Wireless Personal Communications (2019) 109:521–538
https://doi.org/10.1007/s11277-019-06577-7

# A Framework for Enabling Internet Access Using Wi-Fi Peer-to-Peer Links

**Mouaad El Alami[1] · Nabil Benamar[2] · Mohamed Younis[3] · Ahmed A. Shahin[3]**

Published online: 18 May 2019

**Abstract**
To realize the vision of ubiquitous computing and reachability, major technological advances have been achieved to make smart portable devices both highly capable and affordable. To facilitate connectivity to the Internet, these smart devices are equipped with different network interface cards (4G, Bluetooth and Wi-Fi). However, such connectivity may become seriously reduced in places with intermittent network services and/or shortage of Wi-Fi coverage. This limitation may be seriously harmful in rescue scenarios during and after a disastrous event, since blocked users may become difficult to reach. In this paper we present a novel framework to overcome such issue by implementing a hot-spot accessible through device-to-device (D2D) links. In this work, Wi-Fi direct (WFD) is used in order to establish multi-hop P2P paths to the hot-spot. WFD has the same features that carries the standard Wi-Fi in terms of range and speed without using the infrastructure. Wi-Fi direct is largely emerging as a network technology standard enabled in the majority of smartphones. By exploiting nearby devices, which have Internet access, our framework enables users to publish contents. Moreover, our model is validated by using different smartphones and through extensive simulation experiments.

**Keywords** P2P systems · WiFi direct · D2D communication

Nabil Benamar
n.benamar@est.umi.ac.ma

Mouaad El Alami
m.elalami@edu.umi.ac.ma

Mohamed Younis
younis@umbc.edu

Ahmed A. Shahin
ashin1@umbc.edu

[1] Department of Computer Sciences, Faculty of Sciences, University Moulay Ismail, Meknes, Morocco

[2] Department of Computer Sciences, School of Technology, University Moulay Ismail, Meknes, Morocco

[3] Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County, Baltimore, USA

## 1 Introduction

The emergence of smartphones as mobile computing devices and the increasing advances in their capabilities and features has changed the way we communicate and how we "sense" our neighborhood. With such devices running hundreds of mobile apps sensing the environment and accessing the Internet, cellular networks become burdened with new types of traffic that can be more resource consuming than traditional networks. The concept of Device-to-Device (D2D) was introduced to overcome this stress on the cellular infrastructure by offloading it to the end user equipment's. Indeed, Wi-Fi, NFC, Bluetooth, and Wi-Fi Direct, are enabled to even improve the smartness of these devices by using Device to Device communication and thus enriching the user experience. With D2D, users participating in an infrastructure less wireless network exchange packets directly with each other, without the need for any kind of infrastructure. Hence, various D2D based applications are used such as VANETs, where vehicles (drivers) exchange information like road conditions without the need of roadside units (RSU) [1]. WFD remains a suitable choice to implement D2D services thanks to its advantages compared to other types of traditional wireless networks (Bluetooth or ad-hoc Wi-Fi). Indeed, it is well known that both Bluetooth and Wi-Fi in ad-hoc mode have a short range [2]. However, WFD uses a longer transmission range with the same speed of an infrastructure based Wi-Fi transceiver. Hence, WFD introduces new possibilities for implementing opportunistic networks by using user's smartphone exclusively.

One of the main advantages of P2P communication of smart devices is establishing local D2D links which leads to decreasing the huge use of the communication infrastructure. This advantage becomes even more significant when the infrastructure becomes intermittent and the telecommunications services may become unusable while D2D links may be a suitable alternative. It would become greatly beneficial to outreach remote servers, such as crisis management authorities, to share the ground conditions. To realize such goals, it becomes necessary to enable Internet access using multiple D2D links.

Wifi-Direct remains one the suitable alternatives for D2D interaction, where mobile OS such as Android supports this technology since its forth version. As a reminder, Wifi-Direct makes use of P2P groups, where an election process occurs to choose the group owner (GO) which will handle all communications afterwards. The group formation process goes through a negotiation phase, which take time and may require in certain cases, some admin interaction with the involved devices to setup a connection. To avoid such a delay, one can use the service discovery in WFD and thus avoid the phase of negotiation and automatize data forwarding through different devices [3]. However, in the WFD standardized implementation, no communication across groups is possible. Our previous work [4] suggests a practical solution to overcome this limitation and thus facilitating data sharing between groups. Moreover, and to the best of our knowledge, posting data to a server in the cloud, through different devices, has not been studied by previous works. Our current approach presents a feasible solution for this technical gap.

We present in this paper a framework that permits to publish data on a server in the cloud, using WFD through a multitude of D2D links. Our protocol is a suitable alternative especially in critical cases where the infrastructure is intermittent or even absent. The optionally exchanged service discovery frames between smartphones in physical proximity makes it possible to 'discover' the enabled services by a given device. Our approach is based on the service description fields in the broadcasted DNS text record. Hence, each device receiving the data and after checking the availability of Internet access, decides if it can forward data to the cloud server, or broadcast it once again. We have validated our framework by its implementation on Android-based smart devices. Moreover, the performance in large scale setup is evaluated through extensive simulations. Our results show clearly the effectiveness of our approach in terms of delivery rate (successful Internet posts) through muti-hop D2D links, network overhead, average load on Internet gateways and also the number of dropped messages and network latency.

The remainder of paper is organized in the following fashion. In Sect. 2, we compare our protocol to existing works. Section 3 presents the WFD operations overview and the service discovery feature. Section 4 details our proposed protocol, while Sect. 5 focuses on its implementation. Section 6 presents and analyzes the performance results. Section 7 is dedicated to the simulation in large scale. Finally, Sect. 8 concludes the paper and highlights future work future work extensions.

## 2 Related Work

Various works have been recently published in the context of WFD. They are mainly interested in implementing multi-hop D2D communication as well as data dissemination. Most previous works have considered D2D communication without interest to the possible access to Internet through D2D links. Our work presents a solution where involved devices use the infrastructure when it is at hand.

Authors in [5] presented the benefits of WFD in P2P systems with mobile devices and its advantages in various applications such as chat or traffic data dissemination. Authors in [6] have used the service discovery in WFD for sharing time-sensitive data among devices. The service discovery frames are exchanged to send alerts about emerging events, where the alert occurs before the GO negotiation. Thus, devices can decide to join a group that tracks an event or not. Authors in [7], used Wi-Fi direct advantages to enable collaboration in Peer-to-Peer context.

Various previous works have raised the issues related to the creation and maintenance of a group. Authors in [8] and [9] studied the latency in the creation of a group using experiments with different mobile devices. Authors in [10] suggested another alternative where software access points create a mesh network and cryptographic technique is used select the GO. In [11] the group owner continuously tracks the connected and the disconnected devices and the GO informs each group member about the identity and availability of other group members for a smooth interaction between devices within the group.

Furthermore, Duan et al. [12] suggested routing data between devices belonging to two different groups by connecting the two groups owners. The second GO is seen as a legacy client. This approach seems efficient; however, its implementation shows some limitations in Android based devices. In WFD, a virtual interface is configured through a DHCP server in the range (192.168.94.x/24). Hence, all device belonging to the first GO gets an IP address from this very same range. Consequently, an overlap of IP addresses will occur if the DHCP pool is not modified by the other group owners.

Authors in [4] proposed a mechanism to overcome this limitation by enabling the negotiation of DHCP addresses for nearby groups. In [13], the authors implement various protocols based on Wi-Fi Direct and demonstrated the limitation of the service name to 24 characters in the Android OS.

The study in [14] makes use of social network messages when there is no access to Internet. The messages are distributed through various smartphones to end up at the cloud.

Our approach in this paper is similar to [14] but uses Wi-Fi Direct. The Wi-Fi Ad hoc mode in Android is implemented in the kernel level, which makes it inaccessible at the application level. In Ad hoc mode, unlike the network range is limited [2] which makes it unsuitable in cases where the infrastructure becomes not available.

Wi-Fi is extensively deployed for wireless communication in the Internet of Things. Authors in [15] designed a low cost Wi-Fi interface embedded IoT-oriented devices and assessed the performances of a double Wi-Fi interface Mesh network based on the B.A.T.M.A.N approach. They configured two Wi-Fi interfaces one for the mesh backbone and the other one for the external communication. Nevertheless, their solution depends on the platform used and thus can not be ported to stock mobile devices. Furthermore, with WFD concurrent operation, it is possible to achieve similar results without the need for a double Wi-Fi interface. Indeed, the WFD interface can be used for the mesh backbone while the Wi-Fi interface can still manage external communication. By using WFD, the deployment of mesh networks can be realized without depending on specific platform requirements. Furthermore, authors in [16] modeled the energy consumption of the WiFi direct protocol, starting from device discovery to actual data transmissions for intra group D2D communications. They found that for smaller data sizes and device discovery times, the energy consumption is balanced in both device discovery and data transmission phases. Authors in [17] designed and implemented bidirectional, multi-group communication in non-rooted Android devices supporting the Wi-Fi Direct protocol. However, this study did not tackle the large scale case where multi devices are involved. More recently, authors in [18] proposed a new information diffusion method exploiting WFD and delay-tolerant networking (DTN) to better send broadcast message. However this work has not been implemented in real devices.

## 3 Overview of Service Discovery in Wi-Fi Direct

In Wi-Fi direct the device checks for the supported services by accessible devices before attempting any WFD connection. In other words, SD frames are sent in the network before the election of the GO. Figure 1 illustrates this process for the case of two devices “A” and “B”. It is worth noticing that SD in Wi-Fi direct is based on two main protocols, namely the DNS (multicast) [19], which does not need for a DNS server and the DNS-SD [20], which adds the supported services by each device to the DNS record as illustrated in Fig. 2.

The SRV record maps the host to its supported services. The structure of a qualified local domain name starts with the name of the device, and then the name of the service instance, then the transport protocol followed by the local domain name.

In the next section, we present our protocol which exploits the service discovery features to enable D2D data transfer to a node that acts as a gateway to the Internet.

## 4 Wifi-Direct-Based Hotspot Protocol

### 4.1 Overview of the Approach and the Design

The primary goal of this paper is to present a protocol that make it possible for users to send data to Internet without necessarily an Internet access enabled in their smartphones. Our proposal is a capable WFD-based data forwarding protocol permitting to push data to

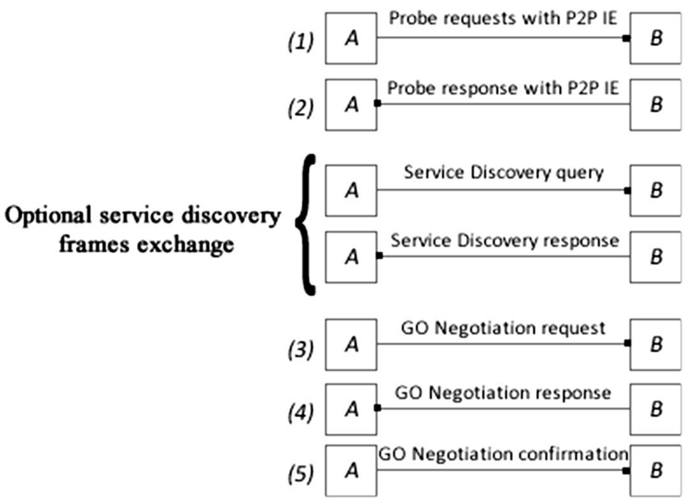

(1)
A
Probe requests with P2P IE
B
(2)
A
Probe response with P2P IE
B
Optional service discovery frames exchange
A
Service Discovery query
B
A
Service Discovery response
B
(3)
A
GO Negotiation request
B
(4)
A
GO Negotiation response
B
(5)
A
GO Negotiation confirmation
B

**Fig. 1** WFD group formation frames exchange

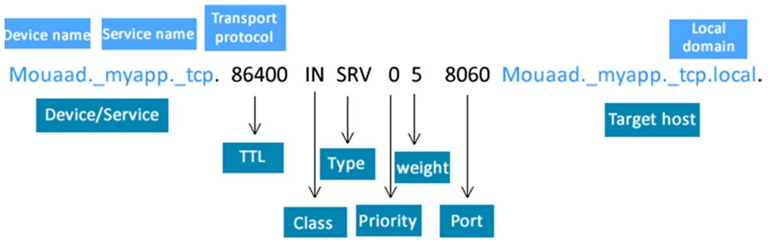


**Fig. 2** An example DNS-SD Service record

the Internet using different D2D links. Packets are routed from one smartphone to another until they reach an area where a gateway (hotspot) to the Internet is found. The Wi-Fi direct based hotspot we realized is in full respect of the WFD specification.

Our approach makes full use of the SD process in Wi-Fi direct. Fundamentally, the SD is an optional parameter in Wi-Fi direct, which can be activated before the election of the GO, which makes it possible to develop a protocol running without human interaction. Figure 3, illustrates the case where two P2P devices communicate through a single hop. It is the case of a smartphone that connects to the Internet directly and then reach the server by sending the data to a nearby device using Wi-Fi direct. This device has access to the communication infrastructure. The implementation of our protocol relies on two main steps; (1) managing SD requests and responses, and (2) managing the connection to the Internet server. The connection can be established through existing application layer protocols such as HTTP.

## 4.2 Protocol Design

To better handle any interruption of the access to Internet, our protocol operates in two phases, where the devices in the first phase are used as relays. In the second phase, the

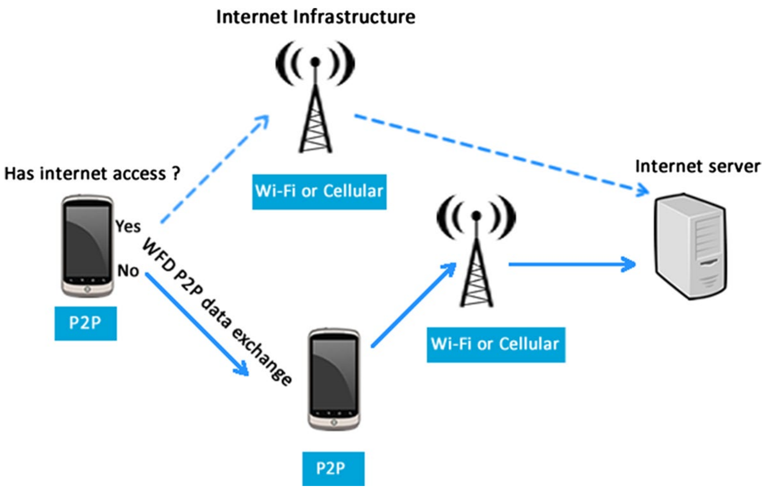
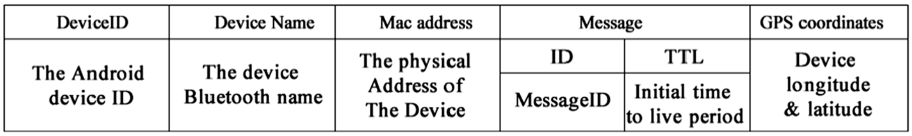


**Fig. 3** A topology example of P2P data forwarding to the Internet



| DeviceID | Device Name | Mac address | Message | | GPS coordinates |
|---|---|---|---|---|---|
| The Android device ID | The device Bluetooth name | The physical Address of The Device | ID | TTL | Device longitude & latitude |
| | | | MessageID | Initial time to live period | |

**Fig. 4** TXT record structure

protocol steps establish the connection to the Internet host. The protocol registers first the name of the service instance that is stored in the SRV record and that identifies each device supporting the same service, by verifying the instance name in common. Substantial information are then collected by the protocol such as the MAC address of the device as well as its coordinates and its ID. Then, the protocol creates a TXT record to store all the collected information. Figure 4 shows the TXT record data structure.

The TTL field in the TXT record represents the initial time to live value, every time the message is rebroadcasted. In a search-and-rescue application, the TTL should be large to increase the rate of successful Internet posts sent by the devices of the trapped users. In the case of high mobility settings, a large TTL value enables more attempts to overcome unstable links. However, increasing the TTL boosts the overhead in the network. To reduce the potentially high overhead, the TTL is decreased by the relaying nodes at a varying rate based on how many copies of the same message are received. Such approach makes the forwarding process more adaptive to the network density.

After defining the TXT record and the instance name, a service discovery request is initiated. If any device discovers an available service from a nearby device, it checks for the instance name in common and then starts the data extraction process from the TXT record. This process checks first the Internet connectivity to connect with the host. In the case of no Internet access, the TXT record information is stored in a local queue, and a timer of waits for *m* seconds is lunched before rechecking for Internet connectivity. The parameter *m* depends on the general stability of the Internet connectivity and may vary from device

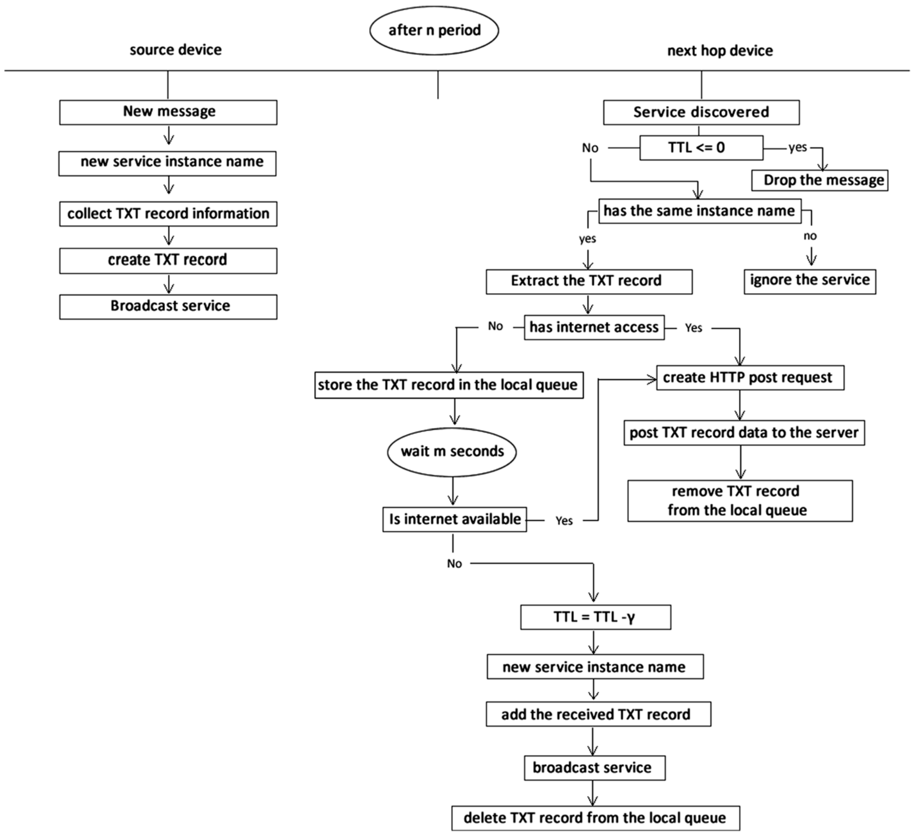


**Fig. 5** Flowchart description of our proposed protocol

to device. In an area with degraded cellular coverage a small value of *m* is advisable so that our protocol converges quickly by reaching a node with Internet access. However, if Internet connectivity is intermittent due to mobility or radio interference, a larger value for *m* may be better for a faster convergence.

When m seconds pass, the second check is made. If the second check fails, the TTL entry in the message is decremented by γ to prevent the flooding of the network with unnecessary messages. The default value of γ is one; yet is can be greater to limit the overhead when the original setting of TTL is large and the network density is high. Our protocol sets γ to the number of duplicate packets a device receives for the same message. Each time there is an Internet access, the protocol the needed information will be extracted from the TXT record and formats an HTTP post request as illustrated in Fig. 5. The variable "*n*" is the time needed for a relay to discover the TXT record. The setting on "*n*" depends on the capabilities of the devices and the quality of the peer-to-peer links.

Our protocol is based on two modules in the server side. The first module manages and stores the HTTP posts in a structured local text file. The server picks up duplicated massages by discarding messages without a unique identifier. The second module extracts the stored messages each time they are requested by a device, format and sent them to the client as a plain text. It is worth reminding that in this work, we are mainly interested in

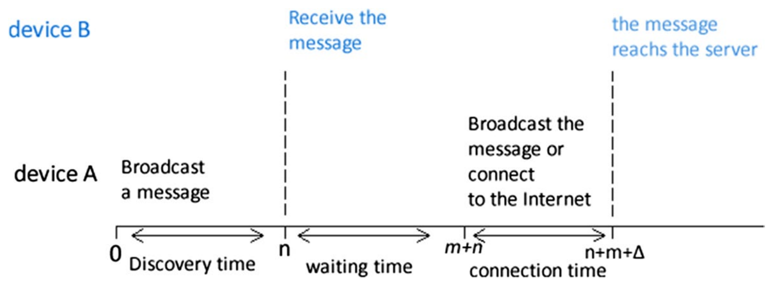


**Fig. 6** Illustration of the of the proposed protocol with different time sequences

sending data to the cloud in scenarios where Internet access may become unavailable after a disaster.

## 5 Protocol Implementation

Our protocol has been implemented on the Open Source Operating System Android, which supports WFD and service discovery. We implemented our protocol as an application allowing users to post an embedded message using the SD frames. The nearby devices discover the service through the same application, and then message is extracted. The device then checks the availability of the Internet access. In case the smartphone is able to access the Internet, it becomes responsible to post to the server the extracted message. On the other hand, the message is encapsulated once more in a service frame and sent to the neighbors as presented in Fig. 5.

In our application/test, we used the GPS coordinates of the node (source) as the event which the server marks the source node position using Google maps. Our testbed is composed of three Android devices (Samsung galaxy s4, galaxy Trend plus and a galaxy grand prime). Two of the devices do not have Internet access, while the third device has. One of the unconnected devices is the source of the message, and the next device should discover the service but fails to connect to the Internet.

In our scenario, we used the GPS coordinates of the node (source). The server marks the source node position using Google maps. Our testbed is composed of three different Android devices. Two of them do not have Internet access, while the third device has. One of the unconnected devices is the source of the message, and the next device should discover the service but fails to connect to the Internet.

In our scenario, the server retrieves the position of the source node using the GPS coordinates and Google maps. Our testbed is composed of three different Android based devices, where only one smartphone has access to the Internet.

The first device without Internet access send a message, while the second device should be able to discover the service but does not succeed to access to the Internet. Hence the message is forwarded to the third device which has access to the Internet.

This scenario permits us to assess the behavior of the devices implementing the protocol. The value of $n$ is related to the capabilities of the individual devices and the quality of the peer-to-peer links among them. Moreover, $n$ is the actual time for the second device to discover the service. On the server side we used centos 7 OS with Apache 2.4.

Figure 6 illustrates all the time sequences of the protocol.

**Table 1** Summary of the device specification

| Device | Android version | CPU | RAM | Release date |
|---|---|---|---|---|
| Galaxy S4 (GS4) | 5.0.1 (lollipop) | Quad core 1.9 GHZ | 2 GB | Apr 2013 |
| Galaxy grand prime (GGP) | 5.1.1 (lollipop) | Quad core 1.2 GHZ | 1 GB | Oct 2014 |
| Galaxy trend plus (GTP) | 4.2.2 (Jellybeans) | Dual core 1.2 GHZ | 0.75 GB | Dec 2013 |

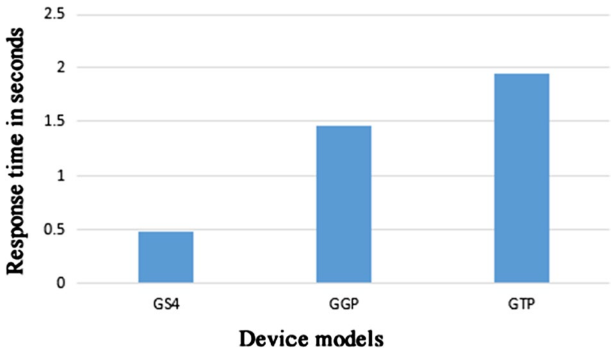


**Fig. 7** Average response time for the different Android-based smartphones

## 6 Performance Analysis

The performance assessment permits us to determine the speed of data transmission in our protocol. The required time for a device to receive the SD TXT records sent by another device, remains a critical factor to determine the time needed to forward the records containing the data to the final destination. In Table 1 we show the characteristics and features of the different devices we used in our experiments. We have fixed the indoor distance between devices to only 50 m. The measurement is performed by using a counter running the system time function as a new process to ensure its independence from the protocol execution. Figure 7, shows the average time needed by the different devices before receiving the announcement of the service, while Fig. 8 illustrates the implementation of our protocol with three Android devices.

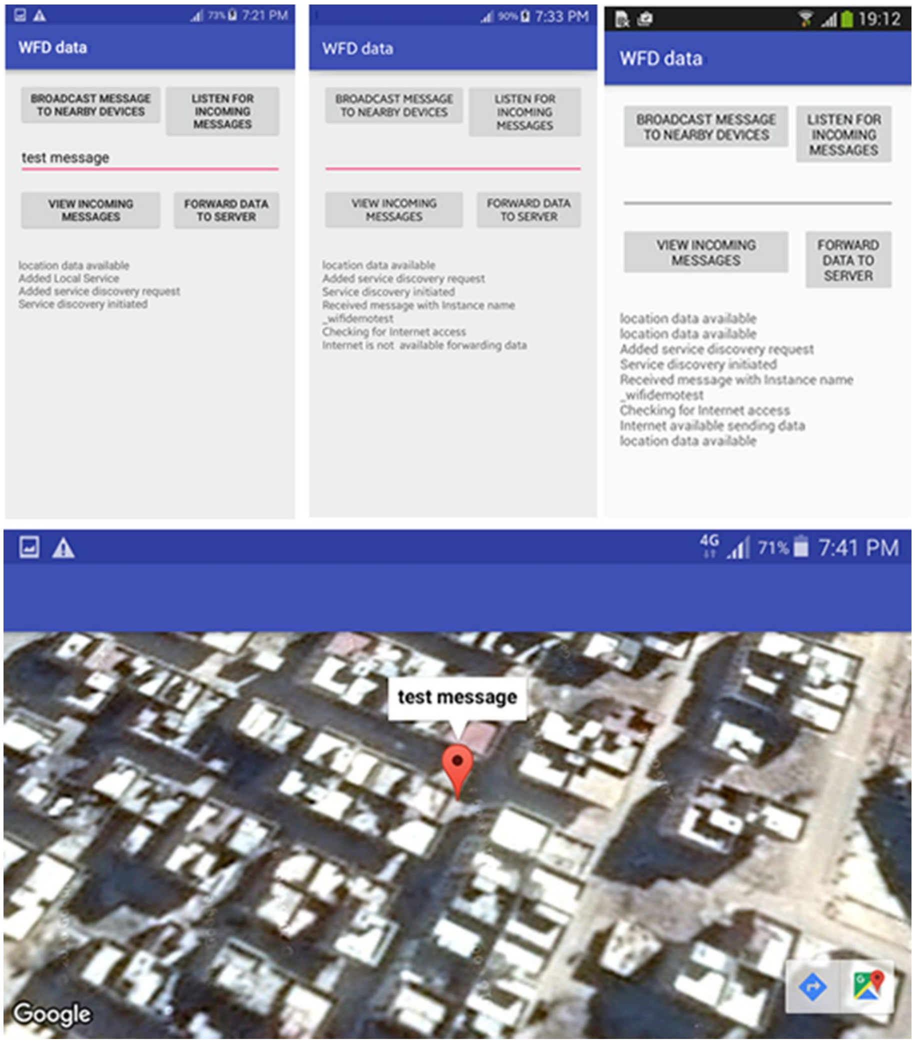
From Fig. 7, we can clearly notice that the time response depends on the model of each device. Indeed, a device with a more recent hardware performs better than the older ones. Also, the current discovery time depends on the propagation time $T_{pr}$ and the transmission time $T_{tr}$.

$$T_{pr} = \frac{Distance}{wave\ speed} \tag{1}$$

$$T_{tr} = \frac{Packet\ seize}{bit\ rate} \tag{2}$$

In general, the radio wave propagation speed is the speed of light: $3.10^8$ m/s. The bit rate of Wi-Fi is 54 Mb/s. By combining all parameters, the actual discovery time, *n,* becomes as follows:

$$n = T_{pr} + T_{tr} + T_c \tag{3}$$



test message

location data available
Added Local Service
Added service discovery request
Service discovery initiated

location data available
Added service discovery request
Service discovery initiated
Received message with Instance name
_wifidemotest
Checking for Internet access
Internet is not available forwarding data

location data available
location data available
Added service discovery request
Service discovery initiated
Received message with Instance name
_wifidemotest
Checking for Internet access
Internet available sending data
location data available

test message

**Fig. 8** Screenshots of devices running the test application

where $T_c$ is the computation time (the time spent by the application to provide the received information).

This parameter depends on the model of the device, which differs from other models in terms of CPU capabilities. Hence, newer models perform better and consequently yield lower **n** values. The variable, *m,* represents the waiting time of a message in the local queue when the receiving device does not have Internet access. The parameter *m* depends closely on the number of nearby devices and thus decreases when the number of neighbours involved increases, which can be explained by the higher probability that the next hop will forward the message to the Internet. Hence, the message will be discarded from the local queue so that it dos not affect the overall delivery time of the protocol.

Based on Fig. 6, the overall time $T_t$ needed for a message to reach the cloud can be expressed as follows:

$$T_t = H_n \times (n + m) + \Delta \tag{4}$$

where $H_n$ represents the hop count, and $\Delta$ is the processing time at the server. We apply the formula (3) to deduce the computation time by using the values in the graph. Using the GS4 device and a packet of 1000 bytes, the computation time is defined as:

$$\begin{aligned} T_c &= n - T_{pr} - T_{tr} = \mathrm{n} - \frac{Distance}{Wavespeed} - \frac{Packetsize}{Bitrate} \\ &= 0.49 - \frac{50}{3 \times 10^8} - \frac{8 \times 1000}{54 \times 10^6} = 0.48\,\mathrm{s} \end{aligned}$$

Thus, $T_c$ represents 98% of $n$, and the values of $T_{pr}$ and $T_{tr}$ are negligible. The overall delivery time of the protocol can now be calculated based on formula (4).

For $m = 5$, the time to make an HTTP post to the server, is usually around 50 ms. Hence, we can calculate the overall delivery time of the protocol using the expression of $T_t$ with a single hop scenario where the second device has internet access as follows:

$$\begin{aligned} T_t &= H_n \times (n + m) + \Delta = n + m + \Delta \\ &= 0.49 + 5 + 0.05 = 5.54\,\mathrm{s}. \end{aligned}$$

By reducing the value of $m$ and using devices with more CPU power, the total delivery time decreases significantly. It is envisioned that our framework will be integrated with a routing function to optimize $m$, if the P2P connectivity to the hotspot is to be sustained for a long time. The next section shows the performances of our proposed protocol in large scale.

## 7 Large Scale Simulation

We have used simulation to capture the performance of our protocol [21] in larger scale where many devices are involved. This section discusses the simulation environment and reports the obtained results.

### 7.1 Simulation Environment

To study the performance in large networks, we have used WiDiSi [22], which is a peer-to-peer simulator that fully supports Wi-Fi direct. WiDiSi is built on top of peerSim [23], a well-known java based peer-to-peer simulation and prototyping tool. WiDiSi implements Wi-Fi direct in the same way that it is implemented on Android devices; this means that any simulation code written in WiDiSi can be easily ported to real Android devices with little modification to be made. WiDiSi implements service discovery using the Apple bonjour DNS-based service discovery. WiDiSi also supports node mobility. WiDiSi is a cycle-based simulator where the simulation time is divided into 100 ms long cycles. We would like to note that the simulation focus on the D2D part of the protocol. The management of duplicate messages at the HTTP server is handled by the server itself which stores only the messages that have a unique identifier. Throughout the simulation, a post is considered successful if it reaches a connected node directly or through multi-hop communication.

## 7.2 Simulation Setup

The network scenario is based on a 500 m × 500 m area. The number of nodes ranges from 20 to 100. Nodes are divided into three categories. The first represents the "connected nodes" which act as Internet gateways. The second category represents the "source node", which is the one that initially broadcasts the request message to its nearby devices. The third category is the "relay nodes", which are not connected to the Internet and happen to be at physical proximity from the source node.

In order to determine the effectiveness of our protocol in delivering messages to the Internet, we used two simulation scenarios. In the first scenario, the nodes acting as gateways are predetermined before starting the simulation, the density of gateway nodes is varied between 5 and 10% of the node population. We used a uniform random distribution to select the gateway nodes in this scenario. Meanwhile in the second scenario each node in the network including the source node has a probability of having Internet access. We study two settings for such probability, 5% and 10%. The second scenario mimics the case when intermittent connectivity to the Internet is experienced. In both scenarios all nodes are using the waypoint mobility model. The TTL value is set to 5 in all simulation scenarios.

Four performance metrics are tracked: (1) the number of successful Internet posts, (2) the number of dropped messages due to the TTL reaching 0, (3) the average load on Internet gateways which represents the average number of massages posted to the Internet by each gateway node, and (4) the overhead which we defined as ratio between the messages that successfully reached the gateway nodes and the total generated messages by source nodes.

$$Overhead = \frac{Generated\,messages}{Successful\,messgaes} \tag{5}$$

## 7.3 Results and Discussion

Recall that the objective of the simulation experiments is to validate the performance of large setups; therefore, the number of nodes in the simulation area has been varied and all the four performance metrics are tracked. The reported results reflect the average over 30 runs and stays within 10% of the sample mean when subjected to 95% of confidence interval.

Figure 9a shows the average number of successful Internet posts for various node count using two different percentages of gateway nodes. For relatively high node densities, the impact of the size of gateway population becomes clear. This is because nodes are mobile during the entire simulation time, and thus there is a greater chance of exchanging information with more than one device at higher densities. Figure 9b shows the results of the second scenario, which is based on the nodes having a probability of Internet access, i.e., intermittent connectivity to the Internet. Each time a node receives a message, it checks its ability to connect to the Internet by calling a *checkInternet* function. The function returns true or false. Each time a node fails to connect after receiving a message, it waits *m* seconds before checking one last time. The main difference between results of the first and the second scenario, i.e., Fig. 9a, b, is the fact that the *m* and TTL values have an impact on the number of successful Internet posts because all nodes in the network have an equal

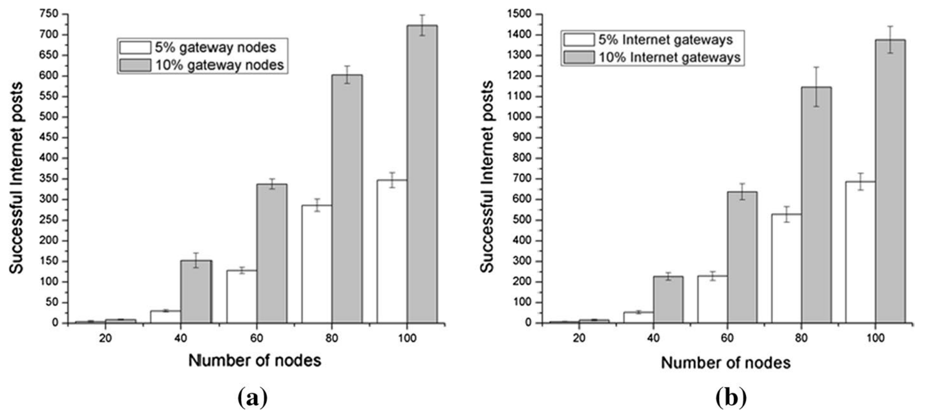


**Fig. 9** Successful Internet posts for **a** the scenario with predetermined gateways, and for **b** the case with intermittent Internet access

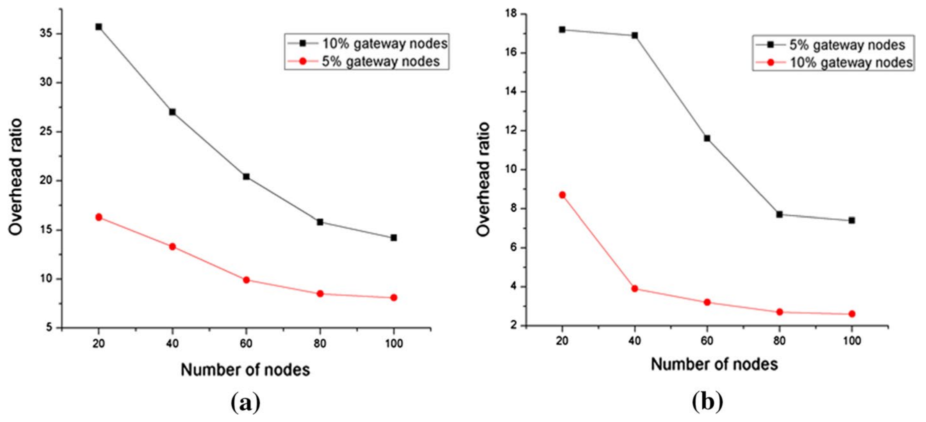


**Fig. 10** The overhead ratio for both scenarios: **a** predetermined gateways, and **b** time varying Internet connectivity

probability of changing their state from unconnected to connected, as long as the TTL value has not reached 0. Whereas in the first scenario the TTL value has only an effect on the number of messages circulating in the network, but no effect on the number of Internet posts.

Figure 10a shows the results of overhead measurement using the first scenario in which we designated nodes having the Internet access beforehand. The overhead is defined as a ratio between the successful Internet posts and the total generated messages in the network. Since the protocol relies only on physical proximity and no connection establishment is required, devices exchange the entire TXT record once they are in range of each other. Thus, the overhead ratio will decrease when the network has more connected devices since fewer broadcasts are needed and the number of dropped messages will decrease. Figure 10b reports the overhead for the second simulation scenario where devices have intermittent Internet access. As seen in the figure, the overhead has decreased for 10% access probability, i.e., 10% gateway nodes, compared to Fig. 10a. Such overhead decline

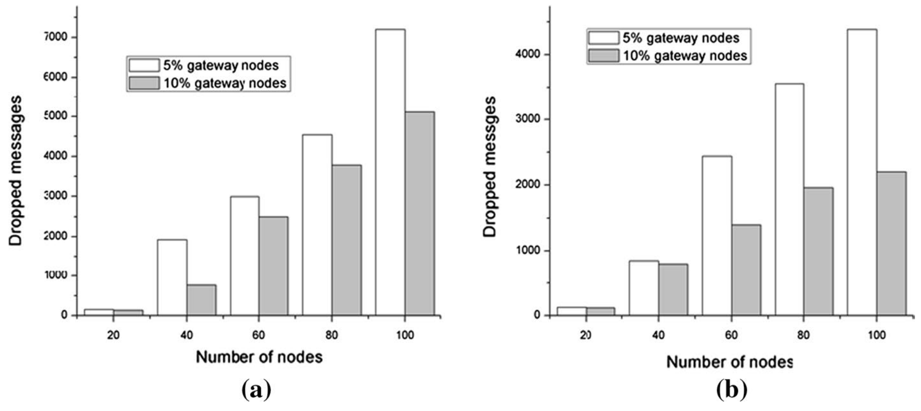


**Fig. 11** The number of dropped messages for: **a** the scenario of predetermined gateway nodes, and **b** the scenario with varying gateway node density

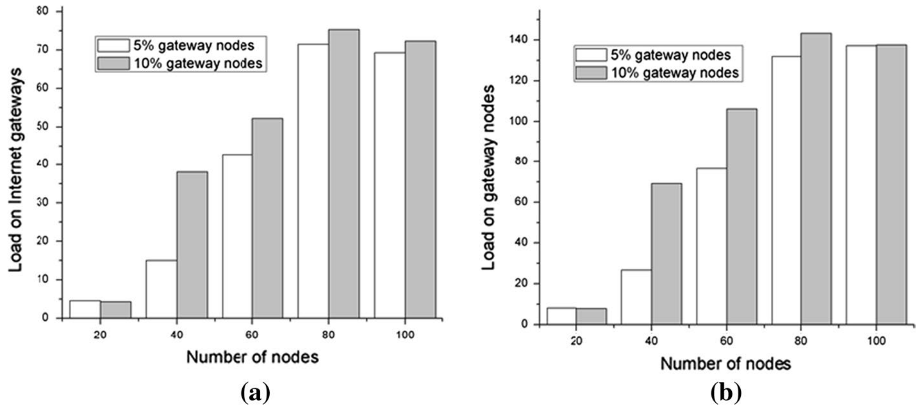


**Fig. 12** The load on Internet gateways for the two considered network scenarios (**a**) preselected gateway nodes, and **b** the intermittent node's connectivity to the Internet

is because the number of dropped messages has decreased as a result of a higher probability of Internet access. It can also be noticed that the overhead ratio decreased substantially in comparison to Fig. 10a, which is due to the better distribution of messages through the network area given the change of gateway distribution overtime. It is worth noting that the second scenario in our simulation is more practical where connectivity could be intermittent in real life.

Figure 11a shows the number of dropped messages per node based on the first scenario. A message is dropped once its TTL values reaches zero. As expected, the number of dropped messages decreases when there are more Internet gateways in the network. On the other hand, the number of dropped messages increases for higher network densities, particularly due to the increase of duplicate messages reaching Internet gateways. Figure 11b shows the results for the second scenario. In Fig. 11b the overall number of dropped messages decreases at both the 5% and 10% connectivity rates. This is due to the increased

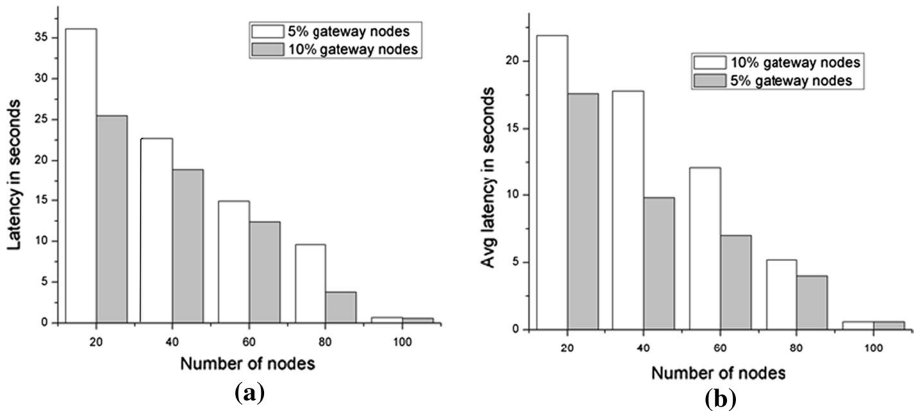


Fig. 13 Internet access latency when the gateways (**a**) are predetermined, and **b** vary overtime

probability of state change from "unconnected" to "connected" which in turn results in fewer rebroadcasts due to higher number of gateway nodes.

Figure 12 shows the average load on Internet gateways in the fixed and probabilistic scenarios, respectively. The load is gauged by the average number of posts that a gateway node makes. The scenario for intermittent connectivity shows better message distribution compared to the fixed scenario and this is due to a better distribution of the connected nodes inside the network. On the other hand, Fig. 13 shows the Internet access latency at different densities for the two considered scenarios. The latency is defined as the average time taken for a request to reach an Internet gateway. As expected at low network densities

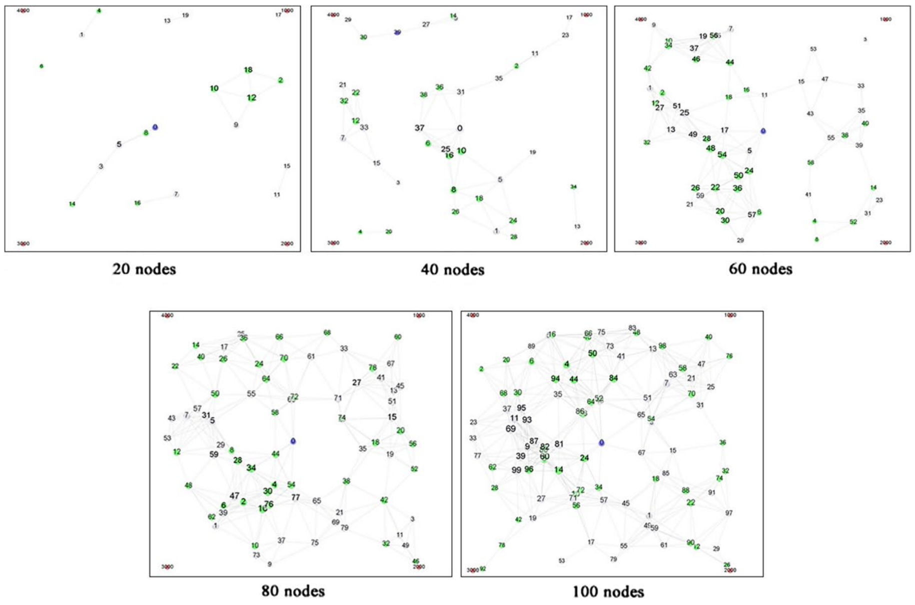


Fig. 14 Illustration of network density for various number of nodes

the time is considerably high. Since nodes exchange data without establishing a prior connection, proximity among nodes has a notable impact on the latency. With higher network densities nodes can disseminate data more quickly because there is more relays that are in the range at various positions in the area. Again the scenario with intermittent connectivity to the Internet yields better results for the same reasons stated earlier when discussing the other metrics.

To better illustrate the effect of node density, Fig. 14 shows sample topologies from our simulation with different node densities. As indicated by Fig. 14, at lower network densities mainly at 20 and 40 node counts, the network becomes clustered with no or limited inter-cluster connectivity. The clustering affects all the measured metrics as it appears on all the graphs where the lowest values were recorded at the 20 and 40 network densities.

## 8 Conclusion and Future Work

In this paper we have presented a framework permitting to smart devices to send data to Internet through D2D links and Wi-Fi direct, in the case where the infrastructure becomes unavailable. Such critical scenario occurs in case of naturel disasters, where other alternatives should be used to reach individuals in stricken zones. Our P2P framework allows message forwarding to the Internet by taking advantage of nearby devices. It exploits the service discovery frames exchange of WFD to spread packets towards devices that have access to the Internet. Our framework has been validated by experimental results as well as extensive simulations for large scale case. The results show that the protocol exhibits good performance even with very low percentage of nodes that have access to the Internet.

## References

1. Fogue, M., et al. (2013). An adaptive system based on roadmap profiling to enhance warning message dissemination in VANETs. *IEEE/ACM Transactions on Networking, 21*(3), 883–895.
2. Wi-Fi Alliance. P2P Technical Group, Wi-Fi Peer-to-Peer (P2P) Technical Specification v1.2, December 2011.
3. Shahin, A. A., & Younis, M. Efficient multi-group formation and communication protocol for Wi-Fi Direct. In *Proceedings of the 40th annual ieee conference on local computer networks* (*LCN 2015*), Clearwater Beach, FL, October 2015.
4. Shahin, A. A., & Younis, M. IP subnet negotiation in Wi-Fi direct for seamless multi-group communications. In *Proceedings of the 9th IEEE international workshop on selected topics in mobile and wireless computing (STWiMob 2016)*, New York, NY, Oct 2016.
5. Motta, R., & Pasquale, J. Wireless P2P: Problem or opportunity. In *Proceedings of the 2nd international conference on advances in P2P systems, Florence, Italy*, Oct 2010.
6. Shahin, A. A., & Younis, M. Alert dissemination protocol using service discovery in Wi-Fi direct. In *Proceedings of the IEEE international conference on communications (ICC 2015), London, UK,* June 2015.
7. Park, J.-E., Park, J., & Lee, M.-.DirectSpace: A collaborative framework for supporting group workspaces over Wi-Fi direct. In *Proceedings of the 4th international conference on mobile, ubiquitous, and intelligent computing (MUSIC 2013), Gwangju, Korea*, Sept 2013.
8. Conti, M., Delmastro, F., Minutiello, G., & Paris, R. Experimenting opportunistic networks with WiFi Direct. In *Proceedings of the 6th wireless days conference, Conference, Valencia, Spain*, Nov 2013.

9. Camps-Mur, D., Garcia-Saavedra, A., & Serrano, P. (2013). Device to device communications with wifi direct: Overview and experimentation. *IEEE Wireless Communications Magazine, 20*(3), 96–104.
10. Wong, P., Varikota, V., Nguyen, D., & Abukmail, A. Automatic android- based wireless mesh networks. *Informatica (Slovenia)*, Vol. 38, No. 4, 2014.
11. Shahin, A. A., & Younis, M. A framework for P2P networking of smart devices using Wi-Fi direct. In *Proceedings of the the 25th IEEE international symposium on personal, indoor and mobile radio communications (PIMRC 2014)*, Washington, DC, Sept 2014.
12. Duan, Y. et al. Wi-Fi direct multi-group data dissemination for public safety. In *Proceedings of the world telecommunications congress (WTC 2014)*, Berlin, Germany, June 2014.
13. Marinho, R. P., Menegato, U. B., Augusto, R., & Oliveira, R. De. Mobile devices routing using Wi-Fi direct technology. In *Proceedings of the 11th advanced international conference on telecommunications (AICT 2015)*, *Brussels, Belgium,* June 2015.
14. Teranishi, Y., & Shimojo, S. MONAC: SNS message dissemination over smartphone-based DTN and cloud. In *Proceedings IEEE international conference on peer-to-peer computing (P2P 2011), Tokyo, Japan*, Aug. 2011.
15. Davoli, L., Cilfone, A., Belli, L., & Ferrari, G. Design and experimental performance analysis of a B.A.T.M.A.N.-based double Wi-Fi interface mesh network, Future Generation Computer Systems (2018), https://doi.org/10.1016/j.future.2018.02.015.
16. Usman, M., Asghar, M. R., Ansari, I. S., Qaraqe M. & Granelli, F. An energy consumption model for WiFi direct based D2D communications. In *2018 IEEE global communications conference (GLOBECOM)*, Abu Dhabi, United Arab Emirates, 2018, pp. 1–6.
17. Casetti, C. E., Chiasserini, C. F., Duan, Y., Giaccone P., & Perez Manriquez, A. Data connectivity and smart group formation in Wi-Fi direct multi-group networks. In *IEEE transactions on network and service management*, vol. 15, no. 1, pp. 245–259, March 2018.
18. Furutani, T., Kawamoto, Y., Nishiyama, H., & Kato, N. Proposal and performance evaluation of information diffusion technique with novel virtual-cell-based Wi-Fi direct. In *IEEE Transactions on Emerging Topics in Computing*. https://doi.org/10.1109/tetc.2019.2891713.
19. RFC 6762 - Multicast DNS. (n.d.). https://tools.ietf.org/html/rfc6762.
20. RFC 6763 - DNS-Based Service Discovery. (n.d.). https://tools.ietf.org/html/rfc6763.
21. El Alami, M., Benamar, N., Younis, M. & Shahin, A. A. A framework for hotspot support using Wi-Fi direct based device-to-device links. In *2017 13th international wireless communications and mobile computing conference (IWCMC), Valencia,* 2017, pp. 552–557.
22. Baresi, L., Derakhshan, N. & Guinea, S. WiDiSi: A Wi-Fi direct simulator. In *2016 IEEE wireless communications and networking conference, Doha,* 2016, pp. 1–7.
23. Montresor, A. & Jelasity, M. PeerSim: A scalable P2P simulator. In *2009 IEEE ninth international conference on peer-to-peer computing*, Seattle, WA, 2009, pp. 99–100.

**Mouaad El Alami** received the M.S. degree in information systems security from the School of Applied Sciences, University of Ibn Tofail, Morocco, in 2016. In 2018 Mouaad was pursuing the Ph.D. degree in the School of Science, University of Moulay Ismail, Morocco.

**Nabil Benamar** received the B.E., M.Sc., and Ph.D. degrees from the University of Moulay Ismail in 1998, 2001, and 2004, respectively. He is currently a Professor of computer sciences with the School of Technology, Department of Computer Engineering, Moulay Ismail University, Morocco. His main research topics are IPv6, vehicular networks, ITS, IoT, and Service Function Chaining. He has authored several journal papers and IETF Internet Drafts. He is a reviewer for Computer Communications (Elsevier), JKSUCS (Elsevier), Adhoc (Elsevier), IJWIN (Springer), AJSE (Springer),IEEE ITS, IEEE ACCESS and IEEE NEtwork. He is also a TPC Member in different IEEE conferences (WCNC, Globecom, ICC, and PIMRC). He is an IPv6 Expert (he.net certified) and a Consultant with many international organisms, such as Academic Agency for Francophonie, AFRINIC, and MENOG. He is also an expert in Internet Governance after completion the ISOC Next generation e-learning programme: "Shaping the Internet, History and Futures" in 2012. He is an ISOC Ambassador to IGF from 2012 to 2013, a Google Panelist in the first Arab-IGF, an ISOC Fellow to IETF'89&92&95&99 and ICANN'50&54 Fellow. He is a member of Task Force for Arabic IDNs which is a part of Global Stakeholder Engagement endorsed by ICANN. He is a member of G6 association for IPv6 and one of the contributors to the IPv6 MOOC.

**Dr. Mohamed Younis** is currently an associate professor in the department of computer science and electrical engineering at the university of Maryland Baltimore County (UMBC). He received his Ph.D. degree in computer science from New Jersey Institute of Technology, USA. Before joining UMBC, he was with the Advanced Systems Technology Group, an Aerospace Electronic Systems R&D organization of Honeywell International Inc. While at Honeywell he led multiple projects for building integrated fault tolerant avionics and dependable computing infrastructure. He also participated in the development of the Redundancy Management System, which is a key component of the Vehicle and Mission Computer for NASA's X-33 space launch vehicle. Dr. Younis' technical interest includes network architectures and protocols, wireless sensor networks, embedded systems, fault tolerant computing, secure communication and distributed real-time systems. He has published over 240 technical papers in refereed conferences and journals. Dr. Younis has seven granted and three pending patents. In addition, he serves/served on the editorial board of multiple journals and the organizing and technical program committees of numerous conferences. Dr. Younis is a senior member of the IEEE and the IEEE communications society.

**Ahmed A. Shahin** is currently an assistant professor in Computer and Systems Engineering Dept. at Zagazig University, Egypt. He has pursued his Ph.D and MSc. in Computer Engineering from University of Maryland Baltimore County (UMBC) in May 2017 and May 2014 respectively. His current research interests include Wireless Sensor Networks, IoT, Cloud Computing, and Metaheuristics for Optimization.