GOCRGO and GOGO: Two Minimal Communication Topologies for WiFi-Direct Multi-group Networking

António Teófilo
ADEETC, ISEL-IPL , Portugal *
NOVA Laboratory for Computer Science and Informatics,
DI-FCT-UNL, Portugal †
ateofilo@deetc.isel.ipl.pt

Diogo Remédios
ADEETC, ISEL-IPL, Portugal *
NOVA Laboratory for Computer Science and Informatics,
DI-FCT-UNL, Portugal †
dremedios@deetc.isel.ipl.pt

João M. Lourenço
NOVA Laboratory for Computer Science and Informatics,
DI-FCT-UNL, Portugal †
joao.lourenco@fct.unl.pt

Hervé Paulino
NOVA Laboratory for Computer Science and Informatics,
DI-FCT-UNL, Portugal †
herve.paulino@fct.unl.pt

ABSTRACT

Although mobile devices can collaborate and interact when connected by a communication infrastructure, such interactions may also be greatly desired or even highly necessary when such infrastructures are not available or cannot be used, like in crowded spaces, disaster situations, and remote areas, or when there is no trust in the existing infrastructures. A major requirement to support such autonomous collaborative systems on top of mobile devices is to build a communication network to interconnect them all. In this quest, Wi-Fi Direct (WFD) stands out as a promising technology to offer infrastructure-less WiFi networking to off-the-shelf devices. However, the WFD standard only addresses communications inside one group of devices, and current WFD inter-group communication solutions have several limitations, as they must contend with intermittent connections, slow communication (*broadcasts* and/or *multicasts*), and/or a high number of nodes to interconnect groups. In this paper, we aim to overcome those limitations by proposing two novel topologies, named GOCRGO and GOGO, which use permanent radio connections, can rely on either UDP or TCP communication, and require the minimum number of nodes necessary to interconnect WFD groups. Both topologies present advantages over each other, in different scenarios, and thus can be used together in a complementary way.

CCS CONCEPTS

•Networks →Mobile networks; Wireless access networks; *Network design and planning algorithms;* **•Human-centered computing →Mobile phones;**

KEYWORDS

Mobile Networking, Device-to-Device Communication, Autonomous Networks, Wi-Fi Direct, Multi-Hop Wi-Fi Direct Networks, Android

1 INTRODUCTION

Mobile devices have become ubiquitous, largely surpassing laptop computers in number and as data generators and consumers. The hardware capabilities of current mobile phones, particularly CPU, memory and networking/communications, enable them to interact with other devices as clients, as well as small servers. As a result, mobile devices have become appropriate for participating in collaborative scenarios, even in cases where there are no communication infrastructures, such as isolated areas and disaster situations; in crowded venues where public networks become saturated, such as sports and cultural events [10, 17]; or when it is not appropriate to use public networks, like political manifestations in countries that censure network traffic or in the event of governmental actions in foreign countries. In such cases, collaborative systems based exclusively on mobile devices may not only help to spread information (like messages and photos) among users, but also to share resources, acting as a mobile cloud for data and computations [11]. Real-world examples of applications that can be used in such scenarios include: the USA military CBMEN (Content-Based Mobile Edge Networking) program[1], aiming at "*enabling efficient, transparent distribution of content in mobile ad hoc network environments*" that will allow soldiers to share information in war scenarios; a photo face recognition system to look for missing persons in crowded scenarios; a system enabling the sharing of videos in sporting events that were recorded by other in-place fans; and a system supporting the exchange of emergency messages in disaster scenarios.

However, the feasibility of such collaborative mobile applications is bound to the ability to form networks able to interconnecting off-the-shelf mobile devices. In such context, and because devices currently do not support WiFi *ad hoc* mode, Bluetooth and Wi-Fi Direct (WFD) are the two most current widespread device-to-device communication technologies. The Bluetooth technology is known to be used to build mesh networks of mobile devices (Scatternets) [21], however its limitations in communication speed and range make this technology impractical for most uses. WFD [1] is a WiFi-based

[1] http://www.darpa.mil/program/content-based-mobile-edge-networking

* ISEL - Instituto Superior de Engenharia de Lisboa, IPL - Instituto Politécnico de Lisboa
† FCT - Faculdade de Ciências e Tecnologia, UNL - Universidade NOVA de Lisboa

This work was partially funded by FCT-MEC in the context of the Hyrax research project (CMUP-ERI/FIA/0048/2013) and the NOVA LINCS research center (UID/CEC/04516/2013).

MobiQuitous 2017, Melbourne, VIC, Australia

DOI: 10.1145/3144457.3144481

protocol that also enables devices to form communicating groups, and presents energy demands similar to those of Bluetooth [15].

WFD provides node discovery and group formation, authentication and messaging services. In a WFD group, one device takes the role of Group Owner (GO) and acts as an access point providing connectivity and routing to group members, called Clients (CLs).

Nevertheless, WFD has some limitations for medium or large scale scenarios, namely: (a) each GO usually supports no more than 8 clients, and all of them must be in the GO's range; (b) the specification does not tackle inter-group communication; and (c) GOs frequently receive the same addresses (192.168.49.1), hindering the communication between GOs and, consequently, isolating WFD groups from each other. Still, once built, a WFD multi-group communication network will enable long-range and fast data transfer in scenarios without a communication infrastructure.

There are currently several proposals [4, 5, 7, 18] for WFD inter-group communication. However, all of them have limitations because they: (a) use intermittent connections (Delay Tolerant Networks - DTN), where nodes only connect to one group at a time; (b) use slow communication protocols, like *broadcasts* and/or *multicasts*; and/or (c) require two additional relay nodes to bridge two WFD groups.

In this paper, we propose two novel topologies, called GOCRGO and GOGO, for WFD inter-group communication to overcome the above limitations and improve the state-of-the-art. Both topologies use permanent connections, avoiding the time lapse incurred by disconnecting and re-establishing connections of DTN; use IP unicasts, i.e., UDP or TCP communications, enabling higher communication rates than those approaches using broadcasts or multicasts; and require the minimum number of nodes to interconnect WFD groups. GOCRGO and GOGO outperform each other in distinct scenarios, and hence may be used in alternative or cooperatively to provide a better overall network.

The contributions of this paper are twofold: i) the proposal of two novel and complementary WFD-based topologies (GOCRGO and GOGO) for large scale autonomous communications with off-the-self Android devices; and ii) a comparative analysis of the proposed topologies against current state-of-the-art.

In the remainder of this paper we discuss background information and related work in §2; describe the proposed GOCRGO and GOGO topologies in §3; make a comparative analysis of the proposed topologies against the current state-of-the-art in §4; present an experimental evaluation that assesses the communication speed and energy consumption of the proposed and existing topologies in §5; and present our final remarks in §6.

2 BACKGROUND AND RELATED WORK

Wi-Fi Direct (WFD) enables intra-group communication, with one node assuming the role of Group Owner (GO) and acting as an Access Point (AP), providing DHCP service and message forwarding (at MAC level) to the remaining group members. A GO can also accept WiFi clients, called Legacy Clients, if they know the SSID and PSK (pre-shared key) of the announced network. A detailed description of WFD can be found in [3].

As WFD does not tackle inter-group communication, much of previous work focused on a single group. Wong et al [20] shared GO PSKs among clients, using the WFD Bonjour service to allow connection as *legacy clients* and avoid user interaction. Menegato et al [13] proposed a dynamic leader election algorithm for a WFD group, favoring stationary nodes. Asadi et al [2] connected WFD groups but with mobile LTE. Pozza et al [15] compared energy consumption of WFD and Bluetooth when routing traffic in a single group. Rodrigues et al [16] analyzed content dissemination with TDLS inside one WFD group. Marinho et al [12] implemented a routing messaging scheme over the WFD Bonjour message broadcast service, which became saturated with only 6 devices. Jung et al [9] proposed a self-organizing multi-group WFD network based on data locality, working in simulation and not approaching the limitations of WFD multi-group communication. Delay Tolerant Networks (DTNs), where one device inter-connects two WFD groups by connecting to one of them at a time, were initially proposed by Trifunovic et al [19], and later adapted to WFD [6–8, 14]. Funai et al [7] extended the model to allow simultaneous control communication using *multicasts*. These DTN enable inter-group communication, but require buffering and impose delays in data delivery. To the best of our knowledge, Duan et al [5], Casetti et al [4], and Teófilo et al [18], are the only solutions that offer persistent connections to support inter-group WFD communication. For that purpose they exploit the simultaneous use of the devices' WiFi and WFD interfaces.

WFD inter-group connection is not trivial, as all GOs usually have the same IP address (192.168.49.1/24) and devices have one interface (either WiFi or WFD) as their priority interface (*priInt*). Consequently, when both interfaces are connected, all traffic to network 192.168.49 normally goes out by the *priInt*. To complicate things further, the *priInt* behavior is not uniform: some devices only support one active interface at a time (e.g., Motorola G2 2nd generation); some have WiFi as *priInt* (e.g., Nexus 7); while others have WFD (e.g., Nexus 5X, 6, 6P and 9). Also, even if Duan [5] and Casetti [4] report that *broadcasts* can be sent by WFD when WiFi is *priInt*, the opposite is not true; when WFD is *priInt*, *broadcasts* do not go out via the WiFi interface (e.g., Nexus 6, Nexus 9 and OnePlus One). In turn, *multicasts* may be sent via any interface.

We now proceed to present the two most relevant existing WFD inter-group topologies (Casetti et al [4] and Teófilo et al [18]). Throughout the paper we will use a scenario with three WFD groups to examine the characteristics of every topology, focusing on: (i) number of nodes needed per group; (ii) radio (WFD or WiFi) connections; (iii) and type of communications in use. In the figures, elements of each group (GO and connected interfaces) are filled with the same color; radio connections are shown in black dashed lines; and communications in use are depicted in colored lines (depending on their nature). Simple clients and relay clients are denoted by Client (CL) and Client Relay (CR), respectively. Lastly, for the sake of conciseness, IP addresses will be shortened to the last two numbers, e.g., the GOs' addresses are shortened to 49.1.

Group-Owner Client-Relay (GOCR) Topology

To circumvent the facts that GOs normally cannot communicate directly due to conflicting addresses and that devices cannot send UDP unicasts through their *non-priInt*, Duan [5] and Casetti [4] proposed a topology, which we refer to as GOCR, where each GO

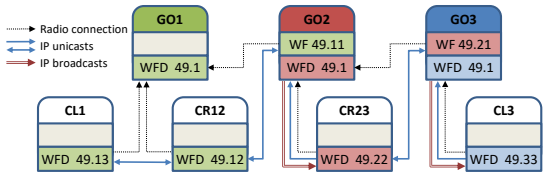


Figure 1: A 3 group scenario in GOCR topology

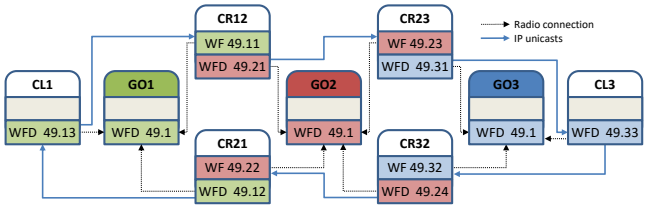


Figure 2: A 3 group scenario in GO2CR topology

has an auxiliary *CR* and uses broadcasts. These works assume WiFi as *priInt*. Fig. 1 depicts this topology, showing the radio connections and the communications links.

In this topology, the GO's clients include the CR and possible other GOs, connected as legacy clients. Accordingly, GO to GO communication, e.g., GO2 to GO3 (from Fig. 1), requires the source GO (GO2) to broadcast a message to its CR (CR23), which relays the datagram to the destination GO (GO3). In the opposite direction, the GO (GO3) has to send a datagram to the CR (CR23) of the destination GO, which resends it to its associated GO (GO2). IP messages between CRs and next group GOs must be forwarded at MAC level by the GO's CR, resulting in 2 MAC messages over the GO's channel.

So, GOCR requires: 2 nodes (1 GO and 1 CR) per WFD group; the use of broadcasts; and 3 routing operations (2 by GOs and 1 by CRs) in each direction.

Group-Owner 2 Client-Relay (GO2CR) Topology

In a previous work [18], we proposed a topology called GO2CR that uses two CRs to interconnect every pair of GOs. For that purpose CRs must connect their WiFi and WFD interfaces, to the GOs, in a symmetrical way, and each CR will only forward data from its *non-priInt* to its *priInt*. This approach does not require CRs to send any data through their *non-priInt*, and so it can depend only on unicast communication. Fig. 2 depicts the radio links (WiFi and WFD) and communication paths, in a 3 WFD group scenario, using this topology. As in [18], due to the characteristics of the devices used, we consider WFD as *priInt* in Fig. 2 and in the remainder of this paper.

Communication-wise, the GO2CR topology is characterized by the fact that a CR can use its *priInt* (WFD interface) to send unicasts directly, at IP level, to the next CR, being relayed at MAC level by the intermediary GO. So, 2 CRs in between 2 GOs, connected in a symmetrical way, will enable communication in both directions. The requirement of 2 CRs between GOs is the main limitation of this topology.

Table 1: Communication assessment

	GO1 – $_{\text{WFD}}$CR$_{\text{WiFi}}$ – GO2				GO4$_{\text{WiFi}.41}$ – GO5$_{\text{WiFi}.51}$ – $_{.61\text{WiFi}}$GO6					
	GO1 ↔ CR		CR ↔ GO2		GO4 ↔ GO5		GO5 ↔ GO6		GO4 ↔ GO6	
	→	←	→	←	→	←	→	←	→	←
TCP	✓$_a$	✓$_a$	✗$_j$	✓$_c$	✗	✗	✗	✗	✗	✗
UDP	✓$_a$	✓$_a$	✗$_j$	✓$_d$	✗	.41$_f$	.61$_g$	.51$_g$	✗	✗
T/U-WF	–	✗	✓$_b$	–	.51$_e$	✗	.61$_h$	.51$_h$	.61$_i$	.41$_i$

T/U-WF: TCP and UDP sockets bound to WiFi interface

3 TWO NOVEL WIFI-DIRECT INTER-GROUP COMMUNICATION TOPOLOGIES

In this section we propose two new inter-group communication topologies that advance the state-of-the-art by requiring the minimum number nodes to interconnect WFD groups. We, first, assess the communication possibilities over the WFD and WiFi interfaces, then present the proposed topologies.

3.1 Communication Assessment

Android 5.0 (API 21) introduced the capability to bind TCP sockets to a specific interface, being the feature extended to datagram sockets in Android 5.1 (API 22). So, with Android 5 Compliant (A5C) devices it is possible to create sockets that force traffic out through a specific interface, circumventing the *priInt*. Given this context, we will assess which communication possibilities are available to A5C and *non*-A5C devices using their WFD and WiFi interfaces. Table 1 lists the communication behavior in two cases: 2 GOs interconnected by a relay node; and 3 GOs. Both cases were tested with Nexus 6, Nexus 9 and OnePlus One devices, all with WFD as *priInt*. Note that table addresses just mention the last octet and are only mentioned when they are not obvious. As UDP broadcasts and multicasts have slow communication speed, they are not used in the proposed topologies and are omitted from the table.

We begin by analyzing the case of a CR in between GOs: case GO1–CR–GO2 in the table. On the CR *priInt* side, both the CR and the GO1 can create UDP or TCP sockets to the other (✓$_a$). On its *non-priInt* side, the CR can communicate with the GO2 by creating UDP or TCP sockets bound to the WiFi interface (✓$_b$). However, binding sockets to the WiFi interface is only available in A5C devices. If CR is a *non*-A5C one, it has to wait for a TCP connection coming from GO2 (✓$_c$) and leverage on the bi-directional nature of that connection. Although the GO2 could send datagrams to the CR (✓$_d$), it would be useless because the CR could not answer in the same protocol.

We move now to the analysis of GO to GO communication: case GO4–GO5–GO6 in Tab. 1. We start by assessing the situation of GO4–GO5, where the GO4 is a client of the GO5. As these GOs have the same IP address (192.168.49.1), they cannot communicate using that address. Ergo, to communicate, these devices must send the data to their peer's WiFi IP address: cases (.41$_f$) and (.51$_e$). The former (.41$_f$) is only possible because, unlike TCP, the UDP stack at GO4 does not eliminate packets coming from an IP address considered local (.1). The latter requires GO4 to create a unicast socket bound to its WiFi interface and direct all communication (TCP or UDP) to the address of GO5's WiFi interface (.51$_e$). This

enables: a) sockets to link the correct interfaces, and to have a source address other than .1 (.41, in this case) needed for TCP; and b) to have a destination address also other than .1 (.51, in this case). The downside is that, to enable WiFi bounded sockets, GO4 must be an A5C device. In summary, data exchange between a GO_{CL} (GO4) that is a client of another GO_{GO} (GO5) is only possible if the GO_{CL} is an A5C device and the GO_{GO} has its WiFi interface connected. In that case, GO_{CL} can create a TCP connection to GO_{GO}, or they can use UDP datagrams. Now we assess the GO5–GO6 connection. In this case GO5 and GO6 are both interconnected by their WiFi interfaces, and can communicate directly with UDP ($.61_g$, $.51_g$) or using UDP or TCP with sockets bound to WiFi ($.61_h$, $.51_h$), if they are A5C. All those sockets or datagrams must be directed to the WiFi interface IP address of the destination. Lastly, GOs that are clients of one common GO, as GO4 and GO6, can communicate using UDP or TCP sockets bound to the WiFi interface and linked to the address on the other GO WiFi interface ($.61_i$, $.41_i$).

In conclusion, it is possible to communicate, using UDP or TCP unicasts: (i) between GOs connected by a CR, if the CR is A5C or if it receives a TCP connection from its non-*priInt* side; and (ii) between GOs, if both are in the same network, if they have the WiFi interfaces connected, and, also, if the client one is A5C.

3.2 Proposed Topologies

So, we propose two new topologies, called Group-Owner Client-Relay Group-Owner (GOCRGO) and Group-Owner Group-Owner (GOGO), that resort only to unicasts and leverage on the ability to send data: (i) bidirectionally in TCP connections; (ii) through a socket bound to the *non-priInt*; and (iii) to the WFD interface of a target device, but directed to its WiFi interface IP address.

Group-Owner Client-Relay Group-Owner (GOCRGO) Topology

The GOCRGO topology inter-connects every pair of GOs using one CR and can be used by any Android device. In this topology, each CR will connect to one of the GOs using its WiFi interface and to the other GO using its WFD interface. From the *priInt* side, the CR has no problems creating TCP sockets or sending UDP datagrams (Tab. 1, $\checkmark_a$). From the *non-priInt* side, the CR cannot create any kind of socket ($\times_j$) without resorting to the features of Android 5 (as in $\checkmark_b$). However, levering on the bidirectional nature of a TCP connection established from the GO on that side (or any CR connected to it) ($\checkmark_c$), the CR can send data to that side. Fig. 3 showcases the use of this topology, resorting only to the TCP protocol. It uses TCP connections from CL1 to $CR12_{.11}$, from CR12 to $CR23_{.23}$, and from CR23 to $CL3_{.33}$ or from CL3 to $CR23_{.31}$, which will enable communication between all nodes. Furthermore, if the CRs are A5C devices, it is also possible to communicate using only UDP datagrams.

This topology can be used directly by devices with WiFi as *priInt*. Hence, if a CR is a non-A5C device, it should receive a TCP connection from its non-*priInt* side. In Fig. 3, considering WiFi as *priInt*, we can create TCP connections from CL3 to $CR23_{.31}$, from CR23 to $CR12_{.21}$ and from CR12 to $CL1_{.13}$ (or from CL1 to $CR12_{.11}$).

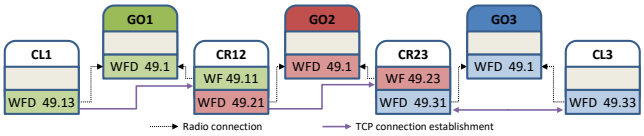


Figure 3: A 3 group scenario in GOCRGO topology

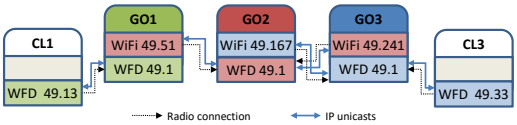


Figure 4: A 3 group scenario in GOGO topology

Group-Owner Group-Owner (GOGO) Topology

The GOGO topology enables a direct GO–GO connection, but all GOs must be A5C devices and have their WiFi interfaces connected.

Fig. 4 depicts the use of this topology for our 3-group scenario. We use the notation "$D_i \rightsquigarrow E_a$" to denote a data transmission from device D, bound to interface i, to address a of device E. If *priInt* is used, i is omitted. In that scenario, UDP communication from CL1 to CL3 follows path $CL1 \rightsquigarrow GO1_{.1}$, $GO1_{WiFi} \rightsquigarrow GO2_{.167}$, $GO2 \rightsquigarrow GO3_{.241}$, and finally $GO3 \rightsquigarrow CL3_{.33}$[2]. In turn, communication from CL3 to CL1 follows the path $CL3 \rightsquigarrow GO3_{.1}$, $GO3 \rightsquigarrow GO2_{.167}$, $GO2 \rightsquigarrow GO1_{.51}$, and finally $GO1 \rightsquigarrow CL1_{.13}$[3].

The use of TCP requires the establishment of the following (bidirectional) connections: $CL1 \rightsquigarrow GO1_{.1}$, $GO1_{WiFi} \rightsquigarrow GO2_{.167}$, $GO2_{WiFi} \rightsquigarrow GO3_{.241}$ and finally $GO3 \rightsquigarrow CL1_{.33}$[2].

If the GOs have WiFi as *priInt*, and even if the devices are *non*-A5C, each GO can create a TCP socket from its WiFi interface to the WiFi address of the GO connected on that interface, allowing bidirectional communication between them. As an example, in Fig. 4, GO1 can create TCP (or UDP) sockets to $GO2_{.167}$ [4]. Furthermore, GO2 can create UDP sockets to $GO1_{.51}$, but only if bound to the WFD interface. So this topology, with WiFi as *priInt*, needs A5C devices to allow UDP communication.

Brief Topology Discussion

GOCRGO requires a single relay node between each pair of GOs, which is an improvement over GO2CR that needs two relay nodes. GOCRGO is also better than GOCR because it eliminates the use of broadcasts and enables the relay node to increase the distance between GOs. GOCRGO can also be used by any Android device, while GOGO is restricted to A5C devices. In turn, GOGO is the only topology to offer a direct GO to GO connection, with no need for CR nodes.

When considering the number of nodes required to interconnect WFD groups (excluding GOs), these topologies need the minimum number possible: GOGO does not require extra nodes, while GOCRGO only needs one more node (with the beneficial side-effect of extending the radio coverage). Thus, by requiring fewer nodes

[2]Path can be shortened in 1 hop by either: $GO1_{WiFi} \rightsquigarrow GO3_{.241}$; or $GO2_{WiFi} \rightsquigarrow CL3_{.33}$.
[3]Path can be shortened in 1 hop by either: $CL3 \rightsquigarrow GO2_{.167}$; or $GO3_{WiFi} \rightsquigarrow GO1_{.51}$.
[4]It can also be $GO2 \rightsquigarrow GO3_{.241}$ and $GO3 \rightsquigarrow GO2_{.167}$ (note: WiFi is *priInt*).

to interconnect WFD groups, GOCRGO and GOGO broaden the scope of possible application scenarios for WFD-based networks.

4 TOPOLOGY ANALYSIS

In this section we perform a comparative analysis of the novel topologies proposed in this paper (GOCRGO and GOGO) with the current state-of-the-art topologies (GOCR and GO2CR) concerning their spatial node requirements, communication speed, routing demands, frequency usage, network redundancy and connection flexibility, and behavior in extreme situations.

Spatial node requirements

We begin by analyzing the node density per WiFi range (*#node*/WFR) imposed by each topology. For that purpose, we consider the conceptual measure WFR, that denotes the maximum coverage distance of a WiFi (and WFD) radio in a mobile phone.

GOCR requires 1 GO and 1 CR per WFR, *#node*/WFR = 2; GO2CR requires 1 GO and 2 CRs per 2 WFRs, *#node*/WFR = 1.5; GOCRGO requires 1 GO and 1 CR per 2 WFRs, *#node*/WFR = 1; and, lastly, GOGO requires 1 GO per WFR, *#node*/WFR = 1. This information is summarized in the table below.

Topology	#GO	#CR	Range (in WFRs)	#node/WFR
GOCR	1	1	1	2
GO2CR	1	2	2	1.5
GOCRGO	1	1	2	1
GOGO	1	0	1	1

So, the proposed topologies, GOGO and GOCRGO, present the best average values for #node/WFR. Among them, GOGO, provides the best radio coverage because it builds only on GOs.

Communication Speed

We now want to assess the maximum communication speed (*MaxComSpeed*) in one direction, while in the presence of bi-directional simultaneous communications. To that end, we consider the conceptual measures:

- **WFS (WiFi Speed)** — the *MaxComSpeed* of unicast communication in WiFi (and WFD);
- **SMAX** — the *MaxComSpeed* of communication in one direction, when the topology is used in a single direction;
- **SBDMAX** — the *MaxComSpeed* of communication in one direction, when the topology is simultaneously used in both directions.

For example, between two directly connected devices, UDP unicast communication has SMAX = WFS and SBDMAX = WFS/2, since, in the latter, the channel has to be shared between two unicast communications.

For our analysis we make three assumptions: i) devices can communicate simultaneously on both interfaces, i.e., they have 2 radios; ii) GOs operate on independent WiFi channels without any interference; and iii) for simplicity's sake, we make the rough approximation of considering the same speed for TCP and UDP communications. To deal with broadcasts, we define **BCS** as the *Broadcast Speed* and **BCF** as *Broadcast Factor*, with BCF = WFS/BCS. To give a sense of proportion, we consider the concrete values of WFS = 54 *Mbps* and BCS = 6 *Mbps*, leading to BCF = 9, i.e., one broadcast will be equivalent to 9 unicast messages in the channel.

Every result computed from these assumptions is marked with *. For short, we will use term "IP message" to denote either UDP datagrams or data over TCP channels. Additionally, both IP and MAC messages will be, by default, unicast messages. The results of our analysis are summarized in the following table and explained below.

Topology	SMAX (*Mbps**)	SBDMAX (*Mbps**)
GOCR	WFS/(2 + BCF) = 4.9, WFS/3 = 18.0	WFS/(5 + BCF) = 3.86
GO2CR	WFS/2 = 27.0	WFS/4 = 13.5
GOCRGO	WFS/2 = 27.0	WFS/4 = 13.5
GOGO	WFS/1 = 54.0	WFS/2 = 27.0

In GOCR, and considering Fig. 1, from right to left, a GO has to handle 1 IP message from the GO at its right to its CR (2 MAC messages), and 1 IP message from the CR to the GO itself. That sums up to 3 MAC messages in the GO's channel, and so SMAX = WFS/3 = 18*Mbps**. From left to right, a GO has to send a broadcast to its CR, which must forward the message as an IP message to the GO at its right, taking 2 MAC messages in the GO's channel. This sums up to 2 + BCF MAC messages, thus SMAX = WFS/(2 + BCF) = 4.9 *Mbps**. Consequently, each GO channel has to support 3 + 2 + BCF messages, resulting in SBDMAX = WFS/(5 + BCF) = 3.86 *Mbps**.

In GO2CR, communication is symmetric, and in each direction requires 1 IP message between CRs, resulting in 2 MAC messages over the channels, hence SMAX = WFS/2 = 27 *Mbps**, leading to SBDMAX = WFS/4 = 13.5 *Mbps**.

In GOCRGO the behavior is similar to GO2CR, 1 IP message between CRs, so the results are the same.

Finally, in GOGO every GO channel carries a single IP message in either direction. Ergo, SMAX = WFS = 54 *Mbps** and SBDMAX = WFS/2 = 27 *Mbps**.

Routing

This section discusses the routing demands for inter-group communication, for each of the four topologies under analysis. For readability's sake, we will use IPRO and MACRO to designate a routing operation at IP or MAC level, respectively.

GOCR is a symmetric topology with respect to routing. Considering Fig. 1 once again, starting at GO2, this device performs an IPRO and sends a broadcast to CR23, which, in turn, performs another IPRO and sends a message to GO3, that requires a MACRO by GO2. In the opposite direction, the routing behavior is symmetrical. Hence, in each direction, communication requires 3 routing operations (2 IPROs and 1 MACRO) to cross one group.

The GO2CR topology is symmetric wrt to routing, but each CR routes traffic in a single direction. This topology requires 1 GO MACRO and 1 CR IPRO, in a total of 2 routing operations, in each direction.

GOCRGO behaves similarly to GO2CR, but CRs route all the channel's traffic, and GOGO requires a single GO IPRO to cross groups.

Finally, given that the topologies present distinct spatial behaviors, to ensure fairness, our analysis must also take into consideration the number of routing operations per WFR (RO/W). In both GOCR and GOGO, 1 WFD group can extend the network by 1 WFR, hence $GOCR_{RO/W} = 3/1 = 3$ and $GOGO_{RO/W} = 1/1 = 1$. In the case of both GO2CR and GOCRGO, 1 WFD group can extend the

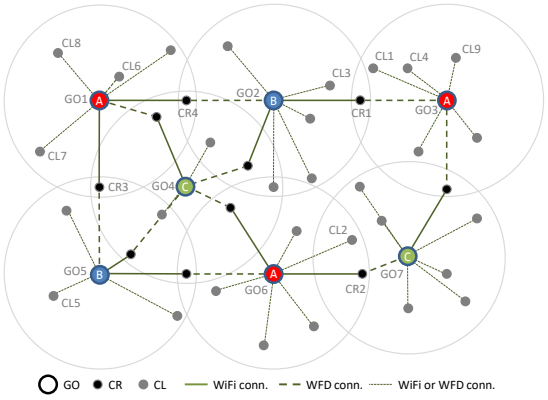


Figure 5: GOCRGO topology applied to a 2D scenario.

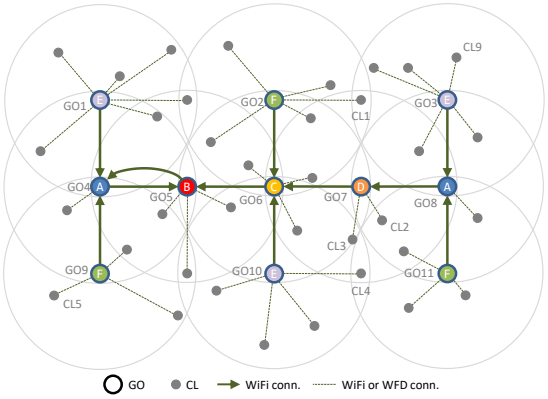


Figure 6: GOGO topology applied to a 2D scenario.

network by 2 WFRs, so $\text{GO2CR}_{\text{RO/W}} = \text{GOCRGO}_{\text{RO/W}} = 2/2 = 1$. This information is summarized in the table below.

Topology	# IPRO	# MACRO	# Total	# RO/WFR
GOCR	1 GO + 1 CR	1 GO	3	3/1 = 3
GO2CR	1 CR	1 GO	2	1/1 = 1
GOCRGO	1 CR	1 GO	2	2/2 = 1
GOGO	1 GO	—	1	2/2 = 1

Network frequency usage

In this section we want to know the number of non-overlapping frequencies needed, by the topologies, to offer interference-free communication.

Concerning frequency, or radio channel, analysis, GO2CR and GOCRGO are identical, as they need intermediary nodes (1 or 2) to connect GOs. Likewise, GOCR and GOGO topologies are also identical, as both require direct GO-GO connection. Thus, for simplicity's sake, we confine our analysis to GOCRGO and GOGO, here and in the remaining analyses. Therefore, to support our comparison and analysis, Figs. 5 and 6 illustrate the application of both topologies to the same bi-dimensional[5] (2D) scenario. The figures depict the radio connections (WiFi or WFD) between CLs/CRs and GOs, always directed from the former to the latter, and display the channels in use with colors and letters. Figure 6 also shows the direction of the GO–GO connections. Note that GO4 and GO5 are double connected, because, in GOGO, all GOs must have their WiFi interface connected.

In **GOCRGO**, conflict-free communication, in straight line (1D), between GOs needs only 2 non-overlapping frequencies. But, it requires that consecutive GOs with the same frequency to be more than $3 \times \text{WFR}$ apart, and consecutive CRs to be more than WFR apart. This can be observed in the GO1–GO3 row of nodes in Fig. 5. Regarding bi-dimensional scenarios, we claim, without proving, that GOCRGO requires only 3 non-overlapping frequencies to avoid interference.

In **GOGO**, straight line conflict-free communication between GOs that are more than WFR/2 apart requires 4 non-overlapping frequencies, that must be assigned circularly, e.g., A, B, C, D, A, B, ..., as can be observed in the GO4–GO8 line of nodes in Fig. 6. To expose why 4 frequencies are needed, lets consider GO6 in the same figure: GO5 and GO7 should have their own non-overlapping frequencies because GO6 needs to use GO5's frequency, and GO8 should not use GO6's frequency in order not to cause interference in GO7. So, a frequency can only be used once every 4 in line GOs. Regarding bi-dimensional communication, we claim (once again without proving) that GOGO needs at least 6 frequencies to avoid interference. For example, Fig. 6 needs 6 frequencies.

To conclude, GOCRGO and GO2CR cover more area per WiFi channel and need fewer non-overlapping frequency channels to communicate without collisions than GOGO and GOCR.

Network path redundancy

In the section we analyse the network path redundancy potential of the GOGO and GOCRGO topologies. We start by characterizing the topologies' geographic structure and, subsequently, analyse their path redundancy potential.

Geographic structure. The GOCRGO topology can be used to connect GOs and CRs according to a mesh structure, like in Fig. 5. In turn, GOGO forcibly connects GOs in a tree structure, with a single loop, arising from the need to connect the WiFi interface of every GO. Although this loop is confined to nodes GO4 and GO5 in Fig. 6, it can be used to create a more extended loop. But such extended loops will be difficult to manage as they will require non-local information.

Path redundancy. The mesh structure of GOCRGO may originate *multiple paths* between distant nodes (e.g., paths GO1–GO2–GO3–GO7 and GO1–GO5–GO6–GO7, in Fig. 5), as well as between adjacent GOs (GO2–GO4, also in Fig. 5). Conversely, the tree-based structure of **GOGO** cannot build redundant paths, other than the ones that use the loop that connects the root of the tree to another GO (GO4–GO5, in Fig. 6). Thus, in GOCRGO, we may leverage on the multiple alternative paths to perform network *traffic splitting*. For instance, given that in GOCRGO the maximum communication speed/bandwidth per link (*SMax*) is WFS/2, n alternative paths between two areas may increase such speed/bandwidth to $n \times \text{WFS}/2$. Conversely, in GOGO, the root tree GO will have to support all the traffic between the GOs connected to it and their trees of connected nodes, which will be a major bottleneck. Alternative paths may

[5]Relative to node disposition.

also be used to improve the network's *energy efficiency* by 1) enabling the choice of *shorter paths*, and 2) providing a framework for incorporating link load, node battery level and path distance in routing decisions, enabling the choice of paths comprising nodes with higher energy levels, and with that prevent node energy exhaustion and defer network reconfigurations as much as possible.

Path redundancy in GOGO is confined to the previously mentioned top loop, but that alone should not be enough to avoid long and congested paths. In Fig. 6 we can observe that traffic between clients of GO2 and GO3 must follow a long path (GO2, GO6, GO7, GO8 and GO3), albeit the proximity of these nodes, and, as already mentioned, traffic between nodes at opposite edges of the network must always travel across GO4, GO5, GO6, GO7 and GO8, contributing to a communication speed bottleneck.

In summary, one can leverage on the path redundancy provided by GOCRGO to implement a smart routing strategy that makes use of traffic splitting to improve communication speed and reduce energy consumption.

Network flexibility

We now assess how topologies react to node churn, that is the egress and ingress of nodes.

In GOCRGO, path redundancy provides general resiliency to node churn and in particular to the egress of CRs. For instance, in Fig. 5, if we remove a single CR, any other node will remain connected. Even, if both CR1 and CR2 leave the network, connectivity may be re-established by replacing CR2 by CL2, or by promoting CL1 to the role of Group Owner to interconnect GO2 and GO3, using CL3 and CL4 as CRs. The egress of a GO will require some reconfiguration to reconnect its clients, but the operation should be mostly local if there are nodes available nearby. Broader reconfigurations may occur when node distribution is very sparse. Usually, the exit of a GO can be solved by electing a nearby CL to become a GO.[6] However, if there are still unconnected nodes, another GO, or GOs, must be elected to connect them. For example, in Fig. 5, if GO1 runs out of battery, CL6 can become a new GO (GOx6). However, as GOx6 cannot reach either CL7 or CR3, CL7 may also become a GO (GOx7) and connect to CR3. In such reconfiguration, with the exception of CR3 and CL7, all the former clients of GO1 should connect to GOx6.

In GOGO, the egress of GOs may require significant and non-local changes because the tree-like structure must be preserved, with the GOs WiFi connections directed to the root of the tree. Looking once again at Fig. 6, if GO7 fails, we can promote CL2 and CL3 to GOs to create a path between GO6 and GO8. However, if for some reason those nodes are not available, we will have to resort to CL1 or CL4. Promoting CL1 to a GO (GOx1) that interconnects GO2 and GO3 forces GOx1 to connect its WiFi interface to GO2, requiring GO3 to disconnect its WiFi interface and connect it to GOx1, and GO8 to disconnect and connect to GO3. A similar situation arises if we promote CL4 to a GO, to interconnect GO11 and GO10. Admittedly, the extent of connection reversal depends on the place in the tree where the reconfiguration must be performed. So, unlike GOCRGO, node egress in GOGO may require non-local reconfigurations even when node distribution is dense.

[6]The electing criterion is out of the scope of this paper.

Regarding *node ingress*, there are many scenarios to consider. Nonetheless, in all the scenarios we analyzed, GOGO always required more actions than GOCRGO. We showcase how the topologies can react to a new node with two paradigmatic scenarios.

Scenario 1: a new node, CLa, comes in reach of (and only of) CL5, at the bottom left in both Fig. 5 and 6. In GOCRGO: 1) CLa switches into a GO (GOa); and 2) CL5 connects its available interface to GOa. In GOGO: 1) CL5 disconnects from GO9; 2) CL5 switches into a GO (GOb); 3) GOb connects via WiFi to GO9; and 4) CLa connects to GOb.

Scenario 2: two non-connected networks, Na and Nb (replicas of Fig. 5 or of Fig. 6, respectively), get in range of each other, with Na located at bottom left and Nb at the top right, and only nodes CL9 from Na ($CL9_{Na}$) and CL5 from Nb ($CL5_{Nb}$) are in range of each other. In GOCRGO (both Na and Nb are replicas of Fig. 5): 1) $CL9_{Na}$ disconnects from $GO3_{Na}$; 2) $CL9_{Na}$ becomes a new GO (GOx); 3) $CL4_{Na}$ connects with its free interface to GOx; and 4) $CL5_{Nb}$ connects to GOx . In GOGO (both Na and Nb are replicas of Fig. 6), both $GO3_{Na}$ and $GO9_{Nb}$ have their WiFi interfaces connected to other nodes, thus the networks cannot be interconnected without reconfiguring the structure of at least one of them. Hence, a runtime analysis is necessary to choose which network will require fewer structural changes to be connected to the tree of the other network, forming a single global tree. If we choose to connect Nb to Na, the following actions are necessary: 1) $CL5_{Nb}$ disconnects from $GO9_{Nb}$; 2) $CL5_{Nb}$ turns into a GO ($GOx5_{Nb}$); 3) $GO4_{Nb}$ disconnects from $GO5_{Nb}$; 4) $GO4_{Nb}$ connects to $GO9_{Nb}$; 5) $GO9_{Nb}$ disconnects from $GO4_{Nb}$; 6) $GO9_{Nb}$ connects to $GOx5_{Nb}$; 7) $CL9_{Na}$ disconnects from $GO3_{Na}$; 8) $CL9_{Na}$ turns into a GO ($GOx9_{Na}$); 9) $GOx9_{Na}$ connects to $GO3_{Na}$; and lastly, 10) $GOx5_{Nb}$ connects to $GOx9_{Na}$. Alternatively, connecting Na to Nb requires changing the connections between nodes GO5, GO6, GO7, GO8 and GO3 of Na.

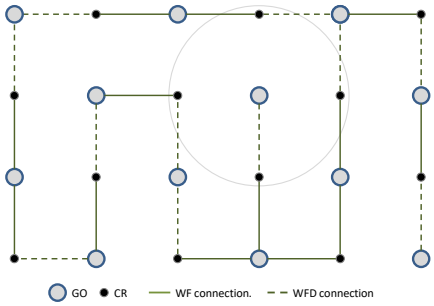
In summary, the GOCRGO mesh structure, with redundant paths, can tolerate some structural network changes without requiring compensation actions, and when some adjustment actions are required they should be mostly local. In GOGO, structural network changes will often require reconfigurations that reach far beyond the place where the change occurred. In the light of these findings, we conclude that GOCRGO is more flexible than GOGO.

Extreme situations: sparse and crowded networks

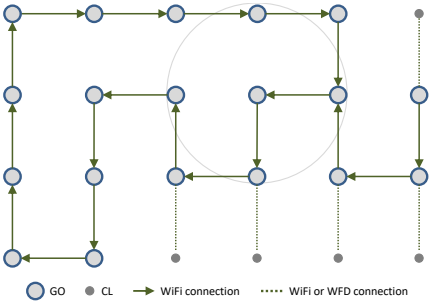
In this section we discuss the topologies' behavior in the extreme cases of sparse and crowded situations.

In *sparse networks*, nodes are at the limit of the GOs' coverage range. Figs. 7a and 7b illustrate the application of the GOCRGO and GOGO topologies, respectively, in the same sparse scenario. Both figures show the radio range of one node, and the nodes are disposed in a grid of equally sized square cells. In this scenario, nodes are so sparse that it is very difficult to create redundant paths with GOCRGO. Thus, we have that GOCRGO's $S_{MAX} = wfs / 2$ and GOGO's $S_{MAX} = wfs$. Also, as every node in GOGO can be a GO, GOGO provides better radio coverage than GOCRGO.

The fact that the nodes are at the coverage boundaries precludes the use of the GO2CR topology, because there are not enough nodes to interconnect two adjacent GOs, over all the mesh. However,

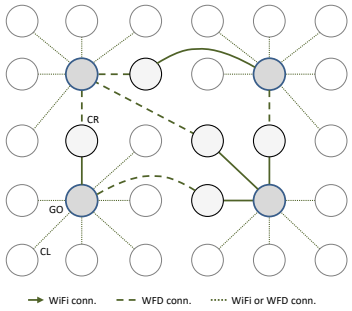


(a) GOCRGO

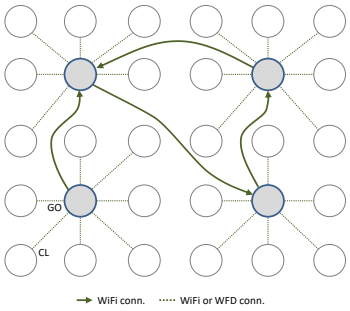


(b) GOGO

Figure 7: Application of GOCRGO and GOGO to a very sparse scenario.



(a) GOCRGO



(b) GOGO

Figure 8: Application of GOCRGO and GOGO to a crowded scenario.

GOCR can still be used because this particular scenario follows a 6×4 node distribution, which falls into one of the two types of scenarios where the topology may be applied: rectangles of $2 \times n$ and $4 \times n$.

In *crowded scenarios*, like in a stadium, nodes are very close to each other. Figs. 8a and 8b depict, respectively, the application of the GOCRGO and GOGO topologies in the same crowded scenario. In this kind of scenario, all the nodes are in radio coverage range of each other, so any node can connect to any other node. With nodes so close, the noise to signal ratio will be very high due to the interference from overlapping GOs. Consequently, in GOCRGO, redundancy with short paths and traffic splitting can't be very helpful, because every communication will probably overlap with many others. Also, when in the presence of high interference, each communication should be as close as possible, otherwise success will be very limited. In turn, GOGO will use less bandwidth than GOCRGO because it only requires 1 message to cross WFD groups, against the 2 messages required by GOCRGO. Furthermore, GOGO also only needs one routing action to cross WFD groups, against the two required by GOCRGO. It is thus expected that GOGO will be more energy efficient than GOCRGO in this kind of scenarios.

From the above we are able to conclude that GOGO can offer better communication speed in sparse and crowded scenarios, and also requires fewer routing actions in the latter.

Table 2: Topologies comparison

Topology	Nodes per WFR	Comm. Speed SBDMAX	Routing per WFR	Freqs per 2 WFRs	Freqs needed 1D/2D	RC	RP TS SP BEM ECS NF	ES S/C	AD
GOCR	2	WFS/14*	3	2	4 / 6†	+	✗	✗/–	ANY
GOGO	1	WFS/2	1	2	4 / 6†	+	✗	+/+	A5C
GO2CR	1.5	WFS/4	1	1	2 / 3†	–	✓	✗/–	ANY
GOCRGO	1	WFS/4	1	1	2 / 3†	–	✓	–/–	ANY

* SBDMAX = WFS/(5 + BCF), and using BCF = 9 (WFS = 54, BCS = 6)
† Needs demonstration.
Acronyms: RC = Radio coverage; RP = Redundant Paths; TS = Traffic Splitting; SP = Short Paths; BEM = Better Energy Management and Efficiency; ECS = Extended Communication Speed; NF = Network Flexibility; ES = Extreme Situations; S/C = Sparse / Crowded scenarios; AD = Android Device; A5C = Android 5 Compliant device.

Final analysis

Table 2 summarizes our findings. Given that GOGO is similar to GOCR, as they both use direct GO to GO connection, and GOGO is always better or equal than GOCR, we choose GOGO as the best option from both. The same happens with GOCRGO and GO2CR, as both need relay nodes in-between GO and the former is always better or equal than the latter, GOCRGO is the best option.

Comparing the two proposed topologies, GOGO and GOCRGO, *GOGO's strong points* are communication speed, radio coverage, and behaviour in extreme cases, while *GOCRGO's strong points* are network path redundancy, traffic splitting, shorter paths, better energy management and efficiency, extended communication speed, better network flexibility, better frequency usage, and the ability to be used by any Android device. Taking into consideration these

Table 3: Experimental setups

Topology	Nodes	TCP connections
GOCRUC	GO2, CR23, GO3	GO2 ⇝ CR23, CR23 ⇝ GO3
GO2CR	GO2, CR23, CR32, GO3	GO2 ⇝ CR23, CR23 ⇝ GO3, GO3 ⇝ CR32, CR32 ⇝ GO2
GOCRGO	GO2, CR23, GO3	GO2 ⇝ CR23, CR23 ⇝ GO3
GOGO	GO2, GO3	GO2 ⇝ GO3

results, our conclusion is to choose GOGO for extreme scenarios and GOCRGO for all the other cases. Furthermore, both topologies can co-exist in networks with a wide diversity of node concentration.

5 EXPERIMENTAL RESULTS

In this section we experimentally evaluate all topologies regarding communication speed and energy consumption: the two current state-of-the-art topologies, GOCR and GO2CR, and the ones proposed in this paper, GOCRGO and GOGO. The experimental testbed is composed of Nexus 6 and Nexus 9 phones, all with WFD as *priInt*. Given that GOCR was proposed for devices with WiFi as *priInt*, we had to develop a variant, which we named Group-Owner Client-Relay UniCasts ($GOCR_{UC}$) to use that topology in our devices.

Group-Owner Client-Relay UniCasts ($GOCR_{UC}$) Variant

$GOCR_{UC}$ is, thus, a variant of GOCR that considers WFD as *priInt*. To respect the nature of GOCR, $GOCR_{UC}$ keeps its radio connections and does not use Android 5 capabilities. In this setting, since GOs cannot send unicast traffic through their WiFi interface, they must accept a TCP connection in that interface and use the connection's bi-directional nature to communicate data out. Hence, each CR of a GO (say GO_i) must establish a TCP socket to every GO that is a client of GO_i. To adapt Fig. 1 to reflect this variant, we can use TCP or UDP communication between pairs CL1 – CR12, GO2 – CR23, and GO3 – CL3, and should use the following TCP connections: CR12 ⇝ GO2 and CR23 ⇝ GO3. Given that communication in $GOCR_{UC}$ is symmetric, for each direction it requires 1 IP unicast from the GO to its CR, and another from the CR to the next GO, resulting in 3 MAC unicasts over the GO channel. Hence SMAX = WFS/3 = 18 *Mbps**, and SBDMAX = WFS/6 = 9.0 *Mbps**. As a side note, if we revert to the case of having WiFi as *priInt*, the $GOCR_{UC}$ topology can still resort exclusively on unicasts, if each CR creates a TCP connection to its GO and this connection is used bi-directionally.

Experimental evaluation

All experiments report on the communication speed computed from the round-trip of transmitting 100 MB between two GOs over TCP channels and using 1 KB data buffers. The exclusive use of TCP communication provides a homogeneous setting for all topologies, as TCP is required in $GOCR_{UC}$. All reported values are the average of 10 experiments conducted with a level of interference smaller than -80 dBm in the GOs' channels.

The test scenarios mimic those of Fig. 1 (adapted to $GOCR_{UC}$), and Figs. 2, 3 and 4 having UDP communication replaced by the TCP connections presented in Tab. 3. Figure 9 depicts the resulting

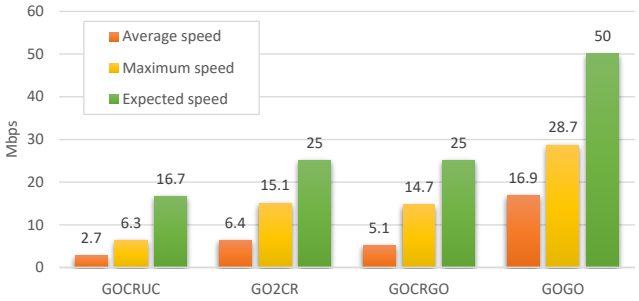


Figure 9: Topologies communication speed

Table 4: Energy consumption per topology

Protocol	Devices	Data Transm.	Energy (Joules/MB)
$GOCR_{UC}$	3	6	4.8
GO2CR	4	4	2.9
GOCRGO	3	4	2.6
GOGO	2	2	1.3

average, maximum, and expected communication speeds, where *maximum* relates to the maximum speed observed in 1 second periods at reception in the final GO, and *average* conveys the average speed from the instant when the GO starts the transfer until it receives the last data byte. For the *expected* value, we considered the maximum speed SMAX = 100 *Mbps*, as the actual measured value was 103.8*Mbps* in a TCP connection from a CL to its GO.

As depicted in Fig. 9, the (*maximum* and *average*) communication speeds are considerably lower than the *expected* values. That is due to the fact that the communication speed slows down when devices have both interfaces connected, making evident the hardware limitations of these commodity smartphones. Nonetheless, the results confirm that GOGO is, as expected, the fastest topology, given that it requires fewer messages per GO channel. GO2CR performed slightly better than GOCRGO, benefiting from less load on the CRs, given that in GOCRGO CRs have to handle all data traffic in both directions, while in GO2CR that load is distributed among CRs. $GOCR_{UC}$ was clearly the worst performing topology of all, but it should be noted that GOCR's performance would be far worse: $GOCR_{UC}$'s expected speed is SBDMAX = 16.7 *Mbps* (WFS / 6), while GOCR's expected speed is SBDMAX = 4.6 *Mbps*, considering WFS = 100 *Mbps*, and BCS = 6 *Mbps*, as the measured broadcast speed (BCS) from a GO was 5.5 *Mbps*.

Regarding energy consumption, Table 4 depicts the (average of the) energy required by all devices in each setup. Values are presented in *Joules/MB*, and were extracted by reading the devices' instantaneous current and voltage every second. Due to the differences between devices, to get more truthful values we subtracted the devices' expected energy consumption at rest. All tests were realized with the screens turned on. As expected, energy consumption depends mainly on the number of data transmissions. Nevertheless, the number of devices is also relevant because more devices will mean more nodes accessing the network, which ultimately causes more conflicts in radio channels.

Comparing GOCR and $GOCR_{UC}$: GOCR is expected to spend more energy than $GOCR_{UC}$, as the use of broadcasts requires more than 10 times the energy of unicasts. We measured the average energy required by a GO (Nexus 6) to send data, and the results are ($joules/MB$): 4.72 for broadcasts, and 0.35 for UDP unicasts.

Another important measure is the *energy consumed per* WFR. Given that the GOCRGO and GO2CR scenarios cover 2 WFRs, we tested GOGO with 3 aligned GOs to cover that same distance. GOGO's average speed was 7.4 $Mbps$ and the energy consumed was 2.9 $joules/MB$. As the speed decreased significantly, the power increased due to the difficulties in the communications, revealing the devices' hardware limits when a GO is using both interfaces simultaneously. We thus conclude that GOGO (with 2.9 per 2 WFRs) is less energy efficient, per WFR, than GOCRGO (with 2.6 per 2 WFRs).

From the conducted experiments we observed that: a) GOGO is better than all other topologies when it comes to communication speed; and b) GOCRGO is faster than GOCR ($GOCR_{UC}$), and more energy efficient than all its competitors (GOCR, GO2CR and GOGO).

6 CONCLUSIONS AND FUTURE WORK

In this paper we proposed two new WiFi-Direct inter-group communication topologies, GOCRGO and GOGO, for off-the-self mobile devices. They advance the state-of-the-art by requiring the minimum number of nodes per radio range (WFR) and offering better communication speed. As a result, they improve the efficiency of communication in networks for off-the-shelf mobile devices, contributing to pave the way for mobile collaborative systems.

The head-to-head comparison of GOCRGO and GOGO shows that none is better than the other in all scenarios. GOGO offers better radio coverage and communication speed in sparse and crowded scenarios. In turn, GOCRGO behaves better in the remainder scenarios, offering shorter and alternative communication paths, traffic splitting, better energy management and efficiency, extended communication speed, more network flexibility and better frequency usage. Additionally, GOCRGO does not make any impositions on the device's Android version, opposed to GOGO that requires GOs to be Android 5 compliant devices.

Future work will focus on the development of algorithms for the automated formation of networks that use the GOCRGO and GOGO topologies, separately or cooperatively, to provide the best communication backbone possible. These algorithms must also be resilient to node churn, agnostic to the devices' priority interface, and take in consideration the devices' Android version.

However, there are still challenges to overcome when using WFD to form multi-group networks of commodity smartphones. The main one is that the topologies build on devices that should efficiently handle simultaneous communications on their WiFi-Direct and WiFi interfaces. Unfortunately, none of devices that we have tested so far fulfills this requirement, as they reveal inconsistent behavior when transferring data on both interfaces simultaneously, with frequent connection breakdowns and communication cessation. In conclusion, the solutions for WiFi-Direct supported connectivity of medium/large scale networks of mobile off-the-shelf devices are here, but they will only become practical when commodity smartphones/tablets offer a good communication service when operating with both interfaces simultaneously.

REFERENCES

[1] Wi-Fi Alliance. 2014. Wi-Fi Direct, Portable Wi-Fi that goes with you anywhere. (2014). http://www.wi-fi.org/discover-wi-fi/wi-fi-direct accessed: 2017-09-15.

[2] Arash Asadi and Vincenzo Mancuso. 2013. WiFi Direct and LTE D2D in action. In *Wireless Days (WD), 2013 IFIP*. IEEE, 1–8.

[3] Daniel Camps-Mur, Andres Garcia-Saavedra, and Pablo Serrano. 2013. Device-to-device communications with Wi-Fi Direct: overview and experimentation. *Wireless Communications, IEEE* 20, 3 (2013), 96–104.

[4] C. Casetti, C.F. Chiasserini, L. Curto Pelle, C. Del Valle, Y. Duan, and P. Giaccone. 2015. Content-centric routing in Wi-Fi Direct multi-group networks. In *16th Int. Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM '15)*. IEEE Computer Society, 14–17.

[5] Y. Duan, C. Borgiattino, C. Casetti, and M. Turolla. 2014. Wi-Fi Direct Multi-group Data Dissemination for Public Safety. In *World Telecommunications Congress 2014, (WTC)*. VDE, 1–6.

[6] Marco Di Felice, Luca Bedogni, and Luciano Bononi. 2016. The Emergency Direct Mobile App: Safety Message Dissemination over a Multi-Group Network of Smartphones Using Wi-Fi Direct. In *14th ACM Int. Symposium on Mobility Management and Wireless Access (MobiWac'16)*. ACM, 99–106.

[7] Colin Funai, Cristiano Tapparello, and Wendi Heinzelman. 2016. Supporting Multi-hop Device-to-Device Networks Through WiFi Direct Multi-group Networking. *CoRR* abs/1601.00028 (2016).

[8] Le Van Hoang and Hitoshi Ogawa. 2014. A platform for building ad hoc social networks based on Wi-Fi Direct. In *2014 IEEE 3rd Global Conference on Consumer Electronics (GCCE)*. IEEE, 626–629.

[9] Woo Sung Jung, Hyochun Ahn, and Young-Bae Ko. 2014. Designing content-centric multi-hop networking over Wi-Fi Direct on smartphones. In *IEEE Wireless Communications and Networking Conference (WCNC'14)*. IEEE, 2934–2939.

[10] Baburajan Kizhakedath. 2016. Verizon makes $70 million investment for Super Bowl 50 users. (2016). http://goo.gl/QGsJKa accessed: 2017-09-15.

[11] Eugene Marinelli. 2009. *Hyrax: Cloud Computing on Mobile Devices using MapReduce.* Master's thesis. School of Computer Science, CMU.

[12] Ricardo Marinho, Urbano Menegato, and Ricardo Oliveira. 2015. IMSN Routing on Wi-Fi Direct Enabled Devices. In *13th ACM Int. Symposium on Mobility Management and Wireless Access (MobiWac '15)*. ACM, 31–38.

[13] U. Menegato, L. Cimino, S. Silva, F. Silva, J. Lima, and R. Oliveira. 2014. Dynamic Clustering in WiFi Direct Technology. In *12th ACM Int. Symposium on Mobility Management and Wireless Access (MobiWac '14)*. ACM, 25–29.

[14] N. Mizumura, H. Saito, J. Takahashi, and Y. Tobe. 2016. Towards Optimizing Time of File Transfer Among Multiple Smartphones Using Wi-Fi Direct. In *13th Int. Conf. on Mobile and Ubiquitous Systems: Computing Networking and Services (MOBIQUITOUS'16)*. ACM, 189–194.

[15] Matteo Pozza, Claudio Palazzi, and Armir Bujari. 2015. Mobile Data Offloading: An Experimental Evaluation. In *10th Int. Workshop on Mobility in the Evolving Internet Architecture (MobiArch '15)*. ACM, 2–7.

[16] J. Rodrigues, J. Silva, R. Martins, L. Lopes, U. Drolia, P. Narasimhan, and F. Silva. 2016. Benchmarking Wireless Protocols for Feasibility in Supporting Crowdsourced Mobile Computing. In *16th Int. Conf. on Distributed Applications and Interoperable Systems, (DAIS)*. Springer-Verlag, 96–108.

[17] Taylor Soper. 2016. Super Bowl fans use a record 10TB of data on Levi's Stadium wifi network, up to 63% from 2015. (2016). http://goo.gl/hDw7Bw accessed: 2017-09-15.

[18] A. Teófilo, D. Remédios, H. Paulino, and J. Lourenço. 2015. Group-to-Group Bidirectional Wi-Fi Direct Communication with Two Relay Nodes. In *12th Int. Conf. on Mobile and Ubiquitous Systems: Computing, Networking and Services (MOBIQUITOUS'15)*. ACM, 275–276.

[19] S. Trifunovic, B. Distl, D. Schatzmann, and F. Legendre. 2011. WiFi-Opp: Ad-hoc-less Opportunistic Networking. In *6th ACM Workshop on Challenged Networks (CHANTS '11)*. ACM, 37–42.

[20] P. Wong, V. Varikota, D. Nguyen, and A. Abukmail. 2014. Automatic Android-based Wireless Mesh Networks. *Informatica (Slovenia)* 38, 4 (2014), 313–320.

[21] Gergely Zaruba, Stefano Basagni, and Imrich Chlamtac. 2001. Bluetrees-scatternet formation to enable Bluetooth-based ad hoc networks. In *Int. Conf. on Communications (ICC'01)*. IEEE, 273–277.