Wireless Netw (2018) 24:3063–3081
https://doi.org/10.1007/s11276-017-1522-1

Efficient data dissemination for Wi-Fi peer-to-peer networks by unicasting among Wi-Fi P2P groups

Meng-Shiuan Pan[1] · Yen-Pei Lin[1]

Published online: 3 May 2017

Abstract Recently, the Wi-Fi peer-to-peer (Wi-Fi P2P) technology is discussed to be able to support communications in infrastructure-less network scenarios. In many of such application scenarios, disseminating data (or information) to all network devices is an important issue. According to the Wi-Fi P2P specification, a device can communicate with other devices after joining a Wi-Fi P2P group. A Wi-Fi P2P group is a star network (rooted at a group owner). The group owner can disseminate data to all network devices by broadcasting. However, the Wi-Fi P2P broadcasting mechanism cannot guarantee successful delivery of packets. In order to disseminate data reliably, a possible solution is to disseminate packets to network devices by unicasting. But, by this manner, the group owner will run out of its energy quickly and the time needed to disseminate data to network devices will be lengthened. To consider the above factors, in this paper, we formally define a Wi-Fi P2P data dissemination (WPDD) problem, and prove that this problem is NP-complete. Instead of using one Wi-Fi P2P group to connect all network devices, we propose to divide devices into multiple groups. We then propose a tree-based dissemination scheme and a ring-based data dissemination scheme to achieve data dissemination among groups. The proposed schemes can be compatible with the Wi-Fi P2P specification. We evaluate the performance and effectiveness of the proposed schemes by simulation programs and prototyping implementations.

✉ Meng-Shiuan Pan
mspan@mail.tku.edu.tw; 602420068@s02.tku.edu.tw

[1] Department of Computer Science and Information Engineering, Tamkang University, New Taipei City 251, Taiwan, ROC

Keywords Data dissemination · Device-to-device communication · Unicasting · WiFi peer-to-peer networks

1 Introduction

In recent years, the Wi-Fi peer-to-peer (Wi-Fi P2P) technology [1] is discussed to be able to support communications in infrastructure-less network scenarios. In many of such application scenarios (e.g., (1) a rescuer announces notifications to people in a disaster scenario; (2) a guide announces information to climbers when climbing; (3) a teacher dispatches instructions to students in an outdoor classroom; (4) a commander announces commands to soldiers in a battlefield; (5) an engineer remotely updates IoT devices' softwares in a smart city environment), disseminating data (or information) to all network devices is an important issue.

Figure 1 shows a Wi-Fi P2P network. In the network, there is a *group owner*. A device can join the network as a *client* after associating to the group owner. Like a Wi-Fi access point, the group owner is responsible for handling the packet exchanges between devices in the group. This work aims to discuss how to disseminate data from any device to all other network devices in a Wi-Fi P2P network. A solution of disseminating data is that a device first sends its data packet to the group owner. Then, the group owner can distribute the data packet to all clients by the IEEE 802.11 media access control (MAC) broadcasting procedure [2]. But, the Wi-Fi P2P broadcasting mechanism cannot guarantee successful delivery of packets since the acknowledgement mechanism is not enabled [3]. In the literature, some works [4–7] propose reliable broadcasting schemes. But, these schemes cannot be compatible with the

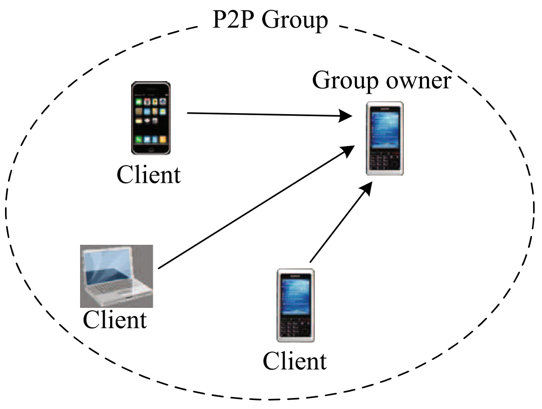


Fig. 1 A Wi-Fi P2P network

IEEE 802.11 MAC layer protocol and cannot be directly applied to the Wi-Fi P2P group transmission mechanism.

Instead of using broadcasting, two APPs (SuperBeam [8] and Wi-Fi Direct+ [9]) send data packets to devices by unicasting, and the group owner can ensure that packets can reach all clients. However, by the unicast transmission fashion, the group owner has to send data to its clients one by one, and there will have the following two drawbacks. First, the group owner's energy will be substantially consumed. Second, the group owner will need a lot of time to distribute data to all clients. It is not hard to see that the above two phenomena will become more severe with increased number of devices in the group.

In this work, to balance devices' energy consumption, we propose to divide network devices into multiple groups (instead of using one group to connect all devices). In a group, the group owner is only responsible for a limited number of clients. So, when receiving a data packet, a group owner only needs to perform a limited number of unicasts. In our design, a device could have two *roles* (i.e., as a client in a group and as the group owner in another group). More specifically, a device can connect to a group owner as a client device, and it can receive/send a data packet from/to its group owner. Then, the device can change to become a group owner in another group, and can unicast packets to its client devices. Based on the above configuration, we model a Wi-Fi P2P data dissemination (WPDD) problem, which goal is to minimize delay of disseminating data to all network devices. We prove that the WPDD problem is an NP-complete problem, and then propose a tree-based data dissemination scheme and a ring-based data dissemination scheme to solve the WPDD problem. For these two schemes, we develop corresponding network formation, operation, and repair procedures. According to our simulation results, the proposed schemes can effectively reduce energy consumption of network devices and shorten the time needed for disseminating data. We also implement the designed schemes on commercial Android platforms. The experimental results show that data packet can be effectively distributed in the formed Wi-Fi P2P networks.

The contributions of this work are threefold. First, given a Wi-Fi P2P network, we discuss how to disseminate data from any device to other network devices efficiently. We formally model a WPDD problem, which goal is to minimize delay of disseminating data to network devices. In this work, we prove the WPDD problem is an NP-complete problem. To the best of our knowledge, this is the first work that formally models the problem and shows the difficulty of the problem under such network scenario. Second, instead of connecting devices by a single group, we propose a concept which involves the cooperation of multiple groups in a Wi-Fi P2P network. This idea can effectively balance energy consumption of network devices when disseminating data. Third, the proposed tree-based and ring-based data dissemination schemes can facilitate communication between multiple groups in a Wi-Fi P2P network, and can shorten delay on disseminating data. More importantly, the proposed schemes can be compatible with the Wi-Fi P2P specification.

The rest of this paper is organized as follows. Section 2 present related works, Wi-Fi P2P overview, and network models. Sections 3 and 4 present the proposed tree-based and ring-based data dissemination schemes, respectively. Simulation and prototyping implementation results are shown in Sect. 5. Finally, Sect. 6 concludes this paper.

2 Preliminaries

2.1 Related works

In the literature, the wired peer-to-peer communication protocols (e.g., [10–12]) are widely discussed. However, those solutions are not suitable for Wi-Fi P2P networks. In references [13–15], the authors propose data transmission schemes based on mobile social networks (MSNs). In an MSN, data transmissions rely on opportunity transmissions. So, these works focus on how to increase packet arrival rates and enlarge transmission scopes. In some researches, the authors assume that an MSN exists an access point (AP), which can be used to further accelerate the speed of data dissemination. However, when disseminating a data packet, the opportunity transmission fashion cannot guarantee that all devices can receive the packet. References [16–18] propose schemes to increase performance of Wi-Fi networks by the concept of cooperating among Wi-Fi APs. In [16, 17], Wi-Fi APs' transmission power is coordinated to reduce interferences. In [18], the operating channels of Wi-Fi APs are carefully arranged to decrease congestion. The proposed schemes in [16–18] can indeed achieve their goals, but these solutions cannot directly apply to Wi-Fi P2P networks.

References [19–22] introduce communication protocols for Wi-Fi P2P networks. In reference [19], the authors integrate Wi-Fi P2P with session initiation protocol (SIP). The goal is to use SIP to manage devices' joining and leaving a Wi-Fi P2P network. In reference [21], the authors combine a Wi-Fi P2P network with an existing 3G cellular network. Their idea is to place a central server in the cellular network, and the server is used to manage devices. When in a hot-spot area, a user's device first communicates with the server. Then, according to the location information of the device, the server assigns the device to an existing Wi-Fi P2P group. As a result, the members in the group can communicate directly without the help of the cellular network. Reference [20] introduces a system that allows devices to manage their connections by their local databases. A device can communicate with another device directly if the target device is found in its local database. We can see that references [19–21] focus on how to manage devices and P2P groups, and do not consider the communications between groups. When using their schemes to distribute data, the transmissions rely on a single group owner, and thus results in long transmission delay and large power consumption. Moreover, in reference [22], the authors propose to use the reversed field of Wi-Fi P2P control frame to carry information. By the proposed scheme, devices can receive broadcast packets without associating to a group owner. But, this scheme is only suitable for transmitting small-sized data packets.

References [3, 23, 24] present inter-group communication protocols for Wi-Fi P2P networks. Reference [23] utilizes the concurrent mode specified in the Wi-Fi P2P specification to achieve inter-group communication. Assume that there are two Wi-Fi P2P groups. In the designed scheme, there will have a bridge device, which can be one group's group owner and be another group's client, simultaneously. The bridge device is responsible for relaying packet among these two groups. But, the concurrent mode will cause the IP address conflict problem. So, the bridge node should have two radio interfaces. Reference [3] presents a scheme to use one radio interface to achieve concurrent mode operation. In their scheme, a bridge node first forms a group and accepts connections from other devices by the Wi-Fi P2P association procedure. At the same time, the bridge node also performs the legacy Wi-Fi association procedure to connect to another group owner. In [3], the authors resolve the IP address conflict by restricting that the bridge device can only use Wi-Fi P2P broadcast mechanism to send packets to its group clients. However, as reported in their experiments, the packet loss ratio is high when broadcasting. Moreover, in reference [24], the authors propose a group communication scheme based on delay-tolerant routing concept. In the proposed scheme, when a device wants to send a packet to another device, this device first sends the packet to its group owner. Then, the group owner checks its routing table. If the destination device does not locate in its group, the group owner finds several devices, which have lower RSSI values, as the next hops. The scheme is so designed because that those devices that have lower RSSI values are prone to leave the group, and thus those devices will have higher probabilities to find the destination node. It is not hard to see that in [24], packets may not be able to reach destinations.

2.2 Overview on Wi-Fi P2P

The goal of Wi-Fi peer-to-peer (Wi-Fi P2P) specification [1] is to allow Wi-Fi capable devices to be able to communicate with each other through direct links or P2P links. According to the specification, when a device enters a network, this device needs to perform *device discovery procedure* to find a group owner. If a group owner is found, the device can ask to join that group by sending an *association request packet* to the group owner. If there is no existing group, the device can compete for being a group owner by the *group owner negotiation procedure*. In the following, we briefly introduce the *device discovery*, *group owner negotiation*, and *service discovery* procedures defined in the Wi-Fi P2P specification.

1. *The device discovery procedure* When a device $\mathcal{D}_i$ does not join a P2P group, it can use the *device discovery procedure* to detect its nearby devices. This procedure is divided into a *scan phase* and a *find phase*. When starting discovery, $\mathcal{D}_i$ first performs the scan phase to collect information of its nearby P2P devices and groups in all Wi-Fi P2P *social channels* (i.e., channel 1, 6, and 11 in the 2.4 GHz band). If $\mathcal{D}_i$ receives a beacon from a group owner during the scan phase, it can perform the association procedure to join the group. If no group owner is found and the scan phase is expired, $\mathcal{D}_i$ enters the find phase. In the find phase, $\mathcal{D}_i$ alternatively switches between *listen state* and *search state*. First, $\mathcal{D}_i$ stays in the listen state for a time interval with length 100, 200, or 300 time units (where the length of a time unit is defined in IEEE 802.11 specification [2]). During the time interval, $\mathcal{D}_i$ stays in a channel. If $\mathcal{D}_i$ receives a *probe request packet* from another device $\mathcal{D}_j$, device $\mathcal{D}_i$ will then reply a *probe response packet* to $\mathcal{D}_j$. After exchanging the probe request and response packets, the devices $\mathcal{D}_i$ and $\mathcal{D}_j$ are said to have been discovered by each other. Then, $\mathcal{D}_i$ enters search state and stays in each of Wi-Fi P2P social channel for 100 time units, and can send probe request packets to discover other devices. Moreover, in the find phase, if $\mathcal{D}_i$ learns that a device $\mathcal{D}_j$ is associated with a group from $\mathcal{D}_j$'s probe response packet, $\mathcal{D}_i$ can also ask to join that group.

2. *The group owner negotiation procedure* Assume that there is no existing group in the network. Two devices, say $\mathcal{D}_i$ and $\mathcal{D}_j$, can negotiate for becoming a group owner. First, devices $\mathcal{D}_i$ locally decides an intent value. According to the specification, if $\mathcal{D}_i$ wants to be the group owner, it can simply set the intent value to be 15. Otherwise, $\mathcal{D}_i$ randomly selects a value between 0 to 14 as its intent value. Then, devices $\mathcal{D}_i$ further decides a tie breaker value to be 0 or 1. At the same time, the device $\mathcal{D}_j$ also decides its intent and tie breaker value. Then, $\mathcal{D}_i$ and $\mathcal{D}_j$ can broadcast *GO negotiation request packet*. Without loss of generality, we assume that $\mathcal{D}_i$ broadcasts a GO negotiation request packet earlier than $\mathcal{D}_j$. When $\mathcal{D}_j$ receives the GO negotiation request packet from $\mathcal{D}_i$, $\mathcal{D}_j$ simply replies a *GO negotiation response packet* containing its intent value. After receiving the GO negotiation response packet from $\mathcal{D}_j$, $\mathcal{D}_i$ decides which devices can be the group owner according to the following rules. (Assume that the intent values of $\mathcal{D}_i$ and $\mathcal{D}_j$ to be I_i and I_j, respectively)
   - If $I_i > I_j$ (resp., $I_i < I_j$), $\mathcal{D}_i$ (resp., $\mathcal{D}_j$) will be the group owner.
   - If $I_i = I_j < 15$, $\mathcal{D}_i$ (resp., $\mathcal{D}_j$) will be the group owner if tie breaker value is 1 (resp., 0).
   - If $I_i = I_j = 15$, the negotiation fails because that both $\mathcal{D}_i$ and $\mathcal{D}_j$ want to be the group owner.

   After the above procedure, $\mathcal{D}_i$ sends the result to $\mathcal{D}_j$ by a *GO negotiation confirm packet*. Then, the decided group owner can operate on the channel designated in the GO negotiation confirm packet, and can accept association requests from other devices.

3. *The service discovery procedure* in a Wi-Fi P2P network, a device uses the service discovery procedure to find services offered by other devices. According to the specification, a service provider cannot directly broadcast the provided service by itself. When a device wants to seek services, this device has to send *SD query packets* to discover the offered services of its nearby devices. In the SD query packet, the sender can also explicitly specify the service that it is seeking. A service provider that receives an SD query packet will then reply a *SD response packet* to notify the offered services and the related parameters used by its services. In this work, we design services to facilitate network management.

2.3 Network models

Given a Wi-Fi P2P network, we model it by a graph $G = (V, E)$, where V contains all Wi-Fi P2P capable devices, and E contains all symmetric communication links between devices in V. The network has C available channels for communication. Those $|V|$ devices will be divided into multiple Wi-Fi P2P groups (instead of one group), and thus there will have multiple group owners in the network. Each group owner selects one channel to operate. A client device needs to switch to its group owner's channel when it wishes to communicate with its group owner. In the network, each device can disseminate a packet to all other network devices, and the packet will be sent to other devices by unicasting. To facilitate data dissemination, a device is able to belong to at most two groups, and thus it will be assigned to two roles (i.e., this device is (1) a group owner of a group and (2) a client device in another group). Figure 2 indicates the basic rule of data dissemination, where there are two groups (G_1 and G_2) and device u has two roles. Assume that v has a packet to all network devices. In Fig. 2(a), v serves as the group owner of group G_1. At this time, v can unicast packet to all its client devices. Then, in Fig. 2(b), device u becomes a group owner, and it can further unicast the packet (which received from v in G_1) to its clients. By the above network configuration, we can see that when a group owner has more client devices, it needs to unicast a packet more times when disseminating. To balance the load of group owners in the network, we further demand that the maximum number of devices in a group will be restricted to M_d.

According to the formed groups, we can construct a group relationship graph $G_g = (V, E_g)$, where $G_g \subseteq G$ and E_g represents the links of client devices to their group owners. A device that has two roles will have links to its client devices and a link to the connected group owner in E_g. Data disseminations in the network are based on the G_g. When transmission, devices may suffer from hidden terminal and exposed terminal interferences. So, by G and G_g, we can further derive the interference relationships between links. If two links are interference links, the corresponding end points (i.e., devices) of these two links cannot send or receive packets at the same time. For example, Fig. 3 indicates the G_g of Fig. 2. In Fig. 3, links e_0 and e_1 are interference links because that e_0 and e_1 have a common end point (i.e., device v). In Fig. 2, if G_1 and G_2 operate in the same channel, links e_1 and e_2 (in Fig. 3) are also interference links since devices v and u will interfere with each other. On the other hand, if G_1 and G_2 operate in different channels, links e_1 and e_2 are interference-free.

Assume that a unicast transmission between two peer devices needs a *transmission time unit* to complete. When a device v generates a packet, an interference-free transmission schedule can be arranged based on G_g and the interference relationship between links. For example, in Fig. 2, if G_1 and G_2 operate in the same channel, a packet generated by device v will need 7 transmission time units to

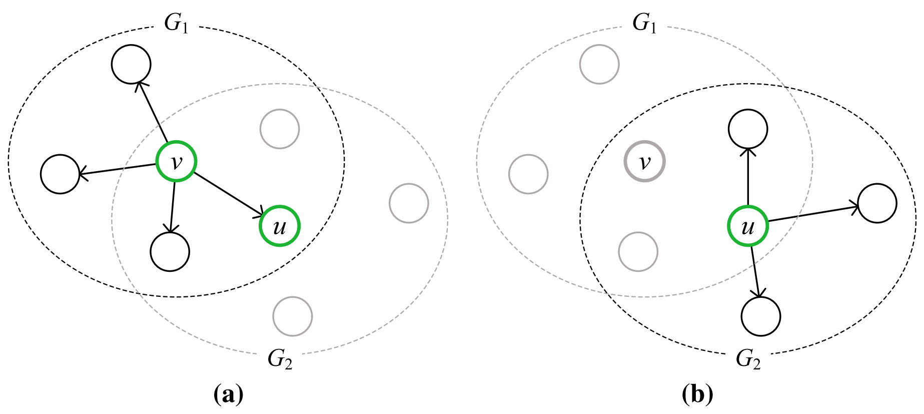
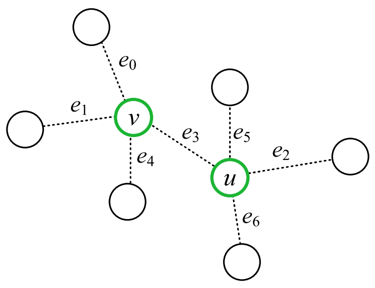


Fig. 2 The basic rule of data dissemination, where device u has two roles. **a** Device v unicasts packets to its clients. **b** Device u unicasts the received packets to its clients



Time	*Case* 1	*Case* 2
0	e_3	e_3
1	e_0	e_0 / e_2
2	e_1	e_1 / e_5
3	e_4	e_4 / e_6
4	e_2	
5	e_5	
6	e_6	

Case 1: G_1 and G_2 operate in the same channel

Case 2: G_1 and G_2 operate in different channels

Fig. 3 The group relationship graph G_g of Fig. 2

distribute to all network devices. (An example of the schedule is shown as the Case 1 in Fig. 3.) On the other hand, in Fig. 2, if G_1 and G_2 operate in different channels, a packet generated by v only needs 4 transmission time units to distribute, i.e., device v first sends the packet to u, and then v and u can unicast this packet to their clients in another three transmission time units at the same time (as shown as the Case 2 in Fig. 3). So, after deciding an interference-free transmission schedule, we can define the *data dissemination time* $d(v)$ of device v to be the number of transmission time units required to complete data dissemination from v to all network devices. Then, we can further define the data dissemination time of the graph G to be $d(G) = \max\{d(v)|v \in V\}$.

Definition 1 Given a Wi-Fi P2P network $G = (V, E)$, the group relationship $G_g = (V, E_g)$, C available channels, multiple Wi-Fi P2P groups containing $|V|$ devices, and group size constraint M_d, the Wi-Fi P2P data dissemination (WPDD) problem is to find an interference-free transmission schedule such that the data dissemination time $d(G)$ is minimized.

We can further revise the WPDD problem to a decision problem.

Definition 2 Given a Wi-Fi P2P network $G = (V, E)$, the group relationship $G_g = (V, E_g)$, C available channels, multiple Wi-Fi P2P groups containing $|V|$ devices, and group size constraint M_d, the decision version of the Wi-Fi P2P data dissemination (DWPDD) problem is to find an interference-free transmission schedule such that the data dissemination time $d(G) \leq d$, where d is a delay constraint.

Moreover, in [25], the authors show a Bounded-Degree Minimum Broadcast Time (BDMBT) problem as below. (Assume that the transmission time unit for completing a packet exchange in [25] is the same as ours.)

Definition 3 Given a graph $G_b = (V_b, E_b)$ with maximum degree M_b, the BDMBT problem is to find a transmission schedule such that (1) at any time instant, the endpoints of the scheduled edges will not overlap and (2) the broadcast time is less than a constraint K_b.

The authors in [25] prove that the BDMBT problem is NP-complete by reducing the 3SAT problem to a special case of BDMBT. In this work, we utilize the BDMBT problem to prove the follow theorem.

Theorem 1 *The DWPDD problem is NP-complete.*

Proof First, given a transmission schedule, it is not hard to see that for each $v \in V$, we can check if $d(v) \leq d$ in polynomial time. This implies that we can check if $d(G) \leq d$ in polynomial time. So, the DWPDD problem is NP.

Next, to prove that the DWPDD problem is NP-hard, we first reduce the BDMBT problem to a single-channel-DWPDD (sc-DWPDD) problem. The sc-DWPDD problem is a special case of DWPDD problem with $C = 1$. After proving that the sc-DWPDD problem is NP-hard, we can further reduce the sc-DWPDD problem to the DWPDD problem easily, and thus the DWPDD problem will be NP-hard.

In the following, we show the details on reducing the BDMBT problem to the sc-DWPDD problem: let $G_b = (V_b, E_b)$, integer M_b, and constraint K_b are arbitrary instances of the BDMBT problem. Now, we first transform G_b to instances G and G_g of the sc-DWPDD problem, where G and G_g will be the same in this transformation. For a vertex v_b in G_b, we generate a vertex v in G. For any two neighboring vertices v_b^i and v_b^j in V_b, if both of the degrees of v_b^i and v_b^j are larger than one, we generate an intermediate vertex v^{ij} locating between the corresponding vertices v^i and v^j in G. For an edge e_b, which connects two neighboring vertices v_b^i and v_b^j in V_b, there are two cases: (1) a edge will be generated in G to connect v^i and v^j if there is no intermediate vertex between v^i and v^j; (2) Two edges will be generated in G to connect (v^i, v^{ij}) and (v^{ij}, v^j) if there is an intermediate vertex v^{ij} between v^i and v^j. In the above transformation, the rationale of adding intermediate vertices are as below. When scheduling, the BDMBT problem only requires that the endpoints of the scheduled edges should not be overlapped. But, in the sc-DWPDD problem, two neighboring vertices cannot be scheduled at the same time (e.g., vertices v and u cannot transmission at the same time in the Case 1 of Fig. 3). So, an intermediate vertex is added to avoid interference when transforming the schedule of BDMBT to sc-DWPDD. So, when a schedule between two vertices v_b^i and v_b^j in BDMBT is decided, the corresponding schedule in sc-DWPDD will need two extra transmission time units.

We now claim that we can find a transmission schedule which satisfies the broadcast time is less than K_b for the BDMBT problem if and only if we can find a transmission schedule for the sc-DWPDD problem such that $d(G) \leq d$, where $d = (K_b + 2 \times D_{ia})$ and D_{ia} is the network diameter of G_b.

To prove the *if* part, if there is a transmission schedule for the sc-DWPDD problem such that $d(G) \leq d$, we can find a schedule in G_b such that the broadcast time is less than K_b. Based on the above transformation, an intermediate vertex in G will induce two extra transmission time units. Since the network diameter of G_b is D_{ia}, the broadcast paths to all network nodes will be lengthened at most $2 \times D_{ia}$ transmission time units. Thus, based on the schedule in sc-DWPDD, a corresponding schedule in BDMBT with time less than K_b can be found. Conversely, to prove the *only if* part, suppose that there is a transmission schedule that can achieve the broadcast time of BDMBT problem to be smaller than K_b. Again, based on our transformation, the corresponding schedule in G will be lengthened at most $2 \times D_{ia}$ transmission time units. Thus, a schedule with $d(G) \leq K_b + 2 \times D_{ia}$ can be determined. $\square$

In this work, we propose two Wi-Fi P2P compatible schemes to solve the WPDD problem. In the proposed schemes, a packet will first be distributed in the group that the packet initiator located, and then be relayed to devices in other groups through the designed tree or ring topology. Following the Wi-Fi P2P specification, we design *services* to support the network formation and network repair procedures. In our design, a device may have at most two roles. To facilitate changes of devices' roles, we divide network time into continually repeated *slots*, where the size of each slot is t_s. Every two slots are designated as a *time frame*. In a time frame, the two slots are labeled as slot 0 and slot 1, respectively. At the beginning of a time slot, a device changes its role and connects to the designated group owner according to the slot number. We remark that the decided slot size is assumed to be large enough to satisfy the following two criteria: first, the group owner can complete unicasting a packet to each of its group clients within a slot. Second, a network device can finish a service discovery procedure on any service within a slot. Moreover, devices perform a local synchronization mechanism to synchronize their time frames with their group owners'. (We will show the implemented local synchronization mechanism in Sect. 5.2.) When communication, devices follow the IEEE 802.1 MAC protocol to compete for the wireless medium, and RTS/CTS handshacking mechanism is applied to resolve hidden terminal and exposed terminal interferences.

To ease of presentation, in the rest of this paper, we use the notation $\mathcal{D}$ and $\mathcal{O}$ to represent a network device and a group owner, respectively. In our schemes, each group will be assigned to a unique group ID G_{id}, and the value of G_{id} will be equal to the MAC address of the group's group owner $\mathcal{O}$, i.e., $G_{id} = mac(\mathcal{O})$.

3 The tree-based data dissemination scheme

In this section, we introduce the proposed tree-based data dissemination scheme. Figure 4(a) shows an example that 15 devices are connected by a tree topology. In this scheme, one device is taken as the *network root*, which is set to be located at layer 0. When operation, devices change to be group owners or client devices according to current slot number. As shown in Fig. 4(b), at slot 0, the network root and those devices that are located in even-numbered layers will be group owners. These group owners take those devices that are located in odd-numbered layers as their client devices. On the other hand, at slot 1, devices that are located in odd-numbered layers will be group owners. Note that a device will not change its role if it is the network root or a *leaf device* (i.e., a device with no client device). Also note that the tree topology in Fig. 4 will be constructed by the proposed network formation scheme, which selects tree links from communication links between devices in a distributed manner.

In this scheme, we design three Wi-Fi P2P services to support network formation and repair.

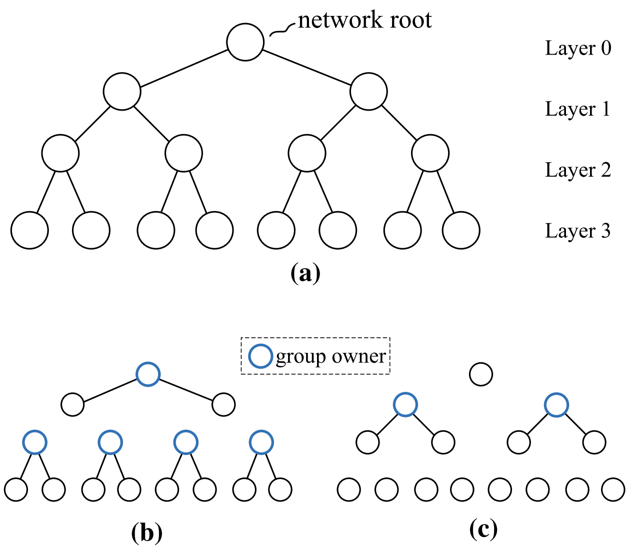


Fig. 4 **a** The connections in a tree topology. **b** The connections at slot 0. **c** The connections at slot 1

1. *FormTreeSvc* ($mac(\mathcal{R})$, $mac(\mathcal{O})$, G_{id}, *slot*, *layer*, *client_num*, M_d) This service is provided by the group owner $\mathcal{O}$, which still has room to accept association requests. In this service, the parameter $mac(\mathcal{R})$ represents the network root's MAC address, *slot* represents $\mathcal{O}$'s operating slot (i.e., slot 0 or slot 1), *layer* represents the layer of $\mathcal{O}$ in the tree, and *client_num* represents the current number of client devices in the group.
2. *LostParSvc*(G_{id}) This service is provided by a client device (which realizes that its group owner is gone) to start the designed instant network repair procedure.
3. *RstGroupSvc*($mac(go(\mathcal{O}_{lost}))$, $mac(\mathcal{O}_{lost})$, $mac(\mathcal{D}_{cdt})$) This service is also used for instant network repair to restore a group. When a group owner $\mathcal{O}_{lost}$ leaves the network, this service asks the chosen candidate device $\mathcal{D}_{cdt}$ to replace $\mathcal{O}_{lost}$, and then the group (originally formed by $\mathcal{O}_{lost}$) can be restored. Note that the first parameter $go(\mathcal{O}_{lost})$ represents the group owner of the device $\mathcal{O}_{lost}$.

Note that our design follows the service discovery procedure in the Wi-Fi P2P specification. In our scheme, devices send SD query packets to other devices to query the above services. If a service provider receives an SD query packet, it will append the contents of the provided service in its SD response packet.

3.1 Network formation

Assume that a device $\mathcal{D}_i$ enters the scope of the network, and does not join a group. Device $\mathcal{D}_i$ first decides a discovery timer, which length is randomly decided within a predefined range. During the discovery timer, $\mathcal{D}_i$ queries if there are *FormTreeSvc* service providers (i.e., group owners). If $\mathcal{D}_i$ finds a group owner, which has room to accept it, $\mathcal{D}_i$ stops the discovery timer and then sends an association request packet to that group owner. Assume that $\mathcal{D}_i$ associates to group owner $\mathcal{O}_j$, and $\mathcal{O}_j$ operates at slot $s(\mathcal{O}_j)$ and locates at layer $l(\mathcal{O}_j)$. Device $\mathcal{D}_i$ then further sets its operating slot $s(\mathcal{D}_i) = (s(\mathcal{O}_j) + 1)\%2$ and its layer $l(\mathcal{D}_i) = l(\mathcal{O}_j) + 1$. In each of slot numbered $s(\mathcal{D}_i)$, $\mathcal{D}_i$ can provide its *FormTreeSvc*($mac(\mathcal{R})$, $mac(\mathcal{D}_i)$, $mac(\mathcal{D}_i)$, $s(\mathcal{D}_i)$, $l(\mathcal{D}_i)$, 0, M_d) service until it has no room to accept other devices' association requests. On the other hand, if $\mathcal{D}_i$ cannot find any *FormTreeSvc* service, $\mathcal{D}_i$ can follow the group negotiation procedure to compete for becoming a group owner. If $\mathcal{D}_i$ wins the negotiation, it first sets $l(\mathcal{D}_i) = 0$ and $s(\mathcal{D}_i) = 0$, and then can start its *FormTreeSvc*($mac(\mathcal{D}_i)$, $mac(\mathcal{D}_i)$, $mac(\mathcal{D}_i)$, 0, 0, 0, M_d) service. In this case, $\mathcal{D}_i$ will be considered as the network root.

In our design, when a device realizes that there are multiple *FormTreeSvc* services, it will choose the service provider, which is located near to the network root, to be its group owner. This design can help to reduce the tree height. Moreover, it is possible that more than one device become the network root. Assume that a device $\mathcal{D}_i$ realizes another device $\mathcal{D}_j$ such that $\mathcal{D}_j$'s *FormTreeSvc* service carries the $mac(\mathcal{R})$ value, which is different to the one carried by $\mathcal{D}_i$'s *FormTreeSvc* service. At this time, the network contains two network root. In this case, $\mathcal{D}_i$ checks if the MAC address of its network root is smaller than that of $\mathcal{D}_j$'s network root. If so, $\mathcal{D}_i$ considers that its priority is lower. Then, device $\mathcal{D}_i$ will distribute an *existingTreeCmd*($mac(\mathcal{R}_j)$) packet, where $mac(\mathcal{R}_j)$ is the MAC address of $\mathcal{D}_j$'s network root. When a device receives an *existingTreeCmd* packet, it first relays this packet to its client device and its group owner. Then, this device will dismiss its client devices and try to associate to a new group owner that are located in $\mathcal{R}_j$'s tree. The above procedures are to make sure that the network will have only one tree.

Theorem 2 *After the network root is decided, the shortest and longest time needed for network formation will be* $t_s \times O(\log|V|)$ *and* $t_s \times O(|V|)$, *respectively.*

Proof After the network root is decided, the network root can start to accept association requests for at most $M_d - 1$ devices in one slot. Then, when a device associates to the network root or a group owner, this device can become a group owner, and it can further accept association requests from $M_d - 1$ devices in its slot. So, in the best case, the tree height will be bounded by $O(\log|V|)$, which implies we need $O(\log|V|)$ slots to connect all network devices. As a result, the shortest network formation time will be

$t_s \times O(\log|V|)$. Moreover, in the worst case, each device can only connect one child device in one slot. So, $O(|V|)$ slots are needed to connect all devices. As a result, the longest network formation time will be $t_s \times O(|V|)$. □

3.2 Disseminating packets

In the following, we describe the procedures of disseminating packets. Assume that a device $\mathcal{D}_i$ generates or receives a data packet. If $\mathcal{D}_i$ is a group owner, it will transmit the packet to all of its client devices in its operating slot $s(\mathcal{D}_i)$. Then, device $\mathcal{D}_i$ sends the packet to its group owner $go(\mathcal{D}_i)$ in its group owner's operating slot.

3.3 Network repair

The network repair is needed because that devices may leave the network or may run out of their batteries. In our design, the network repair can be divided into *regular repair* and *instant repair*. First, the regular repair is performed locally by leaf devices. For a leaf device $\mathcal{D}_{leaf}$, it will periodically query *FormTreeSvc* services from other devices. If $\mathcal{D}_{leaf}$ finds a group owner $\mathcal{O}_i$, which is located closer to the network root than $go(\mathcal{D}_{leaf})$ and $\mathcal{O}_i$ still has room to accept new client devices, $\mathcal{D}_{leaf}$ will re-associate to $\mathcal{O}_i$.

Next, in our instant repair scheme, each group owner selects a client device, which has the smallest MAC address, as the *agent device*. The agent device periodically queries existing *FormTreeSvc* services, and records several service providers, which are leaf devices, in its local database to be *candidate devices*. If the original group owner of the agent device leaves the network, one of the candidate devices will be asked to be the new group owner. Moreover, a group owner and its client devices periodically exchange *aliveCmd* packets according to the following rules:

- For a group owner $\mathcal{O}_i$, it sends *aliveCmd*$(mac(go(\mathcal{O}_i)))$ and *aliveCmd*$(null)$ packets to the agent device and other client devices, respectively. (A client device can know that it is an agent device if it receives *aliveCmd* packets with $mac(go(\mathcal{O}_i))$ information.)
- For a client device that receives an *aliveCmd* packet from its group owner, it will reply an *aliveAckCmd* to that group owner.

Figure 5 illustrates the basic procedures of the instant repair. The instant repair is triggered by client devices, which find that their group owner $\mathcal{O}_{lost}$ leaves the network. Then, the agent device informs the selected candidate device $\mathcal{D}_{cdt}$ to restore the group by the *RstGroupSvc* service. From the first field of *RstGroupSvc* service, the device $\mathcal{D}_{cdt}$ knows that it needs to re-associate to $go(\mathcal{O}_{lost})$. Then, after $\mathcal{D}_{cdt}$ reconnects $\mathcal{O}_{lost}$'s client devices, the instant repair finishes.

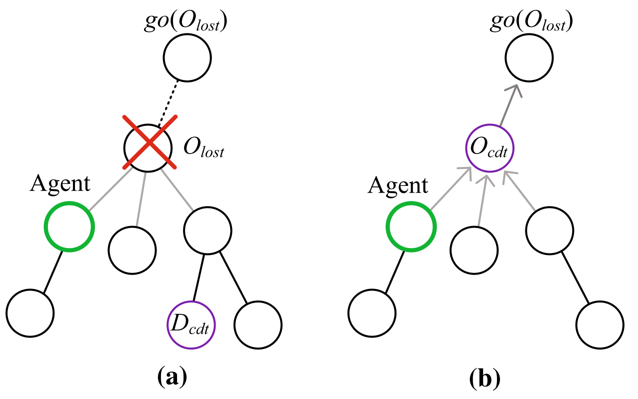


Fig. 5 **a** A non-leaf group owner $\mathcal{O}_{lost}$ leaves the network. **b** The candidate device $\mathcal{D}_{cdt}$ restores the group

In the following, we show the details of the instant repair. Assume a group owner $\mathcal{O}_{lost}$ (which has at least one client device) leaves the network. For a client device of $\mathcal{O}_{lost}$, say $\mathcal{D}_i$, if it cannot detect $\mathcal{O}_{lost}$ for a predefined T_{lost} slots, $\mathcal{D}_i$ will perform the following operations:

- It starts *LostParSvc*$(mac(\mathcal{O}_{lost}))$ service.
- It queries *LostParSvc*$(mac(\mathcal{O}_{lost}))$ from its sibling devices.

Then, after waiting for a time frame, $\mathcal{D}_i$ will perform the following procedures.

1. If $\mathcal{D}_i$ does not detect any *LostParSvc*$(mac(\mathcal{O}_{lost}))$ service from other devices, this implies $\mathcal{D}_i$ is the only client device of $\mathcal{O}_{lost}$. At this time, $\mathcal{D}_i$ will re-associate to another group owner.
2. If $\mathcal{D}_i$ detects *LostParSvc* services from its sibling devices, there are two cases.
   (a) If $\mathcal{D}_i$ is not an agent device, $\mathcal{D}_i$ will try to re-associate the designated candidate device, i.e., $\mathcal{D}_{cdt}$, by querying a *FormTreeSvc* service, which carries $G_{id} = mac(\mathcal{O}_{lost})$.
   (b) If $\mathcal{D}_i$ is an agent device, $\mathcal{D}_i$ starts the restore service by setting *RstGroupSvc*$(mac(go(\mathcal{O}_{lost})), mac(\mathcal{O}_{lost}), mac(\mathcal{D}_{cdt}))$ to inform $\mathcal{D}_{cdt}$ to replace $\mathcal{O}_{lost}$. (Note that if $\mathcal{O}_{lost}$ is network root, $\mathcal{D}_i$ sets $mac(go(\mathcal{O}_{lost})) = -1$.)

Once the device $\mathcal{D}_{cdt}$ detects a *RstGroupSvc* service, which contains its MAC address, $\mathcal{D}_{cdt}$ will perform the following procedures.

1. Device $\mathcal{D}_{cdt}$ first re-associates to $go(\mathcal{O}_{lost})$, and derives $l(\mathcal{D}_{cdt})$ and $s(\mathcal{D}_{cdt})$ information from the *FormTreeSvc* service of $go(\mathcal{O}_{lost})$.
2. Device $\mathcal{D}_{cdt}$ starts *FormTreeSvc*$(mac(\mathcal{R}), mac(\mathcal{D}_{cdt}), mac(\mathcal{O}_{lost}), s(\mathcal{D}_{cdt}), l(\mathcal{D}_{cdt}), 0, M_d)$. (Note that the G_{id} field will be $mac(\mathcal{O}_{lost})$.) Then, $\mathcal{D}_{cdt}$ accepts association requests from $\mathcal{O}_{lost}$'s client devices.

3. After waiting association requests for a predefined period, $\mathcal{D}_{cdt}$ sends *changeGidCmd*($mac(\mathcal{D}_{cdt})$) packets to its client devices to change the group ID to be $mac(\mathcal{D}_{cdt})$.

Note that the group owner $go(\mathcal{O}_{lost})$ can also realize that $\mathcal{O}_{lost}$ has left the network. So, $go(\mathcal{O}_{lost})$ will reserve a space for $\mathcal{D}_{cdt}$. Also note that when a device detects a *FormTreeSvc* service, in which the recorded $mac(\mathcal{O})$ is not equal to the recorded G_{id}, the device will not associate to the service provider. In the above procedures, a leaf device has to query *RstGroupSvc* services every slot. This design may be inefficient since *RstGroupSvc* services can be provided by any agent device. To accelerate the instant network repair, after an agent device has selected its candidate devices, the agent device disseminates its choices to the network. When a leaf device realizes that it is chosen as candidate devices by some agent devices, this leaf device only queries the corresponding *RstGroupSvc* services from those agent devices. Moreover, it is not hard to see that the needed time for instant repair will be roughly $T_{lost} + 2 \times t_s + T_d$, where T_d is the time that needed by the candidate device $\mathcal{D}_{cdt}$ to detect the corresponding *RstGroupSvc* service.

Theorem 3 *The time needed for disseminating a packet to all network devices will be* $t_s \times (2 \times O(\log |V|) - 1)$.

Proof In our scheme, the regular network repair periodically maintains the height of the tree network, and tree height will be bounded by $O(\log |V|)$. The longest path from a leaf device to anther leaf device will be $2 \times O(\log |V|)$. However, when the network root receives a packet from its client, it can relay the packet to all its client devices in the same slot. So, the effective path length can be reduced to $2 \times O(\log |V|) - 1$, and thus the needed dissemination time will be $t_s \times (2 \times O(\log |V|) - 1)$. □

Theorem 4 *After instant network repair, the network is loop-free.*

Proof In the proposed instant network repair scheme, an agent device always selects leaf devices to be the candidate devices. Recall that in a tree network, there must have leaf devices. When moving a leaf device to be an in-tree node, there will have no impact on the original tree topology. So, after instant network repair, the network can be loop-free. □

4 The ring-based data dissemination scheme

In the following, we present the ring-based data dissemination scheme. Before showing the details, we first give two observations on the tree-based scheme.

- When forming a tree topology in Sect. 4, a device has to compete with other devices to become a group owner or associate to an existing group owner. If fails, the device needs to wait for at least one slot. This design may lead to long network formation time.
- When a leaf device of the tree has a data packet for network devices, this packet has to travel through the entire tree. This design may lead to long data dissemination delay.

Thus, the ring-based scheme is designed to improve network formation time and data dissemination delay.

Figure 6(a) indicates the topology of the ring-based scheme. In the ring topology, each group owner chooses two client devices as *bridge devices*, named *bgToNext* and *bgForPrev*, which connect to the next group and previous group, respectively. Other client devices in a group are taken as *leaf devices*. To accelerate data dissemination speed, leaf devices can establish *shortcuts* to other groups. In this scheme, network devices also have different roles in different slots. At slot 0, devices connect to their group owners to perform intra-group communications [as shown in Fig. 6(b)]. On the other hand, at slot 1, devices perform inter-group communications. As shown in Fig. 6(c), each bgToNext device connects to the bgForPrev device of its next group, and leaf devices connect to other leaf devices in other groups through shortcuts.

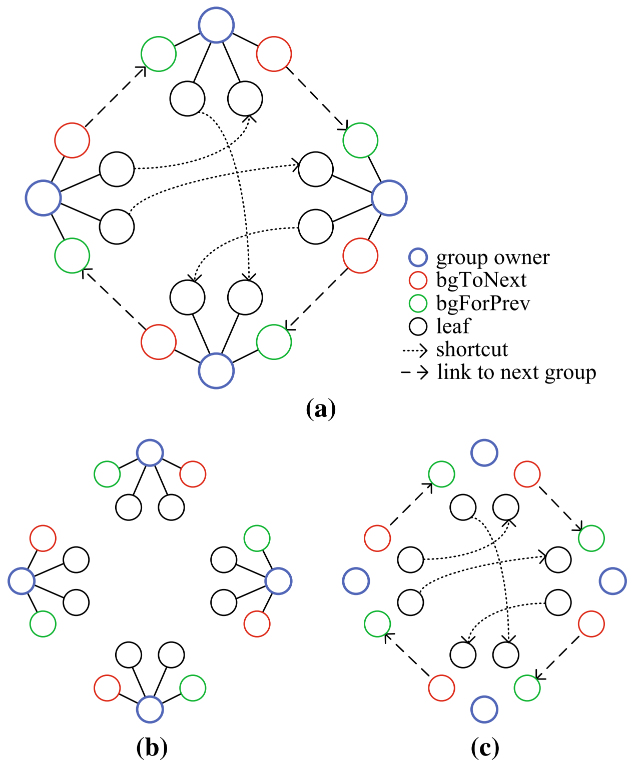


Fig. 6 **a** The connections in a ring topology. **b** The connections at slot 0. **c** The connections at slot 1

In this scheme, four Wi-Fi P2P services are designed to support network formation and repair.

1. *FormRingSvc*(*est_frame*, *cur_frame*, *mac*($\mathcal{O}$), G_{id}, *client_num*, M_d): the functionality of this service is similar to the *FormTreeSvc* service. If the service provider can accept association requests, this service can be discovered. The parameter *est_frame* represents the time frame number that the device $\mathcal{O}$ became a group owner, and *cur_frame* represents the current time frame.
2. *ConnGroupSvc*(G_{id}): this service is provided by a bgForPrev device, where G_{id} is the ID of the group that the service provider is located. This service is used to inform the bgToNext device in the previous group to associate to the service provider.
3. *MergeRingSvc*(*est_frame*, G_{id}): this service is provided by group owners. The functionality of this service is to merge rings that were formed in different time frames.
4. *EstScSvc*(*mac*($\mathcal{D}_{leaf}$)): this service is provided by leaf devices. Leaf devices use this service to establish shortcuts.

4.1 Network formation

Assume that a device $\mathcal{D}_i$ enters the scope of the network. Device $\mathcal{D}_i$ first searches if there exists *FormRingSvc* services. If found, $\mathcal{D}_i$ can request to associate to the service provider. If there are multiple choices, $\mathcal{D}_i$ selects the service provider, which has the smallest MAC address, as its group owner. However, after a period time, if $\mathcal{D}_i$ cannot associate to a group owner, $\mathcal{D}_i$ then locally decides itself to become a group owner by a probability p, where p is a predefined system parameter. If $\mathcal{D}_i$ becomes a group owner, it can provide *FormRingSvc* service in the upcoming time frame. Assume there is no group owner in the network. The new group owner, say $\mathcal{O}_i$, sets the current time frame to be zero and then starts its *FormRingSvc*(0, 0, *mac*($\mathcal{O}_i$), *mac*($\mathcal{O}_i$), 0, M_d) service. We can see that multiple devices can become group owners at the same time. Moreover, if a device cannot be a group owner in a time frame T, it may start its *FormRingSvc* in the next time frame by setting the *est_frame* parameter in its *FormRingSvc* service to be $T+1$. In this scheme, the *est_frame* parameter will be fixed to the time frame number that the *FormRingSvc* service was initiated.

Assume that the group owner $\mathcal{O}_i$ starts its *FormRingSvc* service in time frame T. The operations of $\mathcal{O}_i$ and those client devices in $\mathcal{O}_i$'s group in time frame T are as follows: first, during the slot 0, $\mathcal{O}_i$ accepts client devices and overhears other group owners' *FormRingSvc* services. At this time, $\mathcal{O}_i$ records those *FormRingSvc* services that have the same *est_frame* value. Then, at the end of slot 0 in time frame T, $\mathcal{O}_i$ performs the following two operations:

Deciding client devices' roles $\mathcal{O}_i$ assigns roles to the associated client devices by the following rules. (Let the number of associated client devices be *num_client*.)

- If *num_client* ≥ 2, $\mathcal{O}_i$ designates the client devices with the minimum and maximum MAC addresses to be bgToNext and bgForPrev, respectively.
- If *num_client* $= 1$, $\mathcal{O}_i$ designates the client device and itself to be bgToNext and bgForPrev, respectively.
- If *num_client* $= 0$, $\mathcal{O}_i$ designates itself to be bgToNext and bgForPrev.

After the role assignment, $\mathcal{O}_i$ uses *assignRoleCmd* packets, which carry devices' roles, to the corresponding devices.

Deciding the next group at the end of slot 0, $\mathcal{O}_i$ decides its next group by the following procedures. Assume that several *FormRingSvc* service providers are identified, and then $\mathcal{O}_i$ sorts these *FormRingSvc* service providers and itself according to the MAC addresses in an increasing order. Let the sorted device list be L. $\mathcal{O}_i$ further takes the L as a circular list, and then sets its next group to be the next device in L. After selecting its next group, $\mathcal{O}_i$ sends a *nextGroupCmd*(G_{id}) packet to its bgToNext device, where G_{id} is the group ID of the selected next group.

Next, we describe the operations of $\mathcal{O}_i$'s bgForPrev, bgToNext, and leaf devices at the slot 1 in time frame T.

Operations of bgForPrev device For the bgForPrev device, it starts a *ConnGroupSvc*(G_{id}) service, where $G_{id} = mac(\mathcal{O}_i)$.

Operations of bgToNext device For the bgToNext device, it queries the corresponding *ConnGroupSvc* service, which carries the group ID recorded in the received *nextGroupCmd* packet, and then associates to the service provider.

Operations of leaf devices For a leaf device $\mathcal{D}_l$, it decides whether to start a *EstScSvc*(*mac*($\mathcal{D}_l$)) service by a probability p. If yes, $\mathcal{D}_l$ can accept other leaf devices' association requests, and it can accept at most M_s shortcut connections. Otherwise, $\mathcal{D}_l$ can associate to an *EstScSvc* service provider, which locates in different group.

After the above procedures, a ring that connects those groups established in time frame T is formed. In the following, we describe the procedures of merging the ring formed in time frame T with the existing ring. Assume that a group owner $\mathcal{O}_x^{T'}$ has successfully connected to a next group in the time frame T'. The group owner $\mathcal{O}_x^{T'}$ will start *MergeRingSvc*(T', *mac*($\mathcal{O}_x^{T'}$)) service with the probability p. (The *MergeRingSvc* service will be provided at the same time with $\mathcal{O}_x^{T'}$'s *FormRingSvc* service if any.) Let the group owner $\mathcal{O}_i$ be the group owner that has the smallest MAC address among those group owners formed in time frame T, where $T > T'$. The $\mathcal{O}_i$ will be responsible for merging procedures as follows:

1. $\mathcal{O}_i$ first queries if there exists a *MergeRingSvc* service. If a *MergeRingSvc*$(T', mac(\mathcal{O}_x^{T'}))$ is found, $\mathcal{O}_i$ will take the group formed by $\mathcal{O}_x^{T'}$ as its new next group. Then, $\mathcal{O}_i$ sends a *nextGroupCmd*$(mac(\mathcal{O}_x^{T'}))$ packet to its bgToNext device.
2. When $\mathcal{O}_i$'s bgToNext device $\mathcal{D}_{i,n}$ receives a new *nextGroupCmd* packet, it triggers the following procedures to merge the rings formed in time frames T and T'. (Assume the group owner of $\mathcal{O}_i$'s original next group is $\mathcal{O}_j$.)
   - Operation of $\mathcal{D}_{i,n}$: $\mathcal{D}_{i,n}$ first associates to the bgForPrev device $\mathcal{D}_{x,p}^{T'}$ in $\mathcal{O}_x^{T'}$'s group. If the association procedure fails, $\mathcal{D}_{i,n}$ stops the merging procedure. Otherwise, $\mathcal{D}_{i,n}$ disassociates from its original bgForPrev device in $\mathcal{O}_j$'s group. Then, $\mathcal{D}_{i,n}$ sends $mac(\mathcal{O}_j)$ information to $\mathcal{D}_{x,p}^{T'}$. Finally, $\mathcal{D}_{i,n}$ informs $\mathcal{O}_i$ the association result. [Note that as shown in Fig. 7, at this time, $\mathcal{D}_{x,p}^{T'}$ has two bgToNext client devices; one is the device $\mathcal{D}_{i,n}$, and the other is the bgToNext device, say $\mathcal{D}_{y,n}^{T'}$, of its previous group (formed in time frame T').]
   - Operation of $\mathcal{D}_{x,p}^{T'}$: $\mathcal{D}_{x,p}^{T'}$ sends a *nextGroupCmd* $(mac(\mathcal{O}_j))$ packet to $\mathcal{D}_{y,n}^{T'}$, and then disassociates with device $\mathcal{D}_{y,n}^{T'}$.
   - Operation of $\mathcal{D}_{y,n}^{T'}$: $\mathcal{D}_{y,n}^{T'}$ re-associates to the bgForPrev of the group formed by $\mathcal{O}_j$ (as shown in Fig. 7).

As we can see in Fig. 7, after the above procedures, two rings formed in time frame T and T' will be merged. The ring may be further extended if new rings formed in the upcoming time frames are merged to the original ring. Note that to avoid concurrent merging procedures, a bgForPrev device can connect at most two bgToNext devices at the same time. On the one hand, if $\mathcal{D}_{i,n}$ reports that it fails to associate to the new bgForPrev device, $\mathcal{O}_i$ can know the merging procedure fails. Then, $\mathcal{O}_i$ will restart the merge procedure later. On the other hand, if $\mathcal{D}_{i,n}$ successfully associates to the new bgForPrev device, $\mathcal{O}_i$ sends a *changeEstFrame*(T') to the group owners in the ring it located. When a group owner receives the *changeEstFrame*(T') packet, it will change its *est_frame* parameter to T'. The procedure of sending *changeEstFrame* packets is to make sure that the network can eventually reduce to one ring.

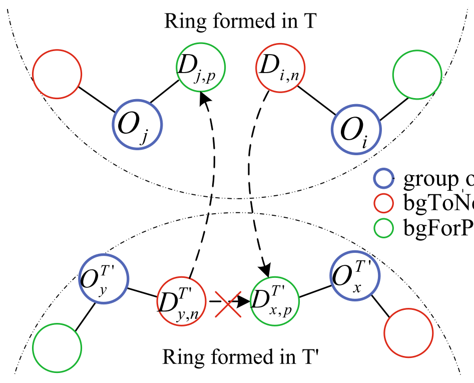


Fig. 7 An example of the merge procedure of two rings

Theorem 5 *The shortest and longest network formation time will be* $2 \times t_s \times O(\log(\frac{1}{p\times(M_d-1)}))$ *and* $2 \times t_s \times O(|V|)$, *respectively.*

Proof In the ring-based scheme, for the best case, we can expect that $|V|p$ devices are able to become group owners in the first time frame. For those $|V|p$ group owners, each of them can connect $M_d - 1$ devices. So, in the first time frame, $|V|p + |V|p(M_d - 1)$ devices can join the network. Then, in the second time frame, each of those $|V|p(M_d - 1)$ devices can further connect $M_d - 1$ devices, and thus there will have $|V|p + |V|p(M_d - 1) + |V|p(M_d - 1)^2$ devices in the network. Assume that we need m time frames to connect all those $|V|$ devices. So, we can have the equation that

$$|V|p + |V|p \times (M_d - 1) + .. + (|V|p \times (M_d - 1))^m \geq |V|.$$

According to the above equation, we can find a $m' \geq m$ such that $(|V|p \times (M_d - 1))^{m'} \leq |V|$, and thus $m \leq m' \leq O(\log(\frac{1}{p\times(M_d-1)}))$. As a result, we can say that the shortest number of time frames to form the network will be less than $O(\log(\frac{1}{p\times(M_d-1)}))$, which implies the expected network formation time will be $2 \times t_s \times O(\log(\frac{1}{p\times(M_d-1)}))$.

Moreover, in the worst case, the network formation needs $O(|V|)$ time frames, and thus network formation time will be $2 \times t_s \times O(|V|)$. □

4.2 Disseminating packets

In the following, we describe the procedures of disseminating packets. Assume that a device $\mathcal{D}_i$ generates or receives a data packet for network devices, it will perform the following procedures.

1. If the current slot is 0, there are two cases. First, if $\mathcal{D}_i$ is a client device, $\mathcal{D}_i$ sends this packet to its group owner $go(\mathcal{D}_i)$. Second, if $\mathcal{D}_i$ is a group owner, $\mathcal{D}_i$ sends the data packet to all its client devices.

2. If the current slot is 1, there are three cases. First, if $\mathcal{D}_i$ is a bgForPrev device, $\mathcal{D}_i$ sends the data packet to the connected bgToNext device. Second, if $\mathcal{D}_i$ is a bgToNext device, $\mathcal{D}_i$ sends the data packet to the connected bgForPrev device. Third, if $\mathcal{D}_i$ is a leaf device and has connected to some shortcuts, it sends the packet to the peers on the shortcuts. When the peer leaf devices receives the packet, they can further relay the packet to the other receivers on the shortcuts that they have connected.

4.3 Network repair

Again, the network repair can also be divided into *regular repair* and *instant repair*. First, the regular repair tries to let each group to contain M_d devices. Assume that a group owner $\mathcal{O}_m$ realizes that it has the maximum MAC address among nearby group owners. In our scheme, $\mathcal{O}_m$ will trigger the regular repair. The $\mathcal{O}_m$ first sorts its leaf devices according to their MAC addresses by an increasing order, and then puts these leaf devices to a list L. After putting leaf devices, $\mathcal{O}_m$ puts its bgToNext and then bgForPrev devices to the tail of list L (if the bgToNext or bgForPrev device is not $\mathcal{O}_m$). Then, $\mathcal{O}_m$ further puts itself to the end of the list L. After determining L, $\mathcal{O}_m$ queries *FormRingSvc* services, and then selects a group, which satisfies the condition that the group has less than M_d devices and has the least number of client devices. Assume that the selected group owner is $\mathcal{O}_i$ and the number of devices in $\mathcal{O}_m$'s and $\mathcal{O}_i$'s group are n_m and n_i, respectively. The $\mathcal{O}_m$ will then demand k devices in its group to re-associate to $\mathcal{D}_i$ according to the following two cases:

- If $n_m \geq (M_d - n_i)$, $\mathcal{O}_m$ sets $k = (M_d - n_i)$, i.e., $(M_d - n_i)$ of $\mathcal{O}_m$'s client devices will re-associate to $\mathcal{O}_i$.
- If $n_m < (M_d - n_i)$, $\mathcal{O}_m$ sets $k = n_m$, i.e., all devices in the group of $\mathcal{O}_m$ will re-associate to $\mathcal{O}_i$.

After determining the k value, $\mathcal{O}_m$ sends *changeGoCmd*(*mac*($\mathcal{O}_i$)) packet to the first k devices in L. For a device that receives the *changeGoCmd* packet, it will re-associate to $\mathcal{O}_i$ in the next time frame. In order to preserve the ring, there are some additional procedures for the three cases below. (Assume $\mathcal{O}_j$ is the group owner of $\mathcal{O}_m$'s next group.)

- Case 1: only $\mathcal{O}_m$ and bgForPrev device are left in $\mathcal{O}_m$'s group. In this case, $\mathcal{O}_m$ will be the bgToNext, and then associate to the bgForPrev device in $\mathcal{O}_j$'s group.
- Case 2: only $\mathcal{O}_m$ is left in $\mathcal{O}_m$'s group. In this case, the original bgForPrev device in $\mathcal{O}_m$'s group sends a *changeGoCmd*(*mac*($\mathcal{O}_m$)) packet to the connected bgToNext device in the previous group at slot 1. The bgToNext device that receives the *changeGoCmd* will then re-associate to $\mathcal{O}_m$ in the next time frame.
- Case 3: no device is left in $\mathcal{O}_m$'s group. In this time, the original bgForPrev device in $\mathcal{O}_m$'s group sends a *changeGoCmd*(*mac*($\mathcal{O}_j$)) packet to the connected bgToNext device in the previous group at slot 1. The bgToNext device that receives the *changeGoCmd* will re-associate to the bgForPrev device in group $\mathcal{O}_j$ in the next time frame.

Note that after the above procedures, if the group of $\mathcal{O}_m$ still exists, $\mathcal{O}_m$ will restart the regular repair after a pre-defined period of time.

Next, we introduce the instant repair procedure. The instant repair starts when there is a device leaves the network. In this scheme, the group owner and bgForPrev periodically sends *aliveCmd* packet to check if the devices that associate to them are still alive. When a client device receives an *aliveCmd* packet, it will reply an *aliveAckCmd*. A device will be considered leaving the network if it has no response for T_{lost} time. There are three possible scenarios in the instant repair.

- The group owner $\mathcal{O}_i$ realizes one of its client device $\mathcal{D}_{lost}$ leaves the network: $\mathcal{O}_i$ performs the following operations according to $\mathcal{D}_{lost}$'s roles. First, if $\mathcal{D}_{lost}$ is leaf device, $\mathcal{O}_i$ does no operation. Second, if $\mathcal{D}_{lost}$ is a bgToNext or bgForPrev, there are three cases:
  - If $\mathcal{O}_i$ has other leaf client devices, $\mathcal{O}_i$ designates one leaf devices to be the role of $\mathcal{D}_{lost}$.
  - If $\mathcal{O}_i$ has only one client device (i.e., currently, the client device must be a bgForPrev or bgToNext device), $\mathcal{O}_i$ demands this client device to be bgToNext and itself be bgForPrev.
  - If $\mathcal{O}_i$ has no client device, $\mathcal{O}_i$ will then play the roles of bgToNext and bgForPrev at the same time.

  Note that the procedures of assigning roles and informing related parameters are the same as the procedures when forming the network.
- The group owner $\mathcal{O}_{lost}$ leaves and the group has other members: In this scenario, the bgToNext device $\mathcal{D}_i$ will start *FormRingSvc*(-, -, *mac*($\mathcal{D}_i$), *mac*($\mathcal{O}_{lost}$), 0, M_d) service, and other client devices will then re-associate to $\mathcal{D}_i$. Then, $\mathcal{D}_i$ will further transmit *changeGidCmd*(*mac*($\mathcal{D}_i$)) packet to its client devices and choose a new bgToNext according to the rule when forming the network. After changing group ID, the bgForPrev and bgToNext devices also use *changeGidCmd* packet to notify the corresponding previous and next groups.
- The group owner $\mathcal{O}_{lost}$ leaves and the group has no other member: In this scenario, both of the next group's bgForPrev device $\mathcal{D}_i$ and the previous group's bgToNext device $\mathcal{D}_j$ can realize the group of $\mathcal{O}_{lost}$ is gone. At this time, $\mathcal{D}_i$ starts *ConnGroupSvc*(*mac*($\mathcal{D}_{lost}$)) service, and device $\mathcal{D}_j$ queries the corresponding

ConnGroupSvc service. After $\mathcal{D}_j$ re-associates to $\mathcal{D}_i$, $\mathcal{D}_i$ sends *changeGidCmd*(*mac*(*go*($\mathcal{D}_i$))) to notify $\mathcal{D}_j$ to change its next group to the group of $go(\mathcal{D}_i)$.

We can see that the last scenario dominates the instant network repair time. The repair time will be $T_e + T_r$, where T_e and T_r represent the time for the device D_j to detect *ConnGroupSvc* service and the time for the bgToNext device $\mathcal{D}_j$ to re-associate to $go(\mathcal{D}_i)$'s group, respectively.

Theorem 6 *The shortest data dissemination time will be* $2 \times t_s \times O(1 + (\lceil(\frac{|V|}{M_d \times M_s \times (M_d - 3) + 1})/2\rceil)$.

Proof According to the designed regular network repair, the network will have $\lceil\frac{|V|}{M_d}\rceil$ groups. Recall that a leaf node can accept M_s shortcuts. For a group, there are at most $(M_d - 3)$ leaf devices, and thus the most number of shortcuts in a group will be $M_s \times (M_d - 3)$. For the best case, these shortcuts are connected to the groups that are evenly distributed in the ring. In other words, the groups in the ring network can be divided into $M_s \times (M_d - 3) + 1$ supergroups. For a supergroup, there will have $\lceil\frac{|V|}{M_d \times M_s \times (M_d - 3) + 1}\rceil$ groups. In a supergroup, we assume that a packet will transmit along the ring topology. So, when a group owner in the supergroup receives a packet, it needs at most $T_n = \lceil(\frac{|V|}{M_d \times M_s \times (M_d - 3) + 1})/2\rceil$ time frames to propagate the packet in the supergroup. Moreover, by our scheme, the group that the packet initiated needs one time frame to process this packet. In the best case, the minimum number of time frames for data dissemination will be $O(1 + T_n)$. As a result, the shortest data dissemination time will be $2 \times t_s \times O(1 + T_n)$. $\square$

Theorem 7 *The longest data dissemination time will be* $2 \times t_s \times O(1 + \left\lceil(\frac{|V|}{M_d})/2\right\rceil)$.

Proof Again, the network has $\lceil\frac{|V|}{M_d}\rceil$ groups. In the worst case, there is no shortcut in the network. So, a packet has to propagate along the ring. By our scheme, the group that the packet initiated needs one time frame to process this packet. As a result, the needed number of time frames to propagate the packet will be $O(1 + \lceil(\frac{|V|}{M_d})/2\rceil)$, which implies the longest data dissemination time will be $2 \times t_s \times O(1 + \left\lceil(\frac{|V|}{M_d})/2\right\rceil)$. $\square$

5 Performance evaluations

In this section, we show performance evaluations of our schemes. First, we show some simulation results, which measure network formation time, data dissemination time, and network repair time based on a simulator upon IEEE 802.11 MAC layer. Second, we further implement the proposed schemes on commercial Wi-Fi P2P platforms to observe the effectiveness of our schemes.

5.1 Simulation results

In this paper, we implement an event-based Wi-Fi P2P network simulator (by JAVA language). In our simulations, each device is an individual thread. Devices perform the IEEE 802.11 CSMA/CA mechanism [2] to avoid collisions. A device will be triggered to perform the designed procedures in Sects. 3 and 4 when being enabled or when receiving packets. When receiving a packet, a device has to reply an acknowledgement packet. If the sender does not receive the corresponding acknowledgement packet from the receiver, the sender will retransmit until the packet reaches the receiver successfully. The transmission ranges of devices are fixed to 100 m, and the network is a circular region with radius 100 m. Other MAC and system parameters are listed in Table 1. In our simulation, each data point is the average of 100 thousand trials conducted on 100 different randomly generated networks. Note that since our designs are upon IEEE 802.11 MAC layer, in our simulator, we did not simulate physical layer functionalities and characteristics such as power control, rate adaptation, path loss model, and so on.

We compare the proposed schemes against two star-topology-based schemes, named Mstar-1 and Mstar-n. Both of Mstar-1 and Mstar-n schemes apply unicast transmissions to disseminate data packets, but they do not apply the slot concept used in our schemes. In Mstar-1, there is only one group owner. All devices associate to the group owner. In a Mstar-n network, there are n group owners. When a device enters the networks, it first checks if there exists n group owners. If not, it can be a group owner; otherwise, it will associate to an existing one. Moreover, in Mstar-n, when a device generates a data packet, it transmits the packet to its group owner. The

Table 1 Simulation parameters

Parameter	Values
Wi-Fi data rate	250 Mbps
Number of channels	4 @2.4 GHz
CTWindow [2]	$9 \times (2^m - 1)\mu s$, $m = 4 \ldots 10$
Time unit (TU) [2]	1024 µs
Slot size t_s	0.5 s
Detect failure interval T_{lost}	Four slots
Receiving power consumption [26]	472 mW
Transmitting power consumption [26]	553 mW
Probability p (ring)	0.6
Maximum shortcut M_s (ring)	M_d

group owner first unicasts this packet to each of its group members, and then the group owner randomly designates a group member to reconnect to another randomly selected group. The selected group member is responsible for distributing the data packet to another group.

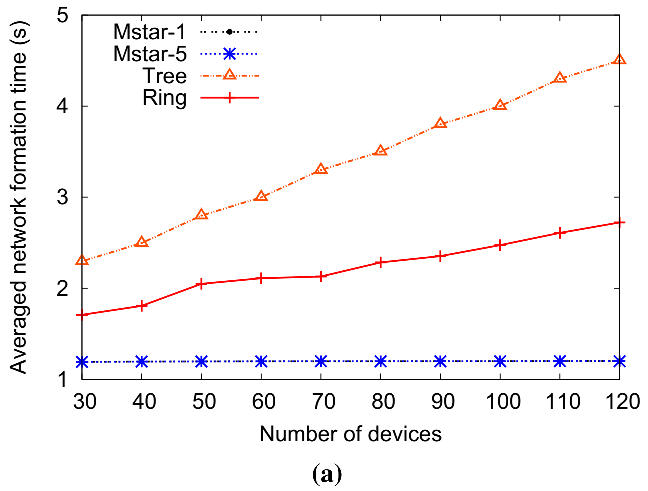
In the following, we compare the proposed tree data dissemination scheme and ring data dissemination scheme against Mstar-1 and Mstar-5 schemes on network formation time, data dissemination time, energy consumption, and the needed time for instant repair. Each simulation result is derived by averaging 1000 runs.

5.1.1 Network formation time

Assume that all devices enter the network at the same time. The network formation time is measured by the time instant that a device becomes the first network group owner to the time instant that the last device joins the network. Figure 8(a) indicates the results when the number of network devices are varied. (For our schemes, we fix $M_d = 5$). From the results, we can see that the network formation times of Mstar-1 and Mstar-5 are lower than our schemes. This is because that Mstar-1 and Mstar-5 and do not apply the slot concept used in our schemes. So, a device can join the network immediately when it finds a group owner. The tree-based scheme will have the longest network formation time. This is because that the network formation procedure starts from a single network root, and other devices can be group owners after they associate to existing group owners. As a result, the network formation time will be lengthened. On the other hand, in the ring-based scheme, multiple group owners will be formed when the network starts, and thus the network formation time can be reduced. Next, we fix the number of network device to be 120, and observe the effects on different M_d values to our schemes. Figure 8(b) indicates the results. As we can expect, when the M_d becomes larger, the network formation time can be lower. The network formation times of the ring-based scheme can still be lower than that of the tree-based scheme.

5.1.2 Data dissemination time and energy consumption

In the following, we first compare the data dissemination time. In each trial of the simulations, a network device is randomly selected as a packet source. The data dissemination time is measured from the time instant that a packet is generated to the time that the last device received this packet. We set $M_d = 5$ and vary the number of network devices. Figure 9(a) shows the results when the packet size is fixed to 1MB. We can see that when there are less devices, the Mstar-1 can perform the best. But, the data dissemination times of Mstar-1 will be longer when there are more devices in the network. This is because that in a Mstar-1 network, all devices operate on the same channel, and the group owner has to send packets to devices one-by-one. So, when there are more devices, the group owner needs more time to distribute data packets. The Mstar-5 may have higher data dissemination times than Mstar-1 because that the generated packets need some times to reach devices in different groups. In this simulation, the ring-based scheme can perform better than that of the tree-based scheme in all cases. This is because the ring-based scheme uses shortcuts to accelerate data spreading. However, we can see that in this simulation, our schemes need more data dissemination times than that of Mstar-1 and Mstar-5 schemes. The main reason is that our schemes adopt the slot structure. So, after a packet is disseminated to all members in a group, this packet will stay at the group



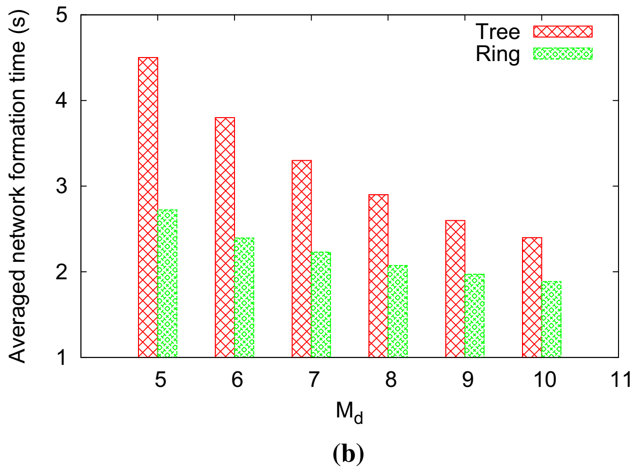


Fig. 8 Simulation results on network formation time when varying **a** number of network devices and **b** M_d values

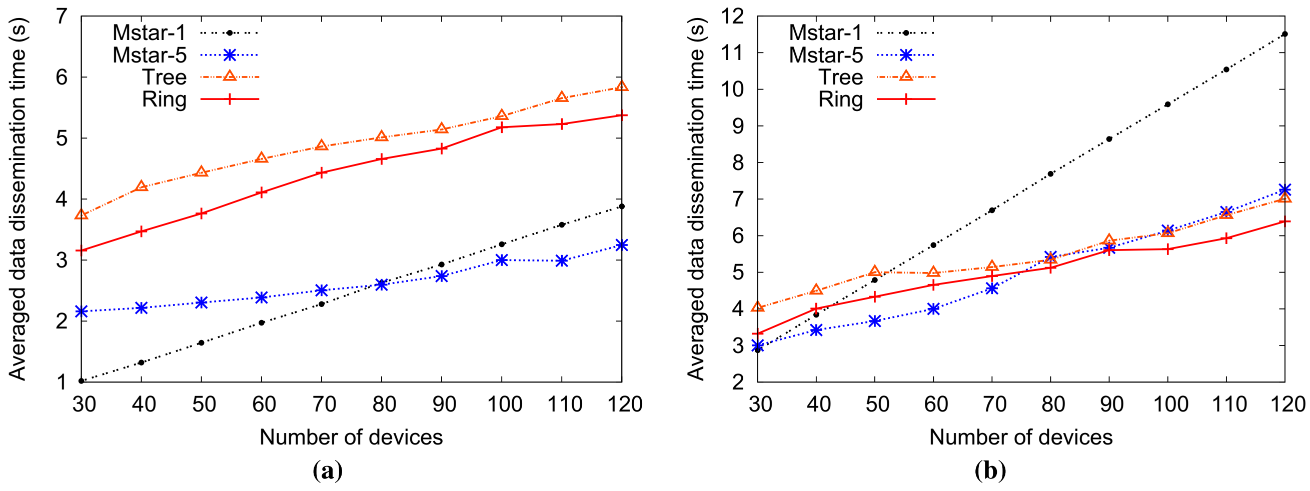


Fig. 9 Simulation results on data dissemination time when data packet sizes are **a** 1 MB and **b** 3 MB

owner until the next slot or next time frame to reach another group. Moreover, Fig. 9(b) further shows the data dissemination time when the data packet size is fixed to 3MB. Again, the ring-based scheme can perform better than that of the tree-based scheme in all cases. When the packet size is larger, the Mstar-1 will have the worst performance. The Mstar-5 can perform better than Mstar-1 because that the group owners in Mstar-5 can operate on different channels to send data packets in parallel. The ring-based scheme can perform the best when there are more devices in the network.

Next, we show the simulation results on maximum energy consumption (in nJ) among network devices for data communications. (The used power consumption parameters for transmitting and receiving in our simulations are listed in Table 1.) Figure 10(a, b) indicate the energy consumption results of the simulations in Fig. 9(a, b), respectively. We can see that the proposed schemes can outperform the Mstar-1 and Mstar-5 in all cases. In the Mstar-1 and Mstar-5 schemes, the group owners will have higher loads than other devices, and thus the group owners will have higher energy consumption than other devices. These results indicate that our design can effectively share the load on disseminating data to multiple group owners. Moreover, in the following, we observe the effect on different M_d values. We fix the number of network devices to be 120 and the data packet size to 1 MB. Figure 11 indicates the results. Recall that the ring-based scheme can utilize shortcuts to accelerate data dissemination. Since we set $M_s = M_d$ in our simulations, when M_d becomes higher, a leaf device in the ring-based scheme can have more shortcuts, and thus the data dissemination time can be further reduced. Figure 11 also indicates the results on energy consumption. We can see that when M_d becomes larger, the consumed energy does not increase greatly. This result further demonstrates that the proposed schemes can

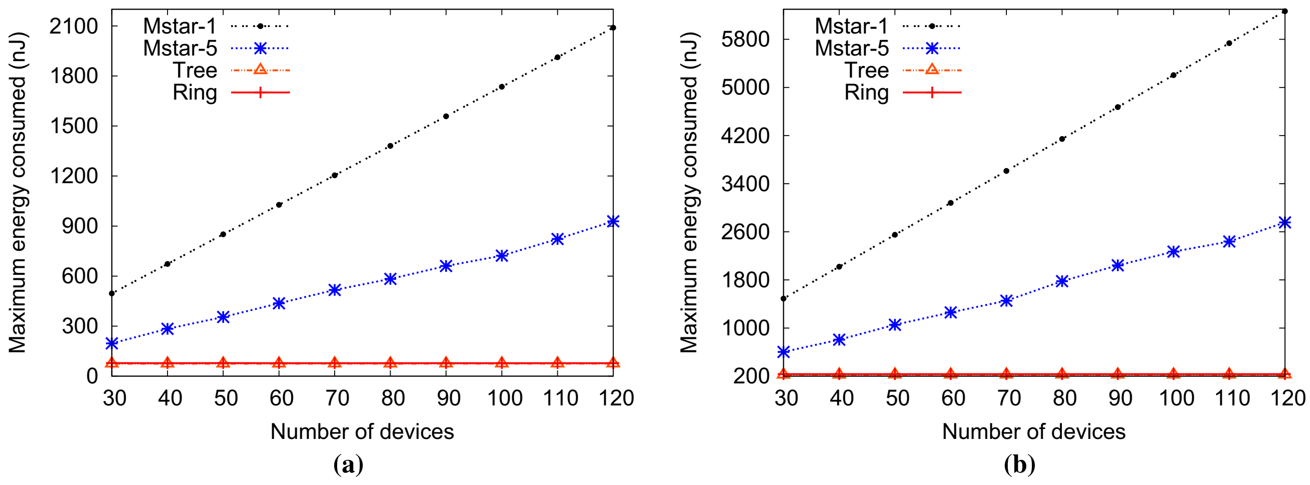


Fig. 10 Simulation results on power consumption when data packet sizes are **a** 1 MB and **b** 3 MB

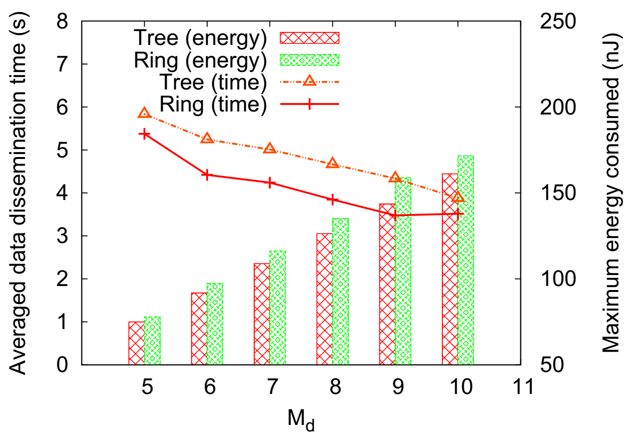


Fig. 11 Simulation results on data dissemination time and power consumption when propagating 1 MB data packets to network devices with varied M_d values

effectively balance loads on disseminating data for network devices.

5.1.3 *Instant repair time*

In the following, we simulate the needed time for instant repair. We assume that 1% to 10% devices may leave the network suddenly. The repair time is recorded from the time instant that a device leaves the network to the time instant that the network is recovered. In the simulations, we fix the number of network devices to be 120 and $M_d = 5$. We do not simulate the Mstar-1 and Mstar-5 because that if a client device leaves the network, group owner does nothing. However, when a group owner leaves the Mstar-1 or Mstar-5 network, the repair time will be equal to the network formation time. Figure 12 shows the simulation results. From Fig. 12, we can see that the repair time of the ring-based scheme can be lower than the tree-based scheme. This is because that when instant repair, the tree-based scheme asks an agent device to replace the left group owner (which needs at least one slot time). On the other hand, in the ring-based scheme, most devices are leaf devices, and a group owner does nothing when a leaf device leaves the network. But, when a bgForPrev or bgToNext device leaves the network, the ring-based scheme may need more times to ask a leaf device to replace the left device. However, in average, the repair time of the ring-based scheme can still be shorter than that of the tree-based scheme.

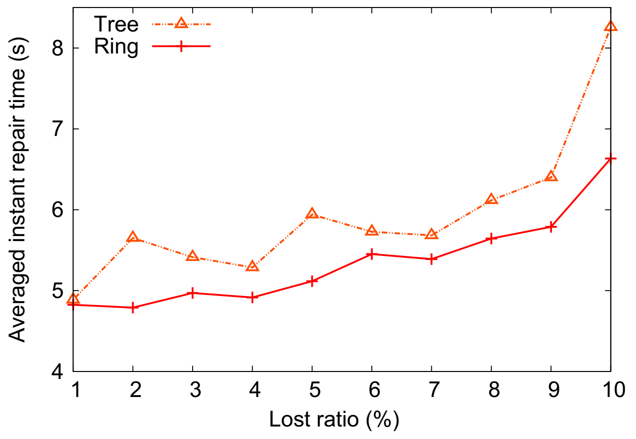


Fig. 12 Simulation results on instant repair

5.2 Prototyping implementation

5.2.1 *Implementation details*

In this work, we use the Android Wi-Fi P2P API [27] to implement our designs. In the following, we briefly describe some details of our implementation. According to the API, a device utilizes the *WifiP2pManager* class to manage its network connections. Before operation, the device needs to create an event handler (a.k.a, Android broadcast receiver) to monitor the states of the device, the connections, the network, and the device discovery procedure. When the event handler realizes a state change, the device can perform the corresponding reactions. Moreover, when a device decides to become a group owner, it utilizes the *createGroup* function to configure the lower layer of the Wi-Fi P2P protocol stack. Then, the device uses the *WifiP2pDnsSdServiceInfo* class to specify the designed services' information, and calls the *addLocalService* function to start its services. Furthermore, before discovering services, a device needs to specify the corresponding reactions by the *DnsSdTxtRecordListener* interface, and uses the *discoverServices* function to discover nearby services. Then, if the device finds a group owner, it can use the *connect* function to associate with that group.

According to the Android Wi-Fi P2P API, a device uses socket to transmit data packets within a Wi-Fi P2P group. Each group owner creates a *ServerSocket* object and designates a communication port. Then, the group owner can initialize a thread to wait for connections. On the other hand, a client device utilizes the *Socket.connect* function to specify its group owner's IP address and port number, and then can send packets to the group owner. However, there is an implementation issue. By the Android Wi-Fi P2P API, a group owner does not know its clients' IP address, and thus the bi-direction communication cannot be achieved. To resolve this problem, in our implementation, the group owner will further analyze the received IP packets from its client devices. From the IP packet header, the group owner extracts the sender's IP address. Then, the group owner uses a table to record its client devices' IP and MAC addresses.

Table 2 Service discovery time on finding a specific service from a specific service provider (in ms)

Run	1 service	2 services	3 services	4 services
1	11,142	7504	2521	3419
2	6020	2014	1662	1838
3	6012	7001	2548	1847
4	7121	15,246	2573	1711
5	2118	1804	271	4181
6	2103	2096	4682	4221
7	2087	1999	1725	6948
8	4694	2499	1832	5102
9	6701	6550	4038	2723
10	4486	1986	4067	2007
11	2838	2025	1710	2838
12	4611	6989	12,580	4502
13	3651	3783	1871	6896
14	3130	3944	2503	11,066
15	10,626	5044	2561	3503
16	66	8341	2975	6125
17	1923	1796	3025	1896
18	2299	1814	1803	2437
19	5435	1841	4874	5271
20	1859	1822	6514	7075
Averaged discovery time (ms)	4446.1	4304.9	3316.75	4280.3

Moreover, a local synchronization mechanism is implemented, and the details are as follows: When a device associates to a group owner, this device will keep tracking its group owner's periodical *SYNC* packets. For a group owner $\mathcal{O}$, a *SYNC* packet records the numbers of time stamps, say $rem(\mathcal{O})$, to the next start timing of $\mathcal{O}$'s slot. So, assume that a device $\mathcal{D}$ is a client of the group owner $\mathcal{O}$, and $\mathcal{D}$ is a group owner of another group. Device $\mathcal{D}$ can know that its next working slot will be started at time $(t + rem(\mathcal{O}) - t_s)$, where t is the time that $\mathcal{D}$ receives the last *SYNC* packet of $\mathcal{O}$, and t_s is the slot size of the system.

5.2.2 Discussions on slot size

In our current implementation, we set the slot size to be 15 s. We have to designate a large slot because of the following two reasons: (1) the Android Wi-Fi P2P API cannot keep the information of two group owners at the same time. According to the Android Wi-Fi P2P API, when a device associates to a group owner, the information of that group owner will be recorded in the device's local database. But, when the device associates to the another group owner, the information of the original group owner will be erased. So, the device has to re-perform the service discovery and then re-connect to the group owner again every time when slot changes. (2) By the Android Wi-Fi P2P API, a service discovery procedure may take a long time to accomplish. We did an experiment with the configuration that there are eight devices and these devices provide 1–4 kinds of services. In the experiment, a device is asked to find a specific service from a specific service provider. Table 2 indicates the experiment results on service discovery time (in ms) for 20 runs. From the results, we can see that service discovery times are varied, and the longest discovery time is 15.2 s.

To resolve the drawback of setting a large slot size, we have to implement the following two features. First, we need to modify the kernel of the Android Wi-Fi P2P API and then root the device to keep the multiple group owners' information. Second, we have to shorten the service discovery time. In reference [28], the authors propose a fast device discovery scheme, which can successfully find a specific service within 1 s. Also, the Wi-Fi standard group is working on designing fast generic advertisement service (GAS) protocol, which can help to accelerate the service discovery time. Some patents (e.g. [29, 30]) also propose schemes to achieve fast service discovery. The implementation of the above two features will be taken as our future work.

Note that because of the long service discovery time, we do not implement the designed instant repair schemes in our current implementation (since it can be expected that the instant repair time will be long). But, we can still rely on the designed regular repair mechanisms to maintain the network.

5.2.3 Demonstrations

We show our implementation result by a video clip. In our demonstrations, we use 9 Android smartphones (which are with several brands and models). The smartphones use the proposed tree-based and ring-based scheme to distribute photos. For the demonstration, please refer to https://www.youtube.com/watch?v=0brPJ0tKP6E.

6 Conclusions

In this paper, we aim to achieve efficient data dissemination in Wi-Fi P2P networks by unicasting among multiple Wi-Fi P2P groups. We model the target network scenario by a WPDD problem and prove the WPDD problem is NP-complete. Instead of using one Wi-Fi P2P group to connect all network devices, we propose to form multiple groups in the network. We then propose a tree-based data dissemination scheme and a ring-based data dissemination scheme to solve the problem. The proposed schemes contain related network formation, data dissemination, and network repair procedures. The designed procedures can be compatible with the Wi-Fi P2P specification. The simulation results show that compared to other schemes, the proposed schemes can effectively balance devices' energy and reduce data dissemination time. The prototyping results further indicate that the proposed schemes can effectively operate on commercial smartphones. In the future, we are going to apply the proposed designs on disaster management applications.

References

1. Wi-Fi peer-to-peer (P2P) technical specification, v.1.5. (2014). Wi-Fi Alliance Technical Committee P2P Task Group.
2. IEEE Std. 802.11. (2012). Wireless LAN medium access control (MAC) and physical layer (PHY) specifications.
3. Casetti, C., Chiasserini, C. F., Pelle, L. C., Valle, C. D., Duan, Y., & Giaccone, P. (2015). Content-centric routing in Wi-Fi direct multi-group networks. In *Proceedings of IEEE international symposium on a world of wireless, mobile and multimedia networks (WoWMoM)*.
4. Drabkin, V., Friedman, R., Kliot, G., & Segal, M. (2011). On reliable dissemination in wireless ad hoc networks. *IEEE Transactions on Dependable and Secure Computing*, *8*(6), 866–882.
5. Hsu, C.-S., Tseng, Y.-C., & Sheu, J.-P. (2007). An efficient reliable broadcasting protocol for wireless mobile ad hoc networks. *Ad Hoc Networks*, *5*(3), 299–312.
6. Lou, W., & Wu, J. (2007). Toward broadcast reliability in mobile ad hoc networks with double coverage. *IEEE Transactions on Mobile Computing*, *6*(2), 148–163.
7. Park, S., & Yoo, S.-M. (2013). An efficient reliable one-hop broadcast in mobile ad hoc networks. *Ad Hoc Networks*, *11*(1), 19–28.
8. SuperBeam. http://superbe.am/
9. Wi-Fi Direct+. https://play.google.com/store/apps/details?id=com.netcompss_gh.wifidirect
10. Li, Z., Xie, G., Hwang, K., & Li., Z. (2011). Churn-resilient protocol for massive data dissemination in p2p networks. *IEEE Transactions on Parallel and Distributed Systems*, *22*(8), 1342–1349.
11. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D. R., Kaashoek, M. F., Dabek, F., et al. (2003). Chord: A scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, *11*(1), 17–32.
12. Yang, M., & Yang, Y. (2010). An efficient hybrid peer-to-peer system for distributed data sharing. *IEEE Transactions on Computers*, *59*(9), 1158–1171.
13. Li, Z., Wang, C., Yang, S., Jiang, C., & Stojmenovic, I. (2014). Improving data forwarding in mobile social networks with infrastructure support: A space-crossing community approach. In *Proceedings of IEEE INFOCOM*.
14. Lu, Z., Sun, X., Wen, Y., & Cao, G. (2014). Skeleton construction in mobile social networks: Algorithms and applications. In *Proceedings of IEEE international conference on sensing, communication, and networking (SECON)*.
15. Wang, X., Chen, M., Han, Z., Wu, D.O., & Kwon, T.T. (2014). TOSS: Traffic offloading by social network service-based opportunistic sharing in mobile social networks. In *Proceedings of IEEE INFOCOM*.
16. Jiang, C., Zhang, Y., Yuan, J., Ren, Y., & Han, Z. (2016). Cooperative WiFi management: Nash bargaining solution and implementation. In *Proceedings of IEEE wireless communications and networking conference (WCNC)*.
17. Zhang, Y., Jiang, C., Han, Z., Yu, S., & Yuan, J. (2016). Interference-aware coordinated power allocation in autonomous Wi-Fi environment. *IEEE Access*, *4*, 3489–3500.
18. Zhang, Y., Jiang, C., Wang, Y., Yuan, J., & Cao, J. (2015). United channel assignments in residential environments. In *Proceedings of IEEE global telecommunications conference (Globecom)*.
19. Duong, T. N., Dinh, N.-T., & Kim, Y. (2012). Content sharing using P2PSIP protocol in Wi-Fi direct networks. In *Proceedings of IEEE international conference on communications and electronics (ICCE)*.
20. Hoang, L. V., & Ogawa, H. (2014). A platform for building ad hoc social networks based on Wi-Fi direct. In *Proceedings of IEEE global conference on consumer electronics (GCCE)*
21. Toledano, E., Sawada, D., Lippman, A., Holtzman, H., & Casalegno, F. (2013). CoCam: A collaborative content sharing framework based on opportunistic P2P networking. In *Proceedings of IEEE consumer communications and networking conference (CCNC)*.
22. Yun, M., Kim, D., Lee, H.-S., Lee, J. (2012). Silent broadcast: Experience of connectionless messaging using Wi-Fi P2P. In *Proceedings of IEEE international conference on information science and digital content technology (ICIDT)*.
23. Jung, W.-S., Ahn, H., & Ko, Y.-B. (2014). Designing content-centric multi-hop networking over Wi-Fi direct on smartphones. In *Proceedings of IEEE wireless communications and networking conference (WCNC)*.
24. Yao, C., Zhang, H., & Song, L. (2015). Demo: WiFi multihop: Implementing device-to-device local area networks by android smartphones. In *Proceedings of ACM international symposium on mobile ad hoc networking and computing (MobiHoc)*.

25. Dinneen, M. J. (1994). The complexity of broadcasting in bounded-degree networks. Combinatorics report LACES-[05C-94-31], Los Alamos National Laboratory.
26. Friedman, R., Kogan, A., & Krivolapov, Y. (2013). On power and throughput tradeoffs of WiFi and bluetooth in smartphones. *IEEE Transactions on Mobile Computing*, *12*(7), 1363–1376.
27. Wi-fi peer-to-peer, Android developers. http://developer.android.com/guide/topics/connectivity/wifip2p.html
28. Sun, W., Yang, C., Jin, S., & Choi, S. (2016). Listen channel randomization for faster Wi-Fi direct device discovery. In *Proceedings of IEEE INFOCOM*.
29. Canpolat, N., & Gupta, V. G. (2015). Broadcast based discovery of Wi-Fi networks, devices and services. US patent No. 9125143B2.
30. Seok, Y., You, H., Lee, J., & Kim, E. (2014). Method and apparatus for finding a neighbor in a wireless communication system. US patent No. 20140092779A1.

Meng-Shiuan Pan received his Ph.D. degree from the National Chiao Tung University, Taiwan. After obtaining his Ph.D. degree, he worked in HTMM corporation and dedicated in designing and implementing 3GPP layer 2/3 protocol stacks. He is now an associate professor of Department of computer science and information engineering, Tamkang University, Taiwan. He also serves as an associate editor of International Journal of Distributed Sensor Networks (IJDSN). His research interests include mobile computing, wireless communication, and Internet of Things.

Yen-Pei Lin received his B.S. and M.S. degrees from Tamkang University, Taipei, Taiwan. His research interests include wireless network and protocol design.