Current Time
resultedPath	The result path that is assigned for the setupMessage
expectedList	List of expected packets in the destination node
setupMessageRequests	List of setup messages that the server receives from the nodes
paths	Limited list of possible paths from node to another node in the network
priorityQueue	Priority Queue that prioritize the list of paths according to the proposed routing approach (ex. minimum weight, minimum distance, random)
packetTimeLimit	Maximum time for the packet in the system before it is re-Requested
negativeResponse	Null path
HasRoute	Function that checks if packet has route or not
RequestSetupMessage	Function that requests a path for a packet from the server
SendPacket	Function that sends the packet to the assigned node
Timeout	Function that checks if the setupMessage is timeout message
Execute	Function that executes the packet in the destination node
RequestReTransmission	Function that Request the specified packet from the source node
UpdateNodesStatus	Function that updates the status of the nodes in the network
PossiblePaths	Function that returns a set of possible paths from one node to another in the network
PathElimination	Function that returns a path from a set of paths using Path Elimination model
UpdatEstimationLoadTable	Function that updates the estimation load table with the new assigned path
SendSetupMessage	Function that sends the resulted path to the node that requested a setupMessage
SendNegativeResponse	Function that sends negative response to the node that requested a setupMessage
PathWeight	Function that returns the weight of the path using EstimationLoadTable
EstimatedDelay	Function that returns the estimated delay for a packet type (using previous network experience)

TABLE 2 shows the variables and functions that are going to be used in the algorithms, and their definitions.

Pseudocode description of the algorithm used for the source node $node_s$ is given below (Algorithm 1). The source node generates the msg data and requests a path to the destination node $node_d$ from the server. Following that, the server sends the path to the source node $node_s$. In case of failure, if there is no path to the destination node $node_d$, server will send no path to the source (negative response). In case of no path, the $node_s$ will push the packet to the queue (buffer), and will process it in its order. In case there is a path, source node will start transmitting the packet p_j to the following node $node_i$ in the path. Once the path is assigned, the packet details are added to destination node $node_d$.

Algorithm 1 For Source Node S

```
p = null
if q.isEmpty() == False :
    p = q.pop()
    if HasRoute(p) == False:
        setupMessage = RequestSetupMessage(p.type, currentID, p.destination)
        if setupMessage == negativeResponse or Timeout(setupMessage) == True :
            q.push(p)
else :
        SendPacket(p.followingNode, p)
```

Algorithm 2 For Intermediate Node I

```
p = null
while q.isEmpty() == False :
    p = q.pop()
        if HasRoute(p) == True and IsReachable(p.followingNode) == True :
        SendPacket(p.followingNode, p)
    else :
        setupMessage = RequestSetupMessage(p.type, currentID, p.destination)
        if setupMessage == negativeResponse or Timeout(setupMessage) == True :
            q.push(p)
        else :
            SendPacket(p.followingNode, p)
```

Pseudocode description of the approach used for the intermediate node $node_i$ is given in Algorithm 2. $node_i$ checks the path for the packet p_j, then checks if the following node in the path is reachable or not. If the following node in the path is reachable, it starts transmitting the packet p_j, otherwise, $node_i$ sends setup message to the server, requesting new path for the packet p_j. Upon receiving the setup message from the server, the $node_i$ start transmitting to the following node in the path of p_j, or push it back to the buffer.

Pseudocode description of algorithm used for destination node $node_d$ is given below (Algorithm 3). The node checks if the packet p_j destination node is the current node $node_d$. Then checks if the packet is fully received. If yes then the packet is removed from the expected list. Once the packet is received, the node starts executing the packet. In case that the packet is duplicated, $node_d$ will drop the packet as it was previously executed.

A pseudocode description is also provided for the server in Algorithm 4. The server updates the status of the nodes in the system (checks if there is a failure in one of the nodes), following this, it updates the status of the nodes in

Algorithm 3 For Destination Node D

```
if q.isEmpty() == False :
    p = q.pop()
    if p.destination == currentID :
        expectedList.remove(p.id)
        Execute(p)
for p in expectedList :
    if currentTime - p.requestTime >= packetTimeLimit:
        RequestReTransmission(p)
```

Algorithm 4 For Cloud Server CS

```
UpdateNodesStatus()
if setupMessageRequests.isEmpty() == False :
    message = setupMessageRequests.pop()
    paths = PossiblePaths(message.source, message. destination)
    resultedPath = PathElimination(paths, message.type)
    if path != null :
        UpdateEstimationLoadTable(resultedPath)
        SendSetupMessage(message.source, resultedPath)
    else :
        SendNegativeResponse(message.source)
```

the estimated load table. If there is a setup message request for a packet from source (S) node $node_s$ to destination (D) node $node_d$, the server checks the possible paths from S to D, and run Path Elimination Model. If a path exists from S to D, the server assigns the path with the highest priority according to the proposed routing algorithm (Minimum load path. Minimum distance path, Random path) to the packet and sends the setup message to S node $node_s$. Once a path is assigned to the packet, expected list in $node_d$ is updated with the new packet details. Nevertheless, the estimation load table is updated for the corresponding nodes in the assigned path. The time complexity of updating the estimation load table time is O(N), depending on the number of nodes and a constant number of paths, using the dynamic programming.

Pseudocode description of the path elimination model is presented as Algorithm 5. This model checks the set of limited number of paths from source to destination. The paths that exceeds the packet time in the system + estimated delay time, are eliminated from the list of possible paths. Prioritization process follows the elimination process, as the model prioritize the list and chooses the best match path according to the routing algorithm (Minimum load path. Minimum distance path, Random path).

Algorithm 5 Path Elimination Model

```
priorityQueue = null
for path in possiblePaths:
    if PathWeight(path) <= PacketTimeLimit + EstimationDelay(packetType):
        priorityQueue.push(path)
if priorityQueue.isEmpty() == False :
    return priorityQueue.top()
else :
    return null
```

V. PERFORMANCE EVALUATION

The proposed Network experience approach is simulated in this section. The in house simulation program is developed using Java (OOP). The new approach is evaluated against, context-awareness (CAR), and Random routing approaches. These two approaches represent our comparison baselines in this paper. We have conducted on the packet level that allows us to measure the performance of these different approaches.

The generation of the network nodes $\{\boldsymbol{node_1}, \ldots, \boldsymbol{node_n}\}$ follows the proposed Network and Traffic Models. According to analyses done on the data retrieved from the European Project SmartSantander. The traffic distribution, aggregated by temperature bins, follows up a Poisson distribution model [28]. The packets arrival time for our system follows Poisson distribution for an identified number of occurrences. The possible paths were generated (and updated) using a modified version of Dijkstra's Algorithm. The arrival time was assigned to the generated packet p at the beginning of the simulation, with a source and destination for each packet. Simulated network nodes $\{\boldsymbol{node_1}, \ldots, \boldsymbol{node_n}\}$ maintain the proposed Node Model. Simulated network ***server*** maintains the Cloud Server Model.

A. PERFORMANCE MATRIX

To compare the performance of Network experience approach, the following three performance metrics are used:

1) Rate of packet drops: number of dropped packets over the total number of packets, the packet drops in two cases:
   a) If there is a failure in the node (outside the range, a problem occurred)
   b) if the queue of the node (the buffer) is full and cannot receive any packet at the moment
2) Throughput: the number of packets served per unit time.
3) Average End-to-End Delay: this is the average time that each data packet spends in the network, from the source node to the destination node.

While studying these performance metrics, we vary the following parameters:

1) Arrival time: represents the time that the event will occur in the system.
2) Service time: represent the time is spent being served in the destination node.
3) Type of packet: represents the packet in the system which is text, voice, and video, etc. (it vary from network to another, as the priority is given to one kind over another, or they are all equal).
4) Number of packets arriving in the system: represents the total number of packets the system

5) Queue size: represent the size of the buffer queue each node has
6) Number of nodes: The number of nodes that are covered by the centralized node (the server node)

B. SIMULATION RESULTS

The simulation program employed is a discrete event simulation program. An event-based scheduling approach is taken into account which depends on the events and their effects to the system state. The system stops sampling once the Average End-to-End, and Throughput of all proposed approaches fall within the confidence interval. Relative precision which is one of the most commonly used stopping criterion is employed for the simulation. In this method, the simulation is stopped at the first checkpoint when the condition $\delta \leq \delta\max$ is met. Where $\delta\max$ is the maximum acceptable value of the relative precision of confidence intervals at the $100(1-\alpha)$ % significance level, $0 < \delta\max < 1$. The results obtained from the simulations are within the confidence interval of 5 % with a confidence level of 95 %; therefore, in the simulation, both default values for α and δ are set to 0.05. Furthermore, 20 trials were used to form simulation average for the matrices. Parameters were formed as a baseline using an average of 20 trials per simulation for the behaviour of the system. A set of tests were made to measure different aspects of the system behaviour while simulating the proposed approaches. The data collected from the simulation has the same behaviour as the benchmark approaches presented in studies [1], [21].

1) CONFIDENCE TEST

In modeling and simulations confidence intervals are frequently used as a method of validation [26], [29]. For $x_1, x_2, x_3, \ldots, x_n$ population, the sample mean is calculated as follows.

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{1}$$

And the sample standard deviation s as

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}} \tag{2}$$

The confidence interval is point estimate $\pm$ error margin. It is calculated between the interval [L,U], where L is the lower limit, and U is the upper limit. The confidence interval for a population of x is

$$[\bar{x} - z_c \frac{s}{\sqrt{n}}, \bar{x} + z_c \frac{s}{\sqrt{n}}] \tag{3}$$

where z_c is the critical value e for the normal distribution for confidence level c. The values for z_c can be found in statistical tables, where TABLE 3 shows a sample from the table.

C. BUFFER SIZE TEST

The test measures the effects of the node's buffer size on the network performance. Fixing the other parameters, as transmission rate 54Mbps for IEEE 802.11a/g, arrival rate is 4 packets / milliseconds following Poisson distribution, the service time for the packet is 15 milliseconds (packet size is 54Kb), for 10000 packets, and 50 nodes network. As seen in FIGURE 2, Random routing and network experience with random routing are the worst in the list, however, CAR approach and Network experience approach with minimum weight (NEAMW) routing behave similarly. Nevertheless, Network experience approach with the minimum distance (NEAMD) is the best in terms of minimum Average End-to-End. FIGURE 3, shows that almost all approaches had the same throughput, which implies that there is no such effect for the buffer size on the Throughput using different routing and scheduling approaches. In addition to that, in FIGURE 4, the Number of Re-requested packets for Random routing approach is the highest, followed by modified Network experience with the random assigning (NEAR) route. On the other hand, the Network experience approach with minimum distance and minimum weight behaved better than CAR approach.

TABLE 3. Normal distribution for confidence level.

Confidence Level c	Normal z	d.f = 5	d.f = 10
0.80	1.282	1.476	1.372
0.90	1.684	2.015	1.812
0.95	1.960	2.571	2.228

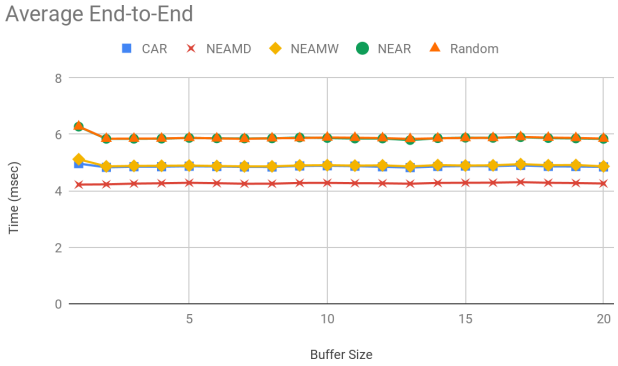

FIGURE 2. Effects of buffer size on the average End-to-End time for the packets.

1) NUMBER OF NODES TEST

This test measures the effects of the number of nodes in the network system performance. The arrival rate is again taken as 4 packets / milliseconds following Poisson distribution, where the service time of 15 milliseconds (packet size is 54Kb), the number of packets is 30∗(number of nodes). As seen in FIGURE 5, FIGURE 6, FIGURE 7, all proposed approaches behaved similarly to the change in the number of nodes in the network. We can observe that the number of nodes in the network is not one of the main factors

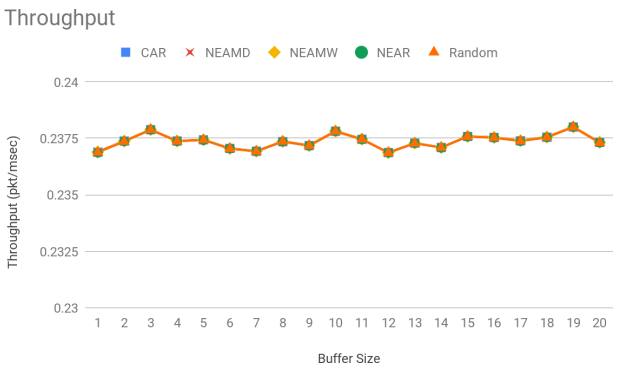

FIGURE 3. Effects of buffer size on the system throughput.

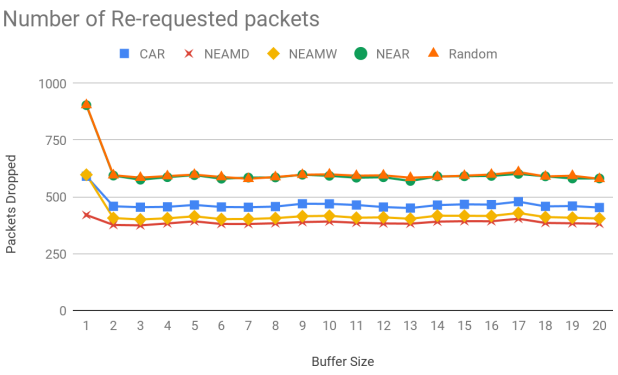

FIGURE 4. Effects of buffer size on the number of re-requested packets.

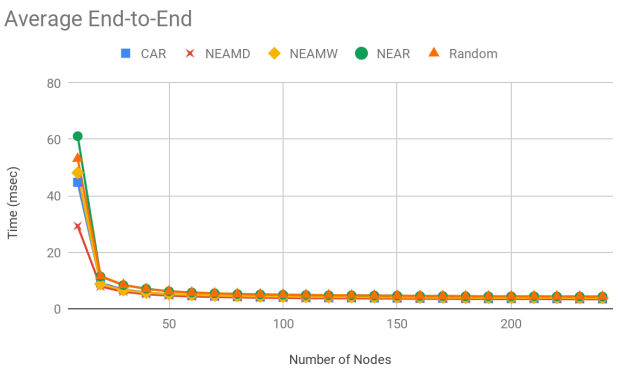

FIGURE 5. Effects of number of nodes on average end-to-end.

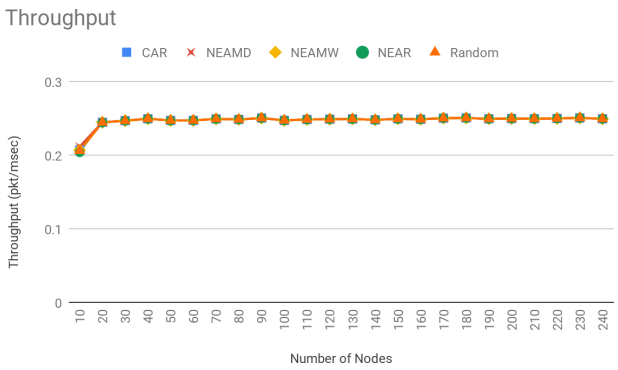

FIGURE 6. Effects of number of nodes on the network throughput.

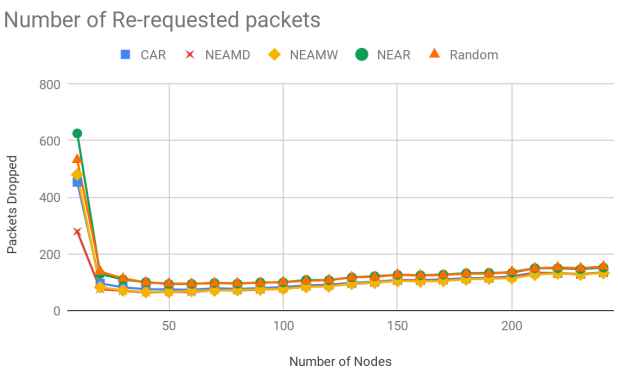

FIGURE 7. Effects of number of nodes on the number of re-requested packets in the network.

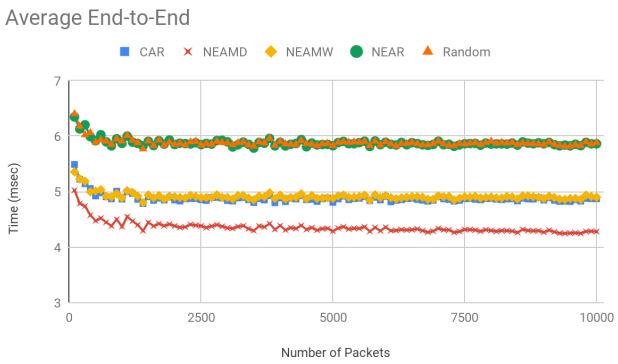

FIGURE 8. Effects of number of packets (high load) on the average end-to-end.

affecting the performance with different routing and scheduling algorithms.

2) NUMBER OF PACKETS TEST

The purpose is to measure the performance of the system when there is a high load on the network (high traffic). The arrival rate is 4 packets / milliseconds, service time 15 milliseconds (packet size is 54Kb), for 50 nodes network. FIGURE 8 shows that Random routing and NEAR are the worst in terms of Average End-to-End delay. On the other hand, NEAMW routing behaves similarly to context-awareness approach (CAR), but when the load gets higher, NEAMW behaved slightly better. NEAMD behaves the best in terms of Average End-to-End. FIGURE 9 compares the Throughput between the proposed approaches. All of the proposed approaches have similar Throughput. In FIGURE 10, a comparison between the Number of Re-requested packets was simulated, as seen, Random approach tend to behave worse than other approaches in terms of the packets dropped in the network, proportion to the increase in the number of packets. On the other hand, NEAMD was the best approach in utilizing the data flow on the network and has the least number of packets dropped, followed by NEAMW. The results also show that Network experience approach can enhance the system performance, for high load cases, in terms of,

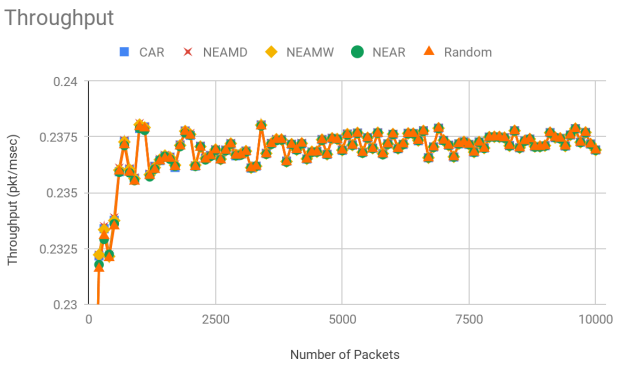

FIGURE 9. Effects of number of packets (high load) on the network throughput.

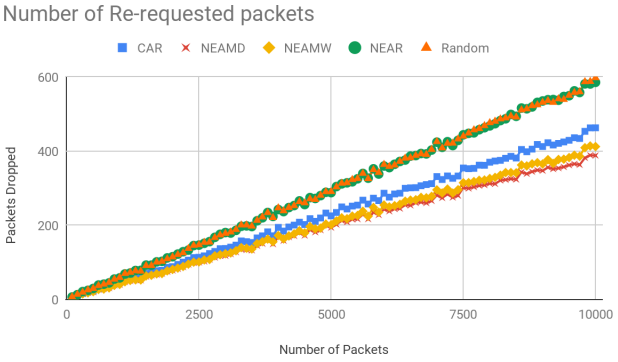

FIGURE 10. Effects of number of packets on the number of re-requested packets.

utilization of the network resources, and the end user experience (as the delay is lower), and the number of packets that the network can execute. This test is important in high load applications, and applications, where speed is important, such as hospital systems [30].

3) ARRIVAL RATE TEST

The test measures the effects of the arrival rate of a packet to the network, on the performance of the system. The test checks the high load on the system from another aspect. We fixed the transmission rate 54Mbps for IEEE 802.11a/g, the number of packets to 10000, service time of 15 milliseconds (packet size is 54Kb), for 50 nodes network. FIGURE 11 illustrates the Average End-to-End delay. The NEAMD outperform other approaches, followed by NEAMW and CAR approach. On the other hand, Random routing was the worst in terms of Average End-to-End delay, however, NEAR was better than normal Random routing. FIGURE 12 compares the Throughput of the proposed approaches. As the results show, all approaches behave similarly. FIGURE 13 shows the Number of Re-requested packets for these approaches, for the high rate the proposed approaches behave better than the low rate for packet arrival to the system. These results are depending on the packets and the order of occurrence, which in this study assumed to follow Poisson distribution. We can conclude that for fast delivery, Network experience

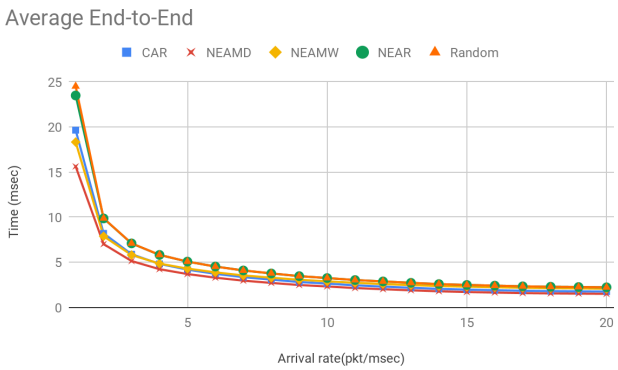

FIGURE 11. Effects of arrival time on the network average end-to-end.

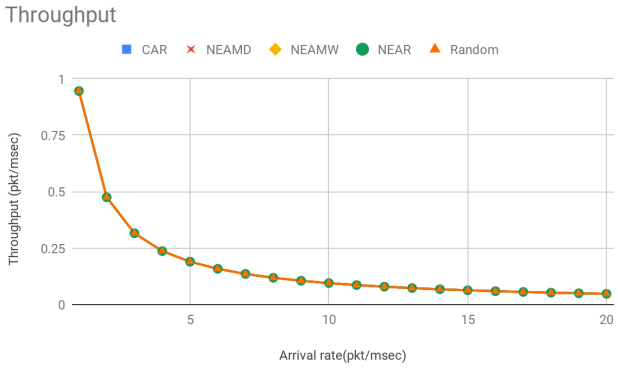

FIGURE 12. Effects of arrival time on the network throughput.

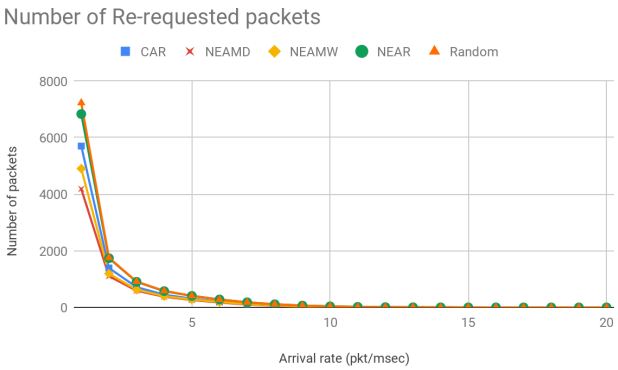

FIGURE 13. Effects of arrival rate on the number of re-requested packets.

approach outperform other proposed approaches, however, they behaved similarly in terms of Throughput.

4) DIFFERENT PACKET TYPES TEST

Tests the system performance for different packet types. This test can reflect the performance of the proposed approaches in the heterogeneous networks. We fixed the transmission rate 54Mbps for IEEE 802.11a/g, the arrival rate as 2 following Poisson distribution, packet size is proportional to the packet type (3∗ packet type) Kb, the number of packets is 100∗ (Number of different packet types). FIGURE 14 shows the Average End-to-End as seen, NEAMD behaves better than other approaches, on the other hand, Random

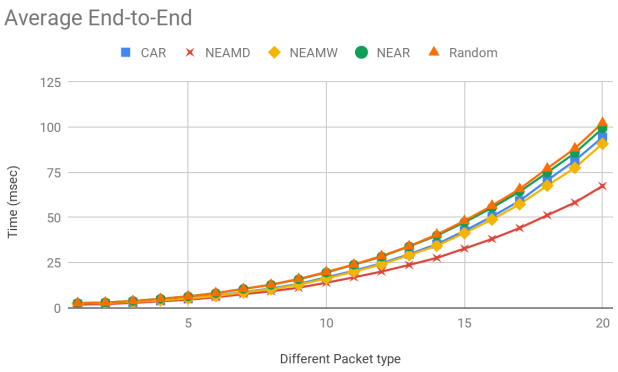

FIGURE 14. Effects of different packet types on average end-to-end.

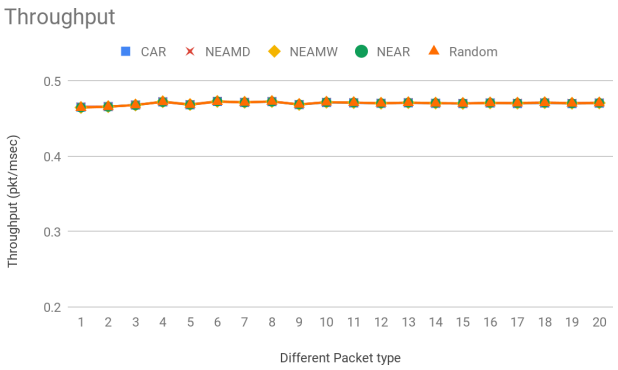

FIGURE 15. Effects of different packet types on network throughput.

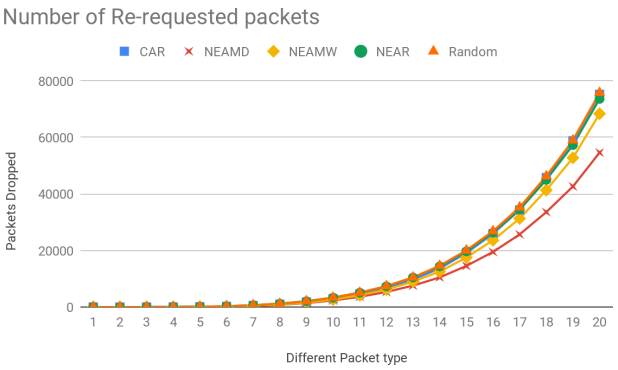

FIGURE 16. Effects of different packet types on the number of re-requested packets.

routing behaves worse, but the NEAR shows better performance, reflecting the impact of using the previous Network experience in heterogeneous systems. FIGURE 15 compares the Throughput of the proposed approaches. As illustrated, all approaches behave similarly. FIGURE 16 compares the Number of Re-requested packets, NEAMD and NEAMW are the best in utilizing the resources. On the other hand, other approaches, such as CAR approach, and Random is the worst in terms of utilizing the network, although NEAR behaves better, relative to Random routing. We can conclude that Network experience can enhance the end user experience, and reduce the Average End-to-End for the packets, and Number

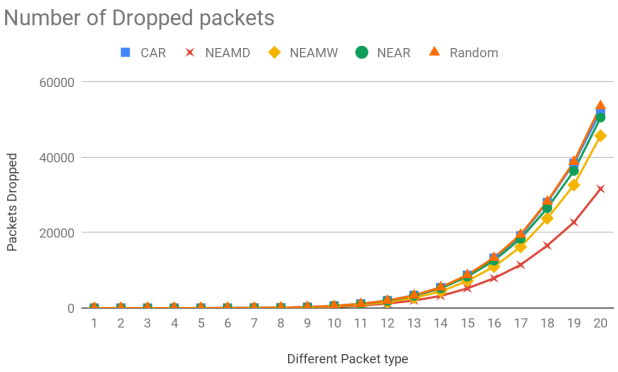

FIGURE 17. Effects of different packet types on the number of dropped packets.

of Re-requested packets, which is proportional to the number of dropped packets as seen the FIGURE 17.

VI. CONCLUSION

In this paper, the performance of CAR, Random routing, and Network experience approaches are examined. According to our results, using previous Network experience to determine the routing path for the packets, can enhance the performance of the overall network significantly. In highly loaded systems, the numerical results indicate that using Network experience approach can enhance the Average End-to-End by around 20%, compared to CAR approach and by 35% compared to Random assignment. Furthermore, it reduces the number of Re-requested packet by around 12% compared to CAR approach. In heterogeneous systems (where there are various type of packets with different service times), Network experience approach shows enhancement of nearly 30% compared to the proposed approaches for Average End-to-End. In addition to that, a further reduction of nearly 25% is observed for the Re-requested packets, and dropped packets in the system. In conclusion, in large-scale applications such as IoT, heterogeneous networks, and networks with high load, Network experience approach can enhance the network performance and utilize the network resources which affects the end user experience significantly.

REFERENCES

[1] F. Al-Turjman and M. Gunay, "CAR approach for the Internet of Things," *Can. J. Elect. Comput. Eng.*, vol. 39, no. 1, pp. 11–18, 2016.

[2] (2017). *Cisco Visual Networking Index (VNI): Global Mobile Data Traffic Forecast, 2016–2021*. [Online]. Available: http://www.cisco.com

[3] P. López, D. Fernández, A. J. Jara, and A. F. Skarmeta, "Survey of Internet of Things technologies for clinical environments," in *Proc. 27th Int. Conf. Adv. Inf. Netw. Appl. Workshops (WAINA)*, Mar. 2013, pp. 1349–1354.

[4] L. Militano, G. Araniti, M. Condoluci, I. Farris, and A. Iera, "Device-to-device communications for 5G Internet of Things," *EAI Endorsed Trans. Internet Things*, vol. 15, no. 1, pp. 1–15, 2015.

[5] M. Woo, J. Lee, and K. Park, "A reliable IoT system for personal healthcare devices," *Future Gener. Comput. Syst.*, vol. 78, pp. 626–640, Jan. 2018.

[6] H. Al-Hamadi and R. Chen, "Trust-based decision making for health IoT systems," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1408–1419, Oct. 2017.

[7] H. Zhang, J. Li, B. Wen, Y. Xun, and J. Liu, "Connecting intelligent things in smart hospitals using NB-IoT," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1550–1560, 2018.

[8] H. Malik, H. Pervaiz, M. M. Alam, Y. Le Moullec, A. Kuusik, and M. A. Imran, "Radio resource management scheme in NB-IoT systems," *IEEE Access*, vol. 6, pp. 15051–15064, 2018.

[9] C. Brewster, I. Roussaki, N. Kalatzis, K. Doolin, and K. Ellis, "IoT in agriculture: Designing a Europe-wide large-scale pilot," *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 26–33, Sep. 2017.

[10] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of IoT: Applications, challenges, and opportunities with China perspective," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 349–359, Aug. 2014.

[11] S. Alabady, F. Al-Turjman, and S. Din, "A novel security model for cooperative virtual networks in the IoT era," *Int. J. Parallel Program.*, pp. 1–16, Jul. 2018, doi: 10.1007/s10766-018-0580-z.

[12] P. Mell and T. Grance, "The NIST definition of cloud computing," NIST Special Publication, London, U.K., Tech. Rep. 800-145, 2011.

[13] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.

[14] K. Zkik, G. Orhanou, and S. El Hajji, "Secure scheme on mobile multi cloud computing based on homomorphic encryption," in *Proc. Int. Conf. Eng. MIS (ICEMIS)*, Sep. 2016, pp. 1–7.

[15] M. Al-Medhwahi and F. Hashim, "The adaptive cognitive radio sensor network: A perspective towards the feasibility," in *Proc. 1st Int. Conf. Telematics Future Gener. Netw. (TAFGEN)*, May 2015, pp. 6–11.

[16] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Commun. Mobile Comput.*, vol. 2, no. 5, pp. 483–502, 2002.

[17] C. Ceken, "An energy efficient and delay sensitive centralized MAC protocol for wireless sensor networks," *Comput. Standards Interfaces*, vol. 30, nos. 1–2, pp. 20–31, 2008.

[18] R. L. Cruz and A. V. Santhanam, "Optimal routing, link scheduling and power control in multihop wireless networks," in *Proc. 32nd Annu. Joint Conf. IEEE Comput. Commun.*, vol. 1, Mar. 2003, pp. 702–711.

[19] M. Dipobagio, "An overview on ad hoc networks," Inst. Comput. Sci., Freie Univ. Berlin, Berlin, Germany, Tech. Rep., 2008.

[20] S. M. A. Oteafy, F. M. Al-Turjman, and H. S. Hassanein, "Pruned adaptive routing in the heterogeneous Internet of Things," in *Proc. Global Commun. Conf. (GLOBECOM)*, Dec. 2012, pp. 214–219.

[21] A. Raniwala, K. Gopalan, and T. C. Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 8, no. 2, pp. 50–65, 2004.

[22] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[23] S. A. Ade and P. A. Tijare, "Performance comparison of AODV, DSDV, OLSR and DSR routing protocols in mobile ad hoc networks," *Int. J. Inf. Technol. Knowl. Manage.*, vol. 2, no. 2, pp. 545–548, 2010.

[24] S. Alanazi *et al.*, "On resilience of wireless mesh routing protocol against DoS attacks in IoT-based ambient assisted living applications," in *Proc. 17th Int. Conf. e-Health Netw., Appl. Services (HealthCom)*, Oct. 2015, pp. 205–210.

[25] M. Gunay, F. Al-Turjman, I. Kucukoglu, and Y. Simsek, "A novel architecture for data-repeaters in the future Internet," *Can. J. Elect. Comput. Eng.*, vol. 38, no. 4, pp. 300–306, 2015.

[26] Y. Xiao, "IEEE 802.11n: Enhancements for higher throughput in wireless LANs," *IEEE Wireless Commun.*, vol. 12, no. 6, pp. 82–91, Dec. 2005.

[27] F. Al-Turjman and S. Alturjman, "5G/IoT-enabled UAVs for multimedia delivery in industry-oriented applications," *Multimedia Tools Appl. J.*, pp. 1–22, Jun. 2018, doi: 10.1007/s11042-018-6288-7.

[28] M. D. Petty, "Calculating and using confidence intervals for model validation," in *Proc. Fall Simul. Interoperability Workshop*, 2012, pp. 10–14.

[29] *PISA 2003 Data Analysis Manual: SAS Users*, OCDE, Paris, France, 2005.

[30] T. Gao, D. Greenspan, M. Welsh, R. R. Juang, and A. Alm, "Vital signs monitoring and patient tracking over a wireless network," in *Proc. 27th Annu. Int. Conf. Eng. Med. Biol. Soc. (IEEE-EMBS)*, Jan. 2006, pp. 102–105.

FADI AL-TURJMAN (M'07) received the Ph.D. degree in computer science from Queen's University, Canada, in 2011. He is currently a Professor with Antalya Bilim University, Turkey. He is a leading authority in the areas of smart/cognitive, wireless and mobile networks' architectures, protocols, deployments, and performance evaluation. His record spans over 200 publications in journals, conferences, patents, books, and book chapters, in addition to numerous keynotes and plenary talks at flagship venues. He has authored or edited more than 12 published books about cognition, security, and wireless sensor networks' deployments in smart environments with Taylor & Francis, and the Springer (Top tier publishers in the area). He was a recipient of several recognitions and best papers' awards at top international conferences. He led a number of international symposia and workshops in flag-ship IEEE conferences. He is serving as the Lead Guest Editor in several journals, including the *IET Wireless Sensor Systems and Sensors*, MDPI *Sensors*, and the Elsevier *Internet of Things*.

LEONARDO MOSTARDA received the Ph.D. degree from the Computer Science Department, University of L'Aquila, in 2006. Afterwards, he cooperated with the European Space Agency on the CUSPIS FP6 Project to design and implement novel security protocols and secure geo tags for works of art authentication. To this end, he was combining traditional security mechanisms and satellite data. In 2007, he was a Research Associate with the Distributed System and Policy Group, Computing Department, Imperial College London, where he was working on the UBIVAL EPRC Project in cooperation with Cambridge, Oxford, Birmingham, and UCL for building a novel middleware to support the programming of body sensor networks. In 2010, he was a Senior Lecturer with the Distributed Systems and Networking Department, Middlesex University, where he founded the Senso LAB an innovative research laboratory for building energy efficient wireless sensor networks. He is currently an Associate Professor and the Head of the Computer Science Department, Camerino University, Italy.

ENVER EVER (M'15) received the B.Sc. degree from the Department of Computer Engineering, Eastern Mediterranean University, Cyprus, in 2002, and the M.Sc. degree in computer networks and the Ph.D. degree in performance evaluation of computer networks and communication systems from Middlesex University, in 2004 and 2008, respectively. He was with Bradford University as a Postdoctoral Research Associate. He was a Senior Lecturer with the Computer and Communications Engineering Department, Middlesex University. He is currently an Associate Professor with Middle East Technical University Northern Cyprus Campus. His current research interests include computer networks, wireless communication systems, parallel computing paradigms, wireless sensor networks, integrated circuits, and performance/reliability modeling. He serves on various programme committees and received the Exemplary Reviewer Award for his contributions as a Reviewer.

AHMED DARWISH is currently pursuing the degree in computer engineering with Middle East Technical University Northern Cyprus Campus. His current research interests include computer networks, wireless communication systems, system simulation, and the IoT applications.

NAZIHA SHEKH KHALIL is currently pursuing the degree in computer engineering with Middle East Technical University Northern Cyprus Campus. Her current research interests include computer networks, and system simulation.