ScuDo

Scuola di Dottorato ~ Doctoral School

WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation

Doctoral Program in Electronic Engineering (29^{th}cycle)

Cooperative Data Transfers for 5G Networks

By

Yufeng Duan

******

Supervisor(s):

Prof. Paolo Giaccone, Supervisor

Doctoral Examination Committee:

Prof. Ilario Filippini, Referee, Politecnico di Milano

Prof. Alberto Tarable, Referee, Consiglio Nazionale delle Ricerche (CNR)

Politecnico di Torino

2017

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Yufeng Duan
2017

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to my loving parents

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. Paolo Giaccone, who has been guiding me on the research road for many years since my Master thesis. Without him, I would never enter the realm of research. He has been training me for a qualified researcher since the first day and always considers for my future by pointing me out what is valuable for my career.

I would also like to thank Prof. Carla-Fabiana Chiasserini and Prof Claudio Casetti, who have offered great guidance and assistance during my Master thesis and my PhD study. From them, I learned plenty of knowledges and skills and gained a wealth of precious research experiences.

I really appreciate working with the people in the Swarm group from the Joint Open Lab (JOL) Telecom Italia, who has financially supported my PhD study and provided me the equipments and places to conduct my research. As the leader of the JOL, Mr. Claudio Borean provided valuable guidance and advices for the direction of the research project. Pino Castrogiovanni, my tutor in Telecom Italia, has been offering great help in conducting the project. Dario Mana always comes up with very valuable opinions and suggestions during my research.

I would like to thank Prof. Filippini Ilario and Prof. Tarable Alberto who spent a huge amount of time reviewing my thesis. I really appreciate their precious reviews and advices that I have never thought about before.

I am grateful for all the people who have helped me during my PhD study.

Abstract

The demand for higher capacity, higher data rate and larger bandwidth has driven the research and industrial world to develop next generation wireless communication technology, namely, the 5G. Among all the approaches proposed for such a high demand, only the cooperative communication approach promises to significantly improve of the performances (capacity, data rate, bandwidth, etc.) with a low cost. In this thesis, we propose a D2D communication scheme as a solution for the out-door scenario and a cooperative scheme among the access infrastructures as the in-door scenario solution.

In the first part, we address the implementation of content-centric routing in a D2D architecture for Android devices based on WiFi Direct, a protocol recently standardised by the Wi-Fi Alliance. After discussing the creation of multiple D2D groups, we introduce novel paradigms featuring intra- and inter-group bidirectional communication. We then present the primitives involved in content advertising and requesting among members of the multi-group network. In addition to the communications, we also devise a mechanism to enable the devices to spontaneously establish the multi-group D2D network. Finally, we evaluate the performance of our architecture and the network formation mechanism in a real testbed consisting of Android devices.

In the second part, we propose, implement and evaluate a bandwidth aggregation service for residential users that allows to improve the upload throughput of the ADSL connection by leveraging the unused bandwidth of neighboring users. The residential access gateway adopts the 802.11 radio interface to simultaneously serve the local home users and to share the broadband connectivity with neighboring access gateways. Differently from previous works, our aggregation scheme is transparent both for local users, who are not required to modify their applications or device drivers, and for neighboring users, who do not experience any meaningful

performance degradation. In order to evaluate the achievable performance and tune the parameters driving the traffic balancing, we developed a fluid model which was shown experimentally to be very accurate. Our proposed scheme is amenable to efficient implementation on Linux networking stack. Indeed, we implemented it and tested in some realistic scenarios, showing an efficient exploitation of the whole available bandwidth, also for legacy cloud storage applications.

Contents

Chapter 1

Introduction

In the past decade, it has been witnessed the explosion of mobile devices such as smart phones and tablets that require Internet connection every second every where. The deployment of the *third generation* (3G) and the *fourth generation* (4G) wireless mobile telecommunications technology has offered such mobile users an unprecedented Internet experience. In the meanwhile, the success of 3G and 4G also brings more and more subscribes: by the end of 2018, the number of LTE subscribes will reach 1.3 billion [1]. The ever growing data traffic has demand operators and researchers for the development of the next generation mobile broadband technology, i.e., the *fifth generation* (5G).

The 5G technology should provide a better user experience with respect to the previous generation with larger number of and denser mobile users and thus it has envisioned several key requirements. Firstly, given the exponential growth of users, a higher capacity and better coverage has to be guaranteed. Secondly, the 5G technology should offer a high data rate and low latency for both out-door and in-door scenarios. Many popular mobile applications rely on large bandwidth such as video streaming and cloud services while the video calls also require a reasonable low delay. Thirdly, the energy consumption has to be reduced significantly for a sustainable connection. These requirements have driven many studies in the literature and the solutions mainly fall into three categories. The first category of studies focus on new coding and modulation approaches such as non orthogonal multiple access (NOMA) and spatial modulation (SM). The second category of studies try to exploit new frequencies for dedicated applications, e.g., millimeter

wave (mmWave) communication and visible light communication (VLC). The third category of studies mainly focus on the solutions from the point of view of the spatial reuse. The advanced radio access networks (RANs) (e.g., heterogeneous networks (HetNets)) and advanced radio access technologies (RATs) (e.g., new wireless wide area network (WWAN)) and wireless local area network (WLAN) technologies aim at providing auxiliary connections along with the cellular connection in order to improve the overall throughput. Neverthless, each approach alone cannot achieve the ambitious targets of 5G. Instead, only by exploiting a series of those technologies can 5G potentially achieve those targets. However, given the nature of the physics, there is always a theoretical limitation for the categories that manipulate coding and modulation. Besides, the spectrum is not infinite, only a few bands can be exploited for mobile communication. Therefore, the spatial reuse becomes the most important approach that can make the 5G technology fulfill the dreamed performance.

While providing higher data rate and capacity, however, the RANs and RATs approaches also share the same flaw: the existing architecture and infrastructures cannot be utilized by the new introduced technologies and new infrastructures have to be established with a high cost. From the operators point of view, they are more interested in introducing 5G in the existing systems with a low cost. In this thesis, we consider the *cooperative communication*, comprising both the out-door and in-door scenario, that exploits the cooperation among network nodes in relaying information which does not need to pass through the base station (BS), without introducing new central infrastructures. Cooperative communication promises several benefits. Firstly, the small distance between two communicating network nodes in proximity leads to higher signal-to-noise ratio (SNR) and further improves the link reliability and reduces the energy consumption. Besides, as the data is exchanged directly among network nodes without traversing the central infrastructure, the spectral efficiency and system capacity is significantly increased. Finally, the relay among network nodes can also help the ones at the edge of a cell obtain a better connectivity and thus extends the overall transmission range.

As the out-door scenario solution, we consider the *device-to-device* (D2D) communication among the mobile devices which allows two devices in the proximity to communicate with each other directly without the relaying of existing base stations. We provide a fashion of constructing D2D communication networks and content exchanging. In the in-door scenario, we focus on the cooperation of the infrastructures in order to provide a higher throughput for the user devices who do not need

any modifications neither in the drivers nor the applications. We also provide a formulation based on fluid model according to which we can tune the parameters of the network in order to obtain a optimal global performance.

Part I

Wi-Fi Direct Multi-group Networks

Chapter 2

Device-to-Device Communication

It can be argued that the vast majority of wireless communicating devices in use today rely on an Access Point (AP)-based paradigm. Cellular networks, Wi-Fi hotspots, all require user devices to "associate" to a common base station before they can operate. Undeniably, such paradigm is convenient: it facilitates a uniform service provision, it simplifies management and it is essential in case billing is required. At the same time, it creates a cumbersome overhead for communications which, by virtue of the location of endpoints, might best be served by a direct link. Exploitation of Device-to-Device (D2D) connectivity, whether in an unrestrained or in a network-controlled fashion, is at the forefront of standardisation and research efforts. Such interest is spurred by the commercial appeal and widespread availability of Bluetooth Low Energy [2] and Wi-Fi Direct [3], technologies that smartphone and tablet manufactures are increasingly incorporating in their products.

While in the past D2D communication was largely relegated to cable-replacement use cases, today it is touted as a game-changing factor in mass communication, thanks to its enhanced spectral efficiency and traffic offloading capabilities. Some commonly envisioned scenarios for D2D are: machine-to-machine communication, Internet of Things architectures, infrastructure replacement (in case of failure), social content sharing. Additionally, D2D (and Wi-Fi Direct in particular) has the potentiality to play a crucial role in future 5G offloading strategies. LTE standardisation is looking at the interoperability with other D2D technologies by introducing the concept of network-assisted D2D communication: the cellular interface would jump-start the D2D link between suitable devices by handling the discovery and authentication

phases, thus serving as broker party [4–6]. Indeed, D2D has been introduced by 3GPP in LTE Release 12 focusing on providing proximity based services for *public safety* applications.

Several recent studies have investigated the features and the performance of the Wi-Fi Direct technology.

One of the first studies has appeared in [7], where Camps Mur et al. consider a single-group Wi-Fi Direct network with the group owner sharing access to a 3G network with a set of connected devices. The work analyzes the power saving protocols defined in Wi-Fi Direct and design two algorithms that use such protocols to save energy while providing good throughput performance. An improved power management scheme for Wi-Fi Direct is proposed in [8], which dynamically adapts the duty cycle of P2P devices to the properties of the application to be supported.

An overview and experimental evaluation of Wi-Fi Direct using two laptops running Linux is presented in [9], where the emphasis is on the standard group formation procedures and the performance that they exhibit in terms of delay and power consumption. Group formation is also the focus of the work in [10], which investigates the ability to create opportunistic networks of devices using Wi-Fi Direct to establish communication links. The performance of group formation is studied experimentally, by varying the protocol parameters and considering scenarios that are typical of opportunistic networks. A preliminary study of multi-group physical topologies of Wi-Fi Direct networks can be found in our previous work [11], where however only some of the limitations of the Android OS are investigated and only unidirectional D2D communication is tackled.

The use of Wi-Fi Direct as a D2D technology to be integrated into LTE and LTE-A cellular networks is explored in [5, 6, 12]. In particular, while [5] mainly focuses on architectural issues, [6] and [12] also quantify the estimated network performance gains from offloading cellular traffic onto Wi-Fi Direct-based, D2D connections.

As for content dissemination and sharing in mobile ad-hoc networks, a number of solutions have been proposed in the literature, e.g., [13–15]. However, very few works exist that specifically address Wi-Fi Direct-based networks. Among these, the study in [16] presents a Wi-Fi Direct-based overlay architecture for content sharing among peers belonging to the same group. In particular, they leverage the P2PSIP protocol, which enables real-time communication using the application-

layer signaling protocol SIP in a peer-to-peer fashion. The work in [17], instead, implements the decentralized iTrust mechanism [18] for information publication and retrieval. In particular, it proposes a peer management technique to facilitate group creation and allow peers to set up and maintain connectivity over Wi-Fi Direct. Another approach for D2D communication on smartphones is proposed in [19]. It leverages the tethering functionality on smart phones to setup access points. Although this solution does not require rooted devices, as in our proposed scheme, it does not support concurrent inter-group communications. Indeed, in the tethering mode only one 802.11 network interface is locally available and a device cannot operate in two groups simultaneously. To act as relay node between two groups, a node must disconnect from one group and associate to the other, introducing very large latencies in the process (1-10 seconds).

As mentioned, to the best of our knowledge, none of the existing works has investigated, solved and experimentally evaluated bidirectional communication in Wi-Fi Direct multi-group networks.

As complementary approach, LTE Direct [20] enables D2D communications among nodes in the proximity, but, differently from our scenario, it requires the cooperation of the telecom operators. Note also that this technology is still not supported by commercial smartphones.

However, many of the promises in store for D2D communication lay bare what is arguably its biggest flaw: lacking a “static” infrastructure, the availability of content is, at best, spotty and unreliable. Even if requested content is cached by a nearby node, reachable through a multi-hop D2D path, a robust content discovery and retrieval mechanism is needed. Such mechanism should be aware, and, if possible, should leverage the peculiarities of the D2D environment: high node churn, volatile topologies and resource-constrained devices.

In the meanwhile, a multi-group topology is not considered in the literature even though it can provide many benefits. First of all, in order to communicate directly, two devices have to be stay close, which greatly limits the coverage of the network. Therefore, we must enable multi-group communication to extend the network coverage. Besides, if two groups cannot communicate directly, the data exchange among groups will have to go through the central infrastructure, which imposes great burden of traffic forwarding on the central infrastructure. Moreover,

for resource sharing networks, a multi-group topology that involves numbers of nodes can increase the chances that a content can be found and shared.

In this part, we focus on the potentiality of Wi-Fi Direct as D2D communication technology in medium and large-scale scenarios, using open-source, non-rooted Android devices. Our contribution is manyfold.

- We investigate in depth the limitations that the current Android OS exhibits in some crucial Wi-Fi direct features, and in the roles that devices can play in a D2D multi-device topology.
- We work around the above limitations by designing a multi-group, interconnected logical topology that overcomes the limitations of the physical one by exploiting transport-layer tunneling. Such logical topology allows us to enable bidirectional, inter-group data transfers, which would otherwise be impossible in today's Wi-Fi Direct-based networks.
- In order to address the content availability issue, we implement a content-centric routing architecture on our D2D topology. In content-centric routing, users do not need to know the physical whereabouts of data (as in traditional IP routing, in which the hosting device is pinpointed by a univocal identifier), but they just focus on the content they need and let the network do the rest. Routing tables thus carry content-oriented routing information that reflects both (i) the availability of specific content either in the local group of devices or in a nearby, reachable group, and (ii) the above transport-layer tunneling mechanism through which content can be reached.
- We implement a novel content registration/advertisement protocol that is designed to populate Content Routing Tables (CRT) consistently with the data that each user is willing to share (and thus advertises in the D2D network).
- We propose a fully distributed smart group formation mechanism in which devices can spontaneously form a physical multi-group communication network. This mechanism also enables devices to select proper roles in order to create an efficient logical topology for inter-group communications. Notably, during the formation of the network, we devise distributed algorithms run in each device focusing accounting for group sustainability, device energy consumption, average throughput, network coverage, etc.

- We implement this smart group formation mechanism on un-rooted Android devices exploiting the procedures defined in Wi-Fi Direct and the Android Wi-Fi Direct framework.

To our knowledge, our work is the first that tackles bidirectional, inter-group communication in Wi-Fi Direct networks, and proposes and implements a solution to support this data transfer paradigm. Furthermore, we realised a small-scale testbed using off-the-shelf Android devices to test both the feasibility of our multi-group topologies, as well as the efficiency of content-centric routing along with the registration/advertisement protocol. We also test the performance of our smart group formation mechanism in terms of topology creation time and coverage leveraging on this testbed.

The rest of the this part is organised as follows. In Chapter 3, Section 3.1 provides an overview of Wi-Fi Direct. Section 3.2 describes in details the P2P Discovery function defined in Wi-Fi Direct. Section 3.3 highlights some of the limitations that topology formation suffers from in Wi-Fi Direct devices and details the multi-group communication mechanism. Our content-centric routing architecture and registration/advertisement protocol are presented in Chapter 4. Specifically, Section 4.1 introduces two types of data structures we exploit for the content-centric routing and Section 4.2 provides the details of their implementation. Section 4.3 describes our content registration, advertisement and request mechanisms. Then in Section 4.4 we present the message format we designed for the content routing and delivery operations. Section 4.5 illustrates the results derived from our testbed implementation. In Chapter 5, we present our devised mechanism for the formation of the multi-group network and its logical topology. Section 5.1 discusses the design principles of the smart group formation mechanism and presents its details. Section 5.2 describes the implementation of the smart group formation mechanism on un-rooted Android devices. We test this mechanism through both simulation and real testbed and show the results in Section 5.3.

Chapter 3

Multi-group Communication with Wi-Fi Direct

3.1 The Wi-Fi Direct Technology

Wi-Fi Direct is a recent protocol standardized by the Wi-Fi Alliance [3], with the aim to enable D2D communications between nodes, referred to as *peers*. Communication among peers in Wi-Fi Direct occurs within a single *group*. One peer in the group acts as Group Owner (GO) and the other devices, called *clients*, associate to the GO (see, e.g., Fig. 3.1).

Such roles within the group are not predefined, but are negotiated upon group formation. After the GO is elected, the role of each peer remains unchanged during the whole group session. Only when the GO leaves the group, the peers become disconnected and a new group must be created.

The group works as an infrastructure Wi-Fi BSS operating on a single channel, through which the peers communicate. The GO periodically transmits a beacon to advertise the group so as to enable other disconnected devices to discover and, possibly, join the group. The new device exploits the standard Wi-Fi authentication and association procedure to join the Group and becomes a client. As depicted in Fig. 3.1, each client is either a *P2P client* or a *Legacy client*. A P2P client supports the Wi-Fi Direct protocol, whereas a legacy client is a conventional Wi-Fi node that does not support Wi-Fi Direct and "sees" the GO as a traditional Wi-Fi AP. P2P clients and legacy clients coexist seamlessly in the same group. It is important to

note that Wi-Fi Direct has been designed to support D2D communication within a group, however its protocol does not prevent the communication between different groups. Indeed, a peer can act as a bridge between two groups, or between the group and other networks.

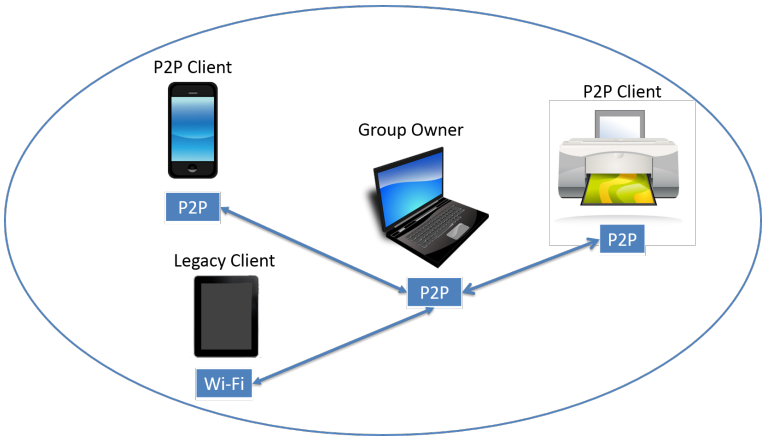


Fig. 3.1 Basic Wi-Fi Direct group with one GO, two P2P clients and one Legacy client.

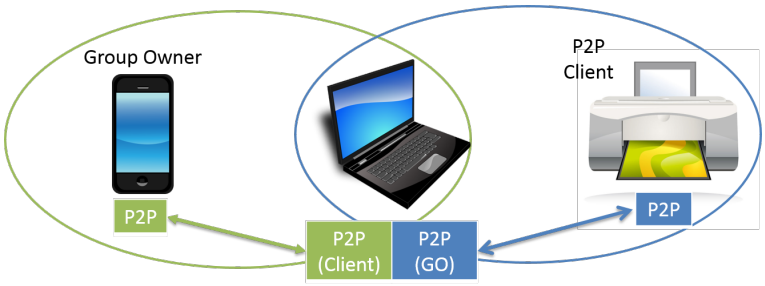


Fig. 3.2 Communication between two Wi-Fi Direct groups.

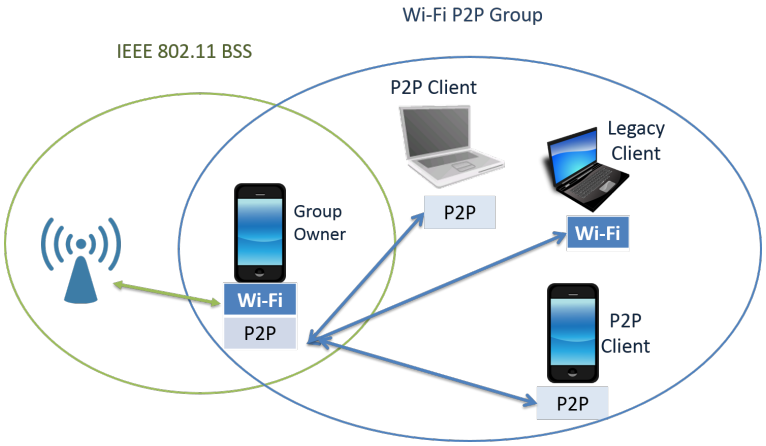


Fig. 3.3 Communication between a Wi-Fi Direct group and a Wi-Fi BSS.

One possible scenario, as shown in Fig. 3.2, consists of a bridge peer (the middle node) behaving as GO for one group and as P2P client in another group. We stress that the bridge peer must support two different MAC entities at layer 2, with two different MAC addresses. A peer can also act as a bridge between a Wi-Fi Direct group and a standard infrastructure BSS. This concurrent operation is shown in Fig. 3.3. Also in this case, the support for multiple MAC entities is required.

In Wi-Fi Direct, devices can quickly find each other and establish connections facilitated by the *P2P Discovery* function. In the following section, we provide the details of this function.

3.2 P2P Discovery

P2P Discovery is a function defined in Wi-Fi Direct that enables devices to find each other, discover nearby services and set up connections. There are several key components contained in P2P Discovery while here we only concentrate on three of them which are related to our work: *Device Discovery*, *Group Formation* and *Service Discovery*.

Before forming a group, devices find each other facilitated by the Device Discovery procedure in which devices can exchange their own information including the device name which is usually a human readable string that helps the users to identify each other. When a device intends to connect to another, following the Group Formation procedure also defined in Wi-Fi Direct P2P Discovery, the two devices have to negotiate the roles, i.e., Group Owner and client, by exchanging an integer value ($[0, 15]$) called *Group Owner Intent* that expresses a measure of the desire to become a GO. Higher values indicate higher desire to be a GO. Each device compares its own GO Intent with the one received from the other to decide the role to take. If a device must be a GO, this device indicates it to the counterpart by setting its Group Owner Intent to 15.

Wi-Fi Direct P2P Discovery also defines the Service Discovery procedure that enables devices to advertise and detect available higher layer services in the vicinity before establishing a connection. All the Devices in the proximity of a service provider can acquire the information of the service, without connecting to the service

provider, which allows devices to set up connections towards dedicated targets instead of forming aimless groups.

In the following we explain the three components in details.

3.2.1 Device Discovery

Device Discovery is aimed at making devices obtain the essential information of each other: the device name, device address, etc. The device name is usually a human readable string that helps the users to identify each other. The device address is the MAC address of the network interface that is function as Wi-Fi Direct of a device.

During device discovery, a device alternates between two state: *listen state* and *search state*. In the listen state, a device listens for Probe Request packets and replies them with Probe Response packets which contain the information of this device we describe above. In the search state, a device sends Probe Request packets (usually in broadcast) to find other devices. A searching device finds another one by receiving the Probe Response packet of that device and thus acquires its device name and MAC address. As all the devices are not synchronized, there is a high probability that the searching state of one device and the listen state of another coincides so that one can find another. After the Device Discovery, a device can select the desired target among the found devices to connect to.

3.2.2 Group Formation

There are two scenarios of Group Formation procedures.

In the common scenario, two devices are involved in the formation. Group Formation in this scenario starts with Group Negotiation which is a three way handshake for the two devices to agree the roles (GO and Client) they are going to take and to agree on several parameters of the Group. As to our work, we only consider how the roles are decided.

The roles of the two devices are decided based on the Group Owner Intent which is an integer value in $[0, 15]$ that expresses a measure of the desire to become a GO. Each device generates its own GO Intent. Higher values indicate higher desire to

be a GO. The algorithm of the generation is not limited and usually depends on the user application. During Group Negotiation, the starting device sends a GO Negotiation Request containing the GO Intent to the target. The receiver replies with the GO Negotiation Response that contains its own GO Intent. As both the two devices ahve received the Intent of each other, they are able to decide the roles to taken by comparing the values of the two Intents. Upon receiving the GO Negotiation Response, the initiator sends back the GO Negotiation Confirmation to the counterpart to conclude the negotiation. After the negotiation, the two devices will conduct the Provisioning procedure that exploits the Wi-Fi Simple Configuration for security reasons. The only point of this procedure we care about is that it requires manual intervention. If the Provisioning succeeds, the two devices are connected and a new Group is formed.

Note that a device can takes part in an existing group using the same procedures above to connect to any peers in the group. To increase the chance of a successful joining, the new device usually joins as a Client by setting its GO Intent to 0, otherwise it may lead to GO transfer which is has a high probability to fail in reality especially when the devices are not in the same proximity.

In the other scenario of Group Formation, a device can autonomously become a P2P Group Owner and set up a group where there is only itself. As explained before, other devices can join this group later.

3.2.3 Service Discovery

Service Discovery (SD) facilitates the devices in advertising and discovering available higher layer services without the need of setting up a connection. Devices can firstly search for desired services and then initiate connections towards the provides, which prevent them from setting up aimless Groups. Similarly to the Device Discovery, in the search state, a device sends SD Query packets to a found device to query for a list of available services or dedicated services. Then this device receives the SD Response from the device that has been just queried which includes the information of the services. The Response Data field in the SD Response is used to store the details of the service but can contain any data the user would like to insert. Indeed, we exploit this field to disseminate the data during the smart group formation described in Section 5.2.

3.3 Communication with Android Devices

As mentioned, we focus on user devices running an open-source Android OS due to their wide popularity, and we investigate how to provide bidirectional multi-group communication in networks composed of such devices. Android devices offer a limited, controlled set of networking capabilities for security reasons. It is of course possible to "root" a device in order to access advanced capabilities, but we do not take this possibility into account since the rooting process requires skills that are beyond the average user, and it renders the warranty null and void. Thus, we only act upon application-layer functionalities, i.e., no changes can be performed at the transport or network layer (like changing IP addresses for P2P interfaces, configuring routing tables, etc).

A multi-group topology could be implemented by letting a device have two virtual P2P network interfaces: in this way, it could act as a bridge using a different MAC entity in each group. In non-rooted Android devices, however, the programmer cannot create a custom virtual network interface. Our experiments revealed that none of the following scenarios are feasible in Android, much though they are not expressly forbidden by the standard:

1. a device plays the role of P2P client in one group and GO in another,
2. a device behaves as the GO of two or more groups,
3. a device behaves as client in two or more groups.

Thus, in order to create a multi-group physical topology (i.e., bridge nodes), we let a GO be a legacy client in another group. Specifically, we proceed as depicted in Fig. 3.4, where three inter-connected groups are formed with six devices. GOs are represented by circles and clients by squares. In each peer, we enable two network interfaces, one of which is the conventional Wi-Fi interface and the other (P2P) is used for Wi-Fi Direct connection. The interfaces used to form a group are highlighted using the same color, while connections are represented by lines. It is important to remark that each group represents a different Wi-Fi Basic Service Set (BSS). Furthermore, note that GO2 and GO3 also act as legacy clients of GO1 and GO2, respectively. GO1 is not acting as a legacy client since it is not associated to any other group. As discussed later in Section 3.3.2, the fact that one GO is a legacy client of another GO affects its forwarding capabilities.

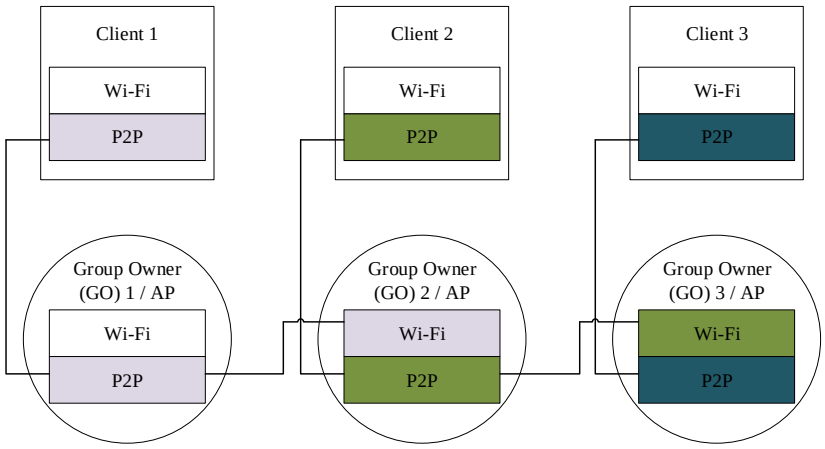


Fig. 3.4 Multi-group physical topology with six devices (three clients and three GOs). GO2 and GO3 are bridge nodes, i.e., they are legacy clients of GO1 and GO2, respectively.

For ease of presentation, in the following we often take the three-group physical topology depicted in Fig. 3.4 as reference scenario and refer to the three groups as Group 1, Group 2 and Group 3, respectively.

3.3.1 IP address assignment

In Android devices, once a Wi-Fi Direct connection is established, the GO automatically runs the DHCP to assign IP addresses to itself (192.168.49.1/24) as well as to the P2P clients or legacy clients in its own group (192.168.49.*x*/24 where x is a random number $\in [2, 254]$ to minimize the chance of address conflicts). Therefore, the P2P interfaces of all GOs have the same IP address, namely 192.168.49.1. The Wi-Fi interfaces of the GOs that act as legacy clients in another group are assigned an IP address in the format 192.168.49.*x*/24. Similarly, P2P interfaces of clients are assigned different IP addresses in the format 192.168.49.*x*/24.

An example of IP assignment for the three-group topology is shown in Fig. 3.5, which highlights the address conflicts for the P2P interface of the GOs. Since GO1 is not associated with a Wi-Fi AP, no IP address is assigned to its Wi-Fi interface.

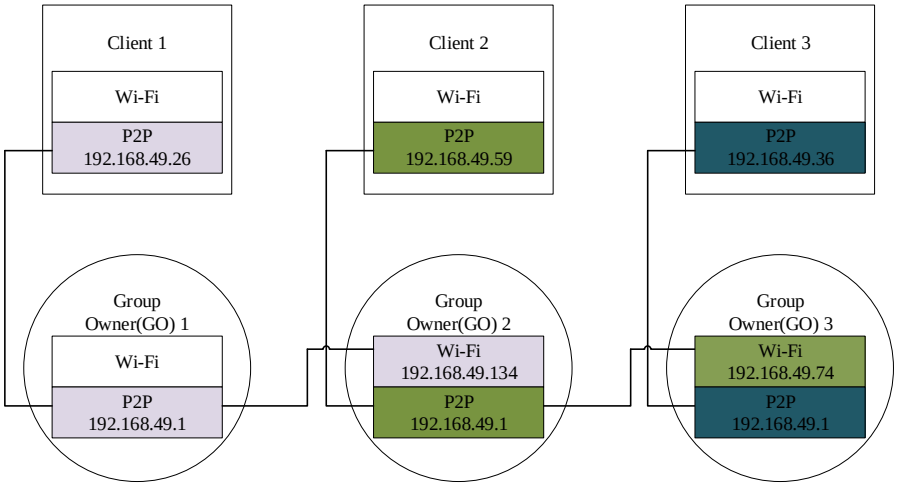


Fig. 3.5 Example of IP addresses (/24) for multi-group configuration.

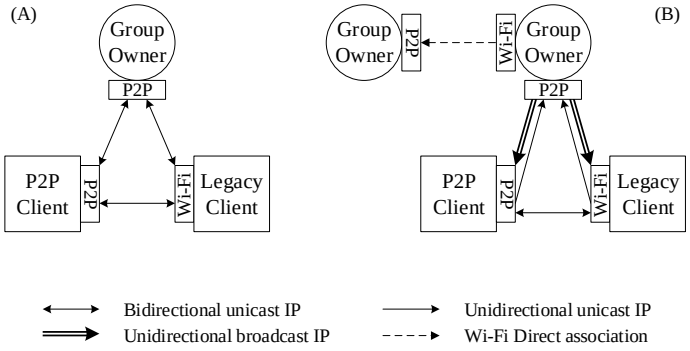


Fig. 3.6 D2D intra-group communications: (A) in an isolated group, (B) in a group whose GO is a legacy client in another group.

3.3.2 Design of the logical topology

Given the above assignment of IP addresses, we show how to design a *logical topology* that implements multi-group communication. Our methodology overcomes the limitations of the physical topology and of its addressing plan, which prevent data transfers on some D2D links.

Let us start by discussing intra-group communication, as it is the basis for enabling bidirectional inter-group communication. Two cases have to be distinguished.

In the first case, depicted in Fig. 3.6(A), the GO is not connected to any other group as legacy client. Since Wi-Fi Direct has been designed to provide full connectivity among all nodes of an isolated group, all possible D2D communications are

enabled. Thus, any pair of devices (GO, P2P clients and legacy clients) can exchange data at the IP layer. Note that, in the specific example in Fig. 3.5, Group 1 falls in this case (hence all D2D communications are allowed) since GO2 is a standard legacy client as far as GO1 is concerned.

In the second case, illustrated Fig. 3.6(B), the GO is also connected to another group as a legacy client. Referring again to the example network in Fig. 3.5, Group 2 and Group 3 fall in this case. All D2D unicast data transfers among clients (P2P or legacy clients) are allowed, thus TCP connections and/or UDP flows between clients are supported. Instead, between two GOs, or between a GO and its clients, *only a subset of D2D data transfers are allowed*. The reasons underlying this limitation are two. First of all, two neighbor GOs cannot communicate directly, because of the IP address conflict. Note that in this case one of the GOs acts as legacy client of the other GO, as in the example of Fig. 3.5 where GO2 is legacy client of GO1. When GO2 wishes to transmit an IP packet to GO1, the destination is set to 192.168.49.1 and the packet is thus sent to its local loop and not to the Wi-Fi interface. Also, when GO1 sends an IP packet to GO2 (192.168.49.134), GO2 discards it since its IP layer detects that the packet source address matches its own (192.168.49.1). The second reason pertains to the ordering of routing table entries in the GO, as implemented by the Android OS. When the GO wants to send a unicast IP packet to any client of its group, the packet is invariably sent through the GO Wi-Fi interface, since the latter entry is listed with higher priority than the P2P interface in the routing table of the device[1]. In the client-to-GO direction, instead, the communication is allowed since client routing tables list only one interface and no conflict occurs. In summary, *bidirectional unicast data transfer between GO and its clients is not allowed*, only unidirectional unicast communication between the client and the GO can take place. Hence, no TCP connection can be established between the GO and its clients, whereas UDP flows are allowed only from the clients towards their own GO.

Conversely, broadcast IP packets sent by the GO are always[1] sent through its P2P interface. This is an important observation as it allows the support of bidirectional data transfer between each client and its GO: broadcast IP packets can be used from the GO to the clients, while unicast IP packets can be adopted to transfer data from the clients to the GO. Note that broadcast packets generated by the GO will

[1]We consistently observed this behavior for different devices, of different brand, running Android 4.3 and 4.4.

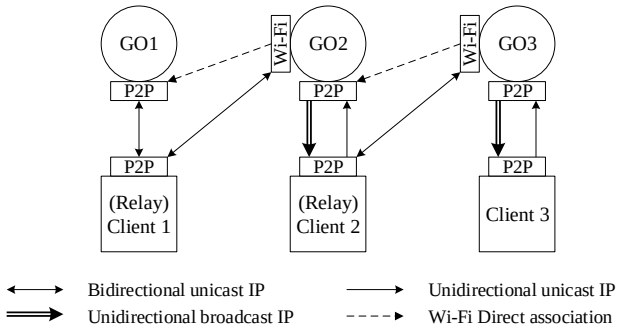


Fig. 3.7 D2D inter-group communication. The picture refers to the example network with three groups and a linear topology. The P2P clients in Groups 1 and 2 are used as relays to reach the right side group (Group 2 and 3, respectively). GO2 and GO3 are used as relays to reach their left side group (Group 1 and 2, respectively). The adopted line style follows the convention of Fig. 3.6.

also reach the GOs associated to it as legacy clients, but then such packets will be discarded because of the conflict of source IP address, as discussed above. So, it is not possible for a GO to directly reach neighbor groups. Lines connecting the nodes in Fig. 3.6(B) summarize the possible intra-group data transfers at IP level.

We can now focus on enabling inter-group communication in light of the issues discussed above. We recall that D2D communications are allowed between any two clients within the same group (i.e., not involving the GO at IP layer). Thus, also the communication between a P2P client and a legacy client that is also the GO of a different group is allowed in both directions. This observation is crucial, since it provides support for our novel design that exploits *a client within the group as relay to reach a neighbor group*. Specifically, we provide bidirectional, inter-group communication between neighbor groups by adopting the communication scheme shown in Fig. 3.7. To send data from the central group (Group 2) to its right side group (Group 3), we leverage a P2P client (client 2) to relay the traffic toward GO3. Instead, to send data from Group 2 to its left side group (Group 1), GO2 itself is responsible to relay traffic toward a client in the left side group (client 1). In other words, we build a logical topology based on transport-level tunnels enabled by IP and MAC-layer connectivity, as follows.

- Unidirectional UDP tunnels between a GO and its P2P clients (e.g., GO1 and client 1). They are based on broadcast IP packets from the GO to clients and on unicast IP packets from clients to the GO. When reliable communication is

required towards a single client, the GO can adopt a classical stop-and-wait protocol.

- Bidirectional UDP or TCP tunnels between P2P clients and legacy clients within the same group (e.g., between client 1 and GO2, or client 2 and GO3).

Full connectivity among nodes in a multi-group network can thus be provided by leveraging a proper sequence of transport-layer tunnels established in the logical topology, and switching packets at the application layer (i.e., *without rooting the devices*).

3.3.3 The role of the relay client

To define a routing process that properly leverages the above transport-layer tunnels, we select one client within each group, to act as a relay node with respect to neighbor groups. We name such node *relay client*. In the example in Fig. 3.5, client 1 (client 2) is the relay client connecting Group 1 (Group 2) to Group 2 (Group 3).

We implemented a basic election scheme at the application layer; more sophisticated solutions could be devised so as to design smart network topologies. According to our scheme, the GO sends a message to one of its clients, chosen at random among those that do not act as GO in another group, to elect it. To reach the desired client, the message is sent via a broadcast IP packet through the P2P interface. Indeed, if a unicast IP packet were used, it could be wrongly sent to the Wi-Fi interface (specifically, when the GO is also legacy client in another group). Note that the role of each client in the group, as well as in other groups, is known to its GO through application-layer signalling (see Section 4.3).

3.3.4 The communication backbone

To disseminate data across a large set of devices, we then propose a logical tree topology, connecting all groups by extending the approach shown in Fig. 3.7 to an arbitrary number of groups. By doing so, we build a communication backbone, as depicted in Fig. 3.8. The figure highlights in grey the GOs and the relay clients that compose the backbone and provide connectivity to all other clients (P2P and Wi-Fi clients that do not act as GOs, i.e., that are not involved in the traffic relay process).

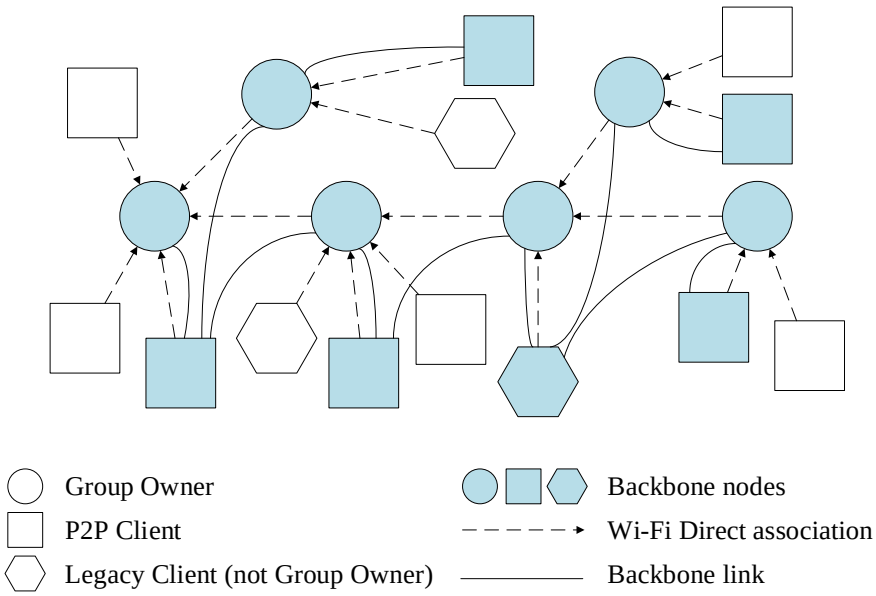


Fig. 3.8 Communication backbone over an arbitrary network topology.

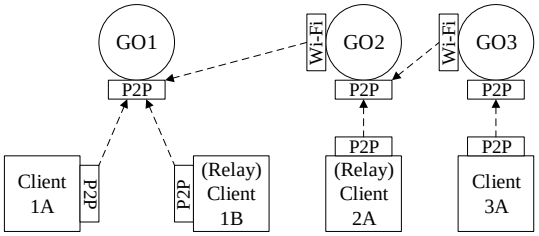


Fig. 3.9 Example scenario with 7 nodes, distributed into 3 Wi-Fi Direct groups. We only show the associations at the Wi-Fi Direct level.

In principle, our approach might scale indefinitely, even if we were able to validate it experimentally only for few groups, as shown in Section 4.5.

It is important to remark that a path over the backbone involving transfers from GO to relay client within the same group requires a broadcast IP transmission for each of such transfers. Instead, transfers from relay client to GO do not require any broadcast IP transmission.

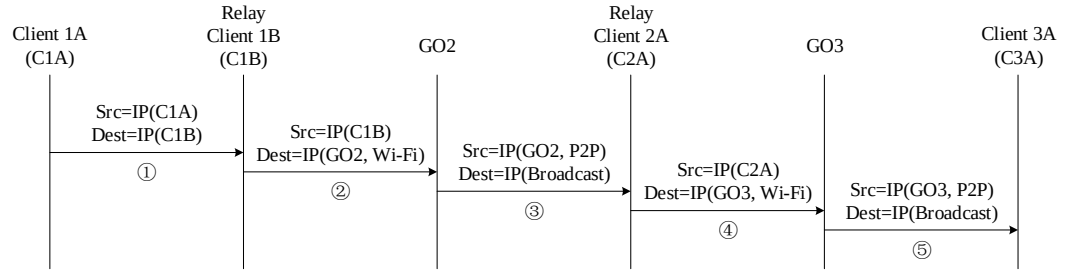


Fig. 3.10 Packet delivery at IP layer in the example network in Fig. 3.9. The packet transfer occurs from client 1A to client 3A. The number is the IP packet identifier.

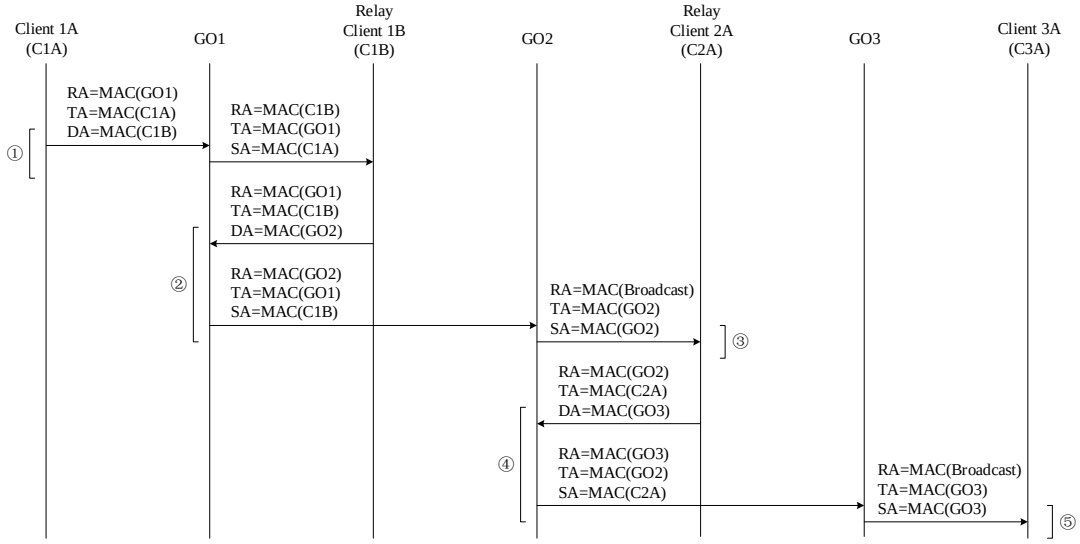


Fig. 3.11 Packet delivery at MAC layer referring to a data transfer from client 1A to client 3A (see Fig. 3.9); for clarity, IP packet identifiers are also reported. Wi-Fi addresses are: TA=transmitter, RA=receiver, SA=source, DA=destination.

3.4 A Step-by-step Example

For the sake of clarity, here we report a detailed description of the packet forwarding process in the scenario depicted in Fig. 3.9, when a packet flows from client 1A to client 3A, and viceversa, using UDP as transport layer. To help elaborate the procedure clearly, the message exchange at IP and MAC layers are shown in Figs. 3.10-3.13. Note that, since our focus is on content-centric routing, here we assume devices to be aware of the next-hop to contact in order to reach a content. The selection of the next-hop is determined by the sequence of transport-layer tunnels

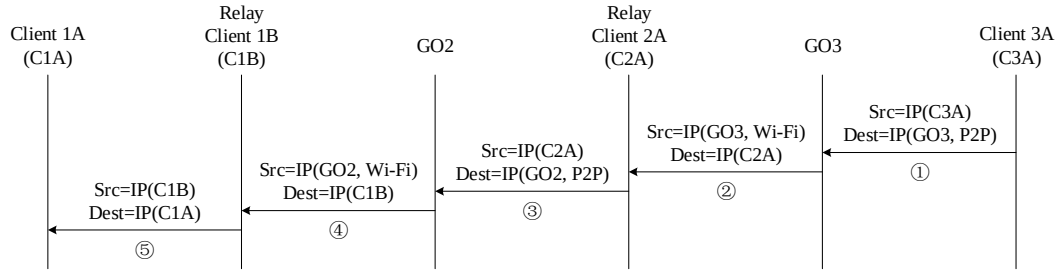


Fig. 3.12 Packet delivery at IP layer, referring to a transfer from client 3A to client 1A (see Fig. 3.9). The number is the IP packet identifier.

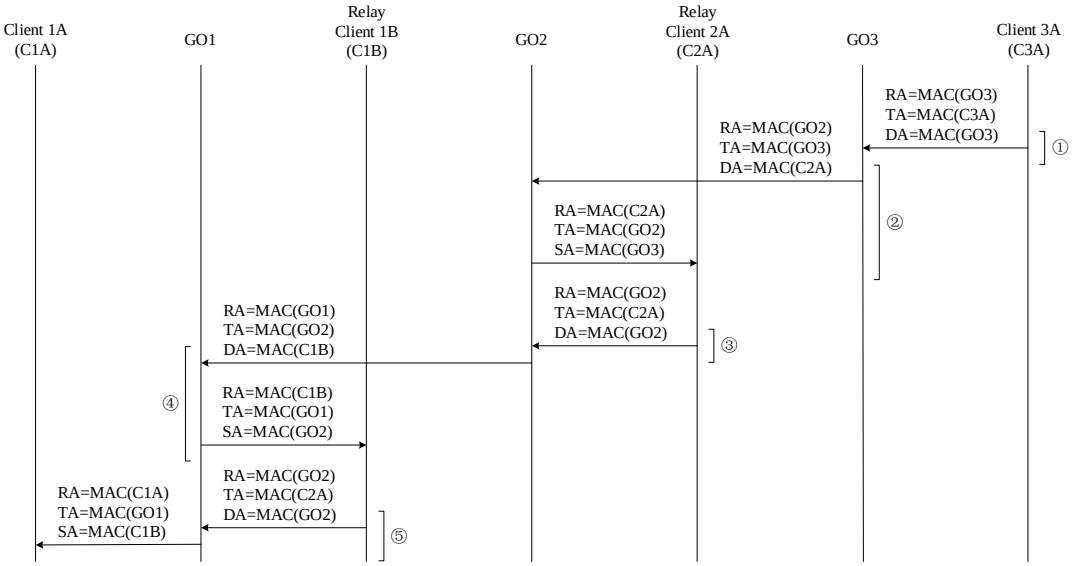


Fig. 3.13 Packet delivery at MAC layer, with IP packet identifiers. The diagram refers to the data transfer from client 3A to client 1A. Wi-Fi addresses are: TA=transmitter, RA=receiver, SA=source, DA=destination.

composing the network backbone. For instance, the path from client 1A consists of the following hops: client 1A $\rightarrow$ client 1B $\rightarrow$ GO2 $\rightarrow$ client 2A $\rightarrow$ GO3 $\rightarrow$ client 3A. The next section explains the acquisition of routing information by devices.

Below, we detail the steps that let a packet be delivered from client 1A to client 3A.

1. client 1A encapsulates the data in the payload of a unicast UDP packet and sends it directly to the relay client 1B. This packet is sent at MAC layer to GO1, which behaves as an AP and re-sends it to the relay client.

2. client 1B processes the packet at the application layer and duplicates the payload into a new UDP packet, sent directly to the IP of the Wi-Fi interface of GO2. At MAC layer, the packet is sent to GO1, which re-sends it to the desired GO2 interface.

3. GO2 processes the packet at the application layer and duplicates the payload into a new UDP packet. The UDP packet is sent as a broadcast IP packet through the P2P interface of GO2, thus reaching the relay client in Group 2.

4. client 2A processes the packet at the application layer and duplicates the payload into a new UDP packet, to be sent directly to the IP address of the Wi-Fi interface of GO3.

5. Finally, GO3 processes the packet at the application layer and duplicates the payload into a new UDP packet, sent directly to destination client 3A.

In case of a packet flowing from client 3A to client 1A, the following procedure takes place. For brevity, only IP layer packet transfers are highlighted.

1. client 3A encapsulates the data in the payload of a unicast UDP packet and sends it directly to GO3.

2. GO3 (which is also a legacy client in Group 2) processes the packet at the application layer and duplicates the payload into a new unicast UDP packet, sent to the relay client 2A.

3. In its turn, client 2A processes the packet at the application layer and creates a new UDP packet destined to the P2P interface of GO2.

4. GO2 (which is also a legacy client in Group 1), again, creates a new UDP packet and sends it to the relay client 1B.

5. Finally, client 1B forges a new UDP packet for the final destination, client 1A.

Note that, in accordance with our discussion in Section 3.3.4, the first example above implies two IP broadcast transmissions (i.e., transfers from a GO to a relay client

within the same group), while the second example does not involve any broadcast IP packet.

Chapter 4

Content-centric Routing on Multi-group Networks

We propose a network architecture in which content delivery leverages the above forwarding scheme through a content-centric approach.

We assume that each node knows the neighbor node (next hop) to which it has to send the request for a specific content. How this knowledge is acquired is explained later in this chapter. When the request reaches the node with the desired content through a sequence of transport-layer tunnels, the content data is forwarded back to the requester, along the same path (i.e., sequence of tunnels) followed by the request packet. Note that this scheme is compatible with possible caching solutions adopted at intermediate nodes; however, for ease of presentation, we will assume that each content is provided by exactly one node in the network and that, when such node disconnects, the content becomes unavailable.

4.1 Data Structures

Two main data structures are responsible for content routing, as detailed below.

The *Content Routing Table* (CRT) provides the routing information to reach content items. For each item, identified by the MD5 hash of its name, the CRT stores the IP address of the next-hop node to reach the content provider, similarly

to the "gateway" field of a traditional IP routing table. Note that the next-hop node definition must be tailored to the data forwarding scheme described in Section 3.3.

The second data structure is the *Pending Interest Table* (PIT), derived from CCN [21], which provides the information to route a content to the requester. Before an intermediate node forwards a content request, the node stores the IP address of the node interface from which the request was received. Devices thus record the "previous hop" for each requested content item and forward the content back to the requester upon receiving it; then, the corresponding entry is removed from the PIT.

The detailed implementation of the CRT and PIT is discussed in next section.

4.2 Implementation of the Routing Protocol

Let us examine the possible next-hop values that can be associated to a specific content item. To this end, we consider two neighbor groups, G and $\hat{G}$, and denote by RC(G) the relay client of group G and by GO(G) the GO of G; a similar notation is used for devices in $\hat{G}$. Let us assume that the user requesting the content is in group G. The CRT entry on the requesting user device will associate the following possible next-hop values to the requested item.

1. The content item is available in G. Then, the next hop is set as the IP address of the client in G providing the content.

2. The content item is available in $\hat{G}$, and $\hat{G}$ is reachable through RC(G). In other words, GO($\hat{G}$) is a bridge node (i.e., it is also a legacy client in G), and, according to our forwarding scheme, RC(G) can relay traffic to it. Thus, the next-hop for the requesting client, and for all group members with the exception of GO($\hat{G}$), is RC(G). The next-hop for RC(G) is the IP address associated with the Wi-Fi interface of GO($\hat{G}$). In the example of Fig. 3.9, for a content available in Client 3A, the next-hop for Client 1A and GO1 is the relay client 1B; the next-hop for Client 1B is the Wi-Fi interface of GO2.

3. The content item is available in $\hat{G}$, and $\hat{G}$ is reachable through GO(G), i.e., GO(G) is a bridge node as it acts as legacy client in $\hat{G}$. The next-hop for the requesting client, and for all members of G, is the IP address of the P2P interface of GO(G). The next-hop of GO(G) is the IP address of RC($\hat{G}$). In

the example of Fig. 3.9, for a content available at Client 1A, the next-hop for Client 3A is the P2P interface of GO3; the next-hop for GO3 is Client 2A.

If no match is found in the CRT of an intermediate node, the packet is discarded and a notification message is returned to the requester.

The information present in the CRT is updated only when new content becomes available, or a content item becomes unavailable, according to the protocol described in Section 4.3. Note that the proposed scheme can be easily extended with a weighted list of next-hops to support multiple copies of the same content, in case of cooperative caching mechanisms implemented in the network. Furthermore, standard methods to aggregate content routes in the CRT can be implemented to reduce the CRT size and propagate differential updates. Such methods are complementary to our scheme and out of the scope of this thesis.

In the PIT presented at a given device, there may be multiple pending requests for the same item. In this case, when the item reaches the device, the latter forwards it toward all requesting devices, duplicating it and sending it over the previous hops from which the corresponding requests were received. A timeout is set to remove pending requests for unavailable content. A content received by an intermediate node without any matching entry in the PIT is discarded.

The flow chart in Fig. 4.1 summarises the packet processing at an intermediate node of the communication backbone (relay client or GO), when a content data packet or a content request packet is received.

4.3 Content Registration, Advertisement and Request

To build the CRT, we adopt a simple protocol based on the following two phases. *Content registration* is the initial phase in which a client advertises the availability of new content within the group. The message is sent from the client to the GO, which returns an acknowledgment (ACK), guaranteeing reliable registration of content. *Content advertisement* is the subsequent phase in which content is advertised internally and externally to the group.

First, the GO sends a broadcast message to all (P2P and legacy) clients, to update their CRT and waits for an ACK from the relay client. Thanks to its broadcast

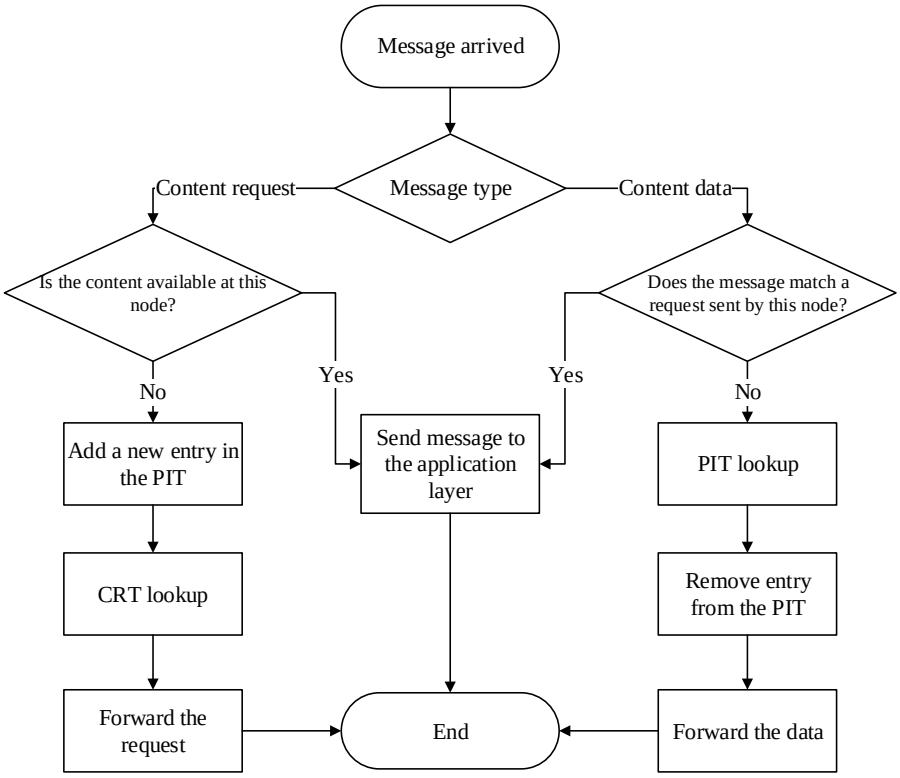


Fig. 4.1 Packet processing at each intermediate node of the communication backbone (relay client or GO).

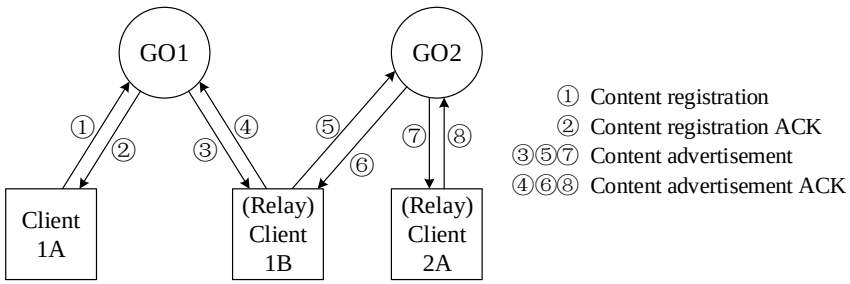


Fig. 4.2 Message exchange triggered by a new content available in Client 1A.

nature, a single message is needed, regardless of the number of clients in the group. However, reliable reception is guaranteed only for the relay client. The broadcast message sent by the GO is discarded at the IP layer by the legacy clients that are GOs of other groups; thus, it will not propagate outside the group where it has

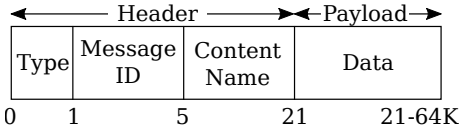


Fig. 4.3 Application-layer message for content registration, advertisement, request and delivery.

been generated. In order to advertise the content to other groups, the relay client sends a content advertisement message to each legacy client that is also a GO of another group, and waits for the ACK. The example of Fig. 4.2 shows the sequence of application-layer packets that are exchanged when new content becomes available at Client 1A. After message ⑧, all nodes in the two groups have updated their own CRT with the new content item.

Based on the above procedure, updated content information can be (surely) found only at the GOs and at the relay clients. Thus, upon generating a content request, a device that is neither a GO nor a relay client, first looks up the content information in its CRT. If no entry is available, it sends the request to its GO, which will process it as described in the previous section. When delivering the content, each node along the path forwards the data packet from the provider to the next stop according to the locally managed PIT. Under the help of all the intermediate nodes in the path, the content is finally delivered to the requester.

4.4 Message Format

Our content-centric routing is based on one application-layer message, which may carry control information (e.g., content registration and advertisement), a content request, or the desired content item. The message is encapsulated in standard TCP or UDP segments, carried by IP packets. Fig. 4.3 reports the message format:

- Type (1 byte) specifies the message type, among: Content registration (and ACK), Content advertisement (and ACK), Content data, Content request, Relay election (and ACK), notification of GO role in another group by a legacy client (and ACK).
- Message identifier (4 bytes) is a random nonce that associates the ACK with the message it refers to.

- Content name (16 bytes) is the MD5 hash of the content name. Note that any other hash function could be used to encode the name.
- Data (0-64 kbytes) is the main payload, carrying control or content data.

4.5 Experimental Evaluation Through Content-centric Routing

4.5.1 Testbed

We setup a testbed including several Android devices of different type, namely, Google Nexus 7 and ASUS Transformer Pad TF300 tablets and 2 different smartphones (LG P700, Sony Xperia Miro ST23i). The Nexus tablets were equipped with Android 4.4.2 (API level 19), but our application was also tested with Android 4.3 (API level 18) on the same devices, before the operating system upgrade. LG smartphones used Android 4.0 (API level 14), which is the oldest version supporting Wi-Fi Direct. In our tests, LG smartphones acted as P2P clients and never as GOs, since the transport-layer tunnels from/to the GO discussed in Section 3.3.2 are fully enabled only for Android 4.3 and later versions. The ASUS tablets and the Sony Xperia were equipped with Android 4.2.1 (API level 17) and Android 4.0.4 (API level 14), respectively. Neither of them support Wi-Fi Direct; we used such devices only as legacy clients and not as group owners. This variety in the choice of devices allowed us to validate our multi-group communication mechanism in presence of heterogenous devices and different conditions. *No device was rooted*, to be sure that we could validate the approach for off-the-shelf devices.

We developed an Android application to implement our solution for bidirectional, multi-group communication and content-centric routing, as well as to validate the whole approach and assess its performance. In order to program the devices, we used the integrated development environment (IDE) Eclipse (version v22.0.1) with the ADT (Android Developer Tools) on Ubuntu 13.04. The ADT is officially provided by Google and allows users to build, test, and debug applications on Android.

For brevity and ease of presentation, in the following we show the results that we obtained using the experimental setup depicted in Fig. 4.4, i.e., two Wi-Fi Direct groups. Group 1 includes 4 devices (GO1, Client 1A, Client 1B and GO2, the latter

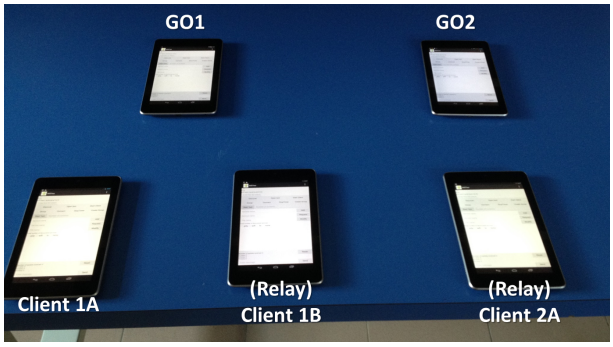


Fig. 4.4 Testbed setup with five tablet devices, forming two Wi-Fi Direct groups.

acting as legacy client in Group 1), while Group 2 comprises 2 devices (GO2 and Client 2A). Client 1B and Client 2A operate as relay clients in their own groups. This setup is equivalent to the scenario discussed in the example of Fig. 4.2.

All the tablets were located in proximity of each other, to reduce the effects of propagation delays and signal attenuation due to distance. All experiments have been carried out in the laboratory, during evening hours to reduce interference from active neighbor APs. We manually chose channel 11 for Wi-Fi and Wi-Fi Direct communications, since, according to a preliminary monitoring of the spectrum, it was the least interfered channel.

4.5.2 Experimental results

We tested our inter-group communication and content sharing in two phases. In the initial phase, we investigated the performance of the content delivery scheme (i.e., of the forwarding mechanism through transport-layer tunnels), while in the second phase we compared the performance to that obtained when the Bluetooth technology is used.

Content delivery performance

Here, we focus on the performance that can be achieved for the content data transfer, from one device to another, based on the data delivery scheme explained in Section 3.3. We manually configured the CRT and PIT tables to avoid any protocol overhead due to content requests and table updating. Each content is divided into chunks of fixed size equal to 1400 bytes, to avoid IP fragmentation. To vary the

offered traffic load, the content provider periodically sends a new chunk, encapsulated into a Content Data message, with the chunk rate being a varying application parameter.

We validated the data delivery mechanism by picking different pairs of devices among the possible ones, and letting them act as source-destination nodes. We therefore verified the full bidirectional connectivity over the whole multi-group network, and recorded the application-layer throughput and the packet losses experienced at the IP layer, as functions of the application-layer traffic offered load. For each configuration, we run 100 different experiments, to obtain throughput results with a 1% relative width of the 95% confidence interval.

In the following, we mainly focus on the scenarios detailed below, run on the testbed shown in Fig. 4.4:

1. “2 devices - 1 group” (2d1g), in which the source is Client 1A and the destination is GO1. The communication between a client and its GO involves just one hop at IP and MAC layer, since each message is sent through a single unicast IP packet, carried by a single MAC frame.

2. “3 devices - 1 group” (3d1g), in which the source is Client 1A and the destination is Client 1B. The communication between two clients in the same group involves one hop at the IP layer, but two hops at the MAC layer (Client 1A $\rightarrow$ GO1 $\rightarrow$ Client 1B).

3. “4 devices - 2 groups” (4d2g), in which the source is Client 2A and the destination is Client 1B. The communication between the two clients in 2 groups requires two hops at IP layer (Client 2A $\rightarrow$ GO2 $\rightarrow$ Client1B) and three hops at MAC layer (Client 2A $\rightarrow$ GO2 $\rightarrow$ GO1 $\rightarrow$ Client 1B).

4. “2 devices - 1 group - broadcast” (2d1g-B), in which the source is GO2 and the destination is Client 2A. The communication within the same group now occurs in the opposite direction with respect to the 2d1g case, but notably the single-hop communication is based on a broadcast transmission, since GO2 is also legacy client of GO1.

5. “4 devices - 2 groups - broadcast" (4d2g-B), in which the source is Client 1B and the destination is Client 2A. The communication between the two clients in two different groups involves 2 hops at IP layer (Client 1B $\rightarrow$ GO2 $\rightarrow$

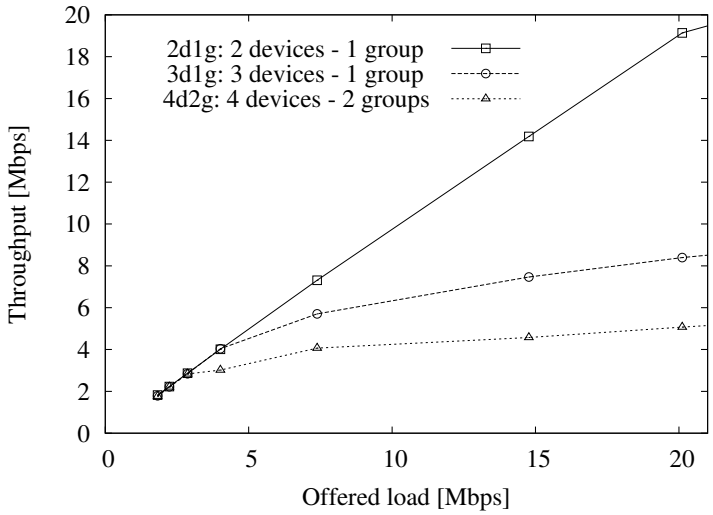


Fig. 4.5 Throughput at application layer as a function of the offered traffic load, for packet transfers involving only unicast transmissions.

Client 2A) and 3 hops at MAC layer (Client 1B $\rightarrow$ GO1 $\rightarrow$ GO2 $\rightarrow$ Client 2A) in which the last hop is based on a broadcast transmission.

Note that we do not show the case of content transfer from GO1 to Client 1A since it is equivalent to the 2d1g case; indeed, GO1 is not a legacy client of any other group, thus it can send unicast IP packets directly to Client 1A.

For fair comparison, we start by evaluating the first three cases, which imply only unicast transmissions; then, we will move on to the last two cases, involving broadcast transmissions.

Fig. 4.5 shows the application-layer throughput vs. the offered load. As expected, the throughput increases with the load, and reaches a maximum value of about 19 Mbit/s (2d1g scenario), 8.4 Mbit/s (3d1g scenario) and 5.0 Mbit/s (4d2g). These results are coherent with the fact that the throughput decreases proportionally to the number of hops, due to the channel contention among the transmitters operating on different hops. Note that current available Wi-Fi Direct interfaces work only on a single frequency channel and, thus, the whole multi-group network is part of the same collision domain. In general, the number of hops traversed by a packet depends only on the distance, in terms of number of groups, over the backbone between source and destination (usually, we have two hops at the MAC layer per each traversed group), whereas it is independent of the total number of devices

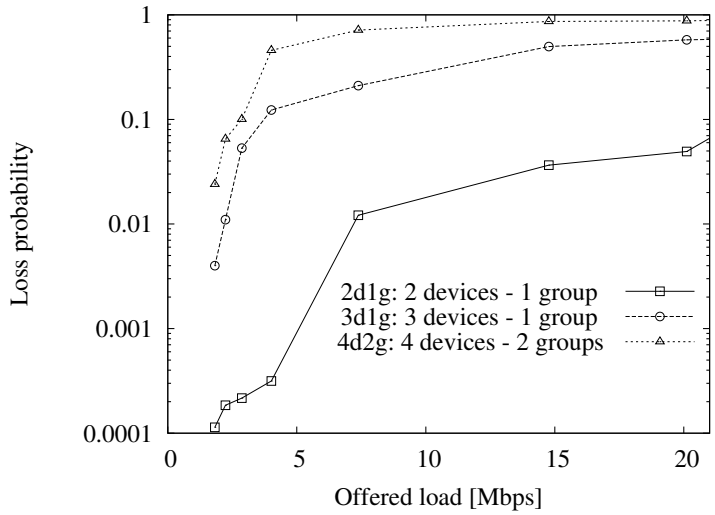


Fig. 4.6 Packet loss at IP layer vs. offered traffic load, for packet transfers involving only unicast transmissions.

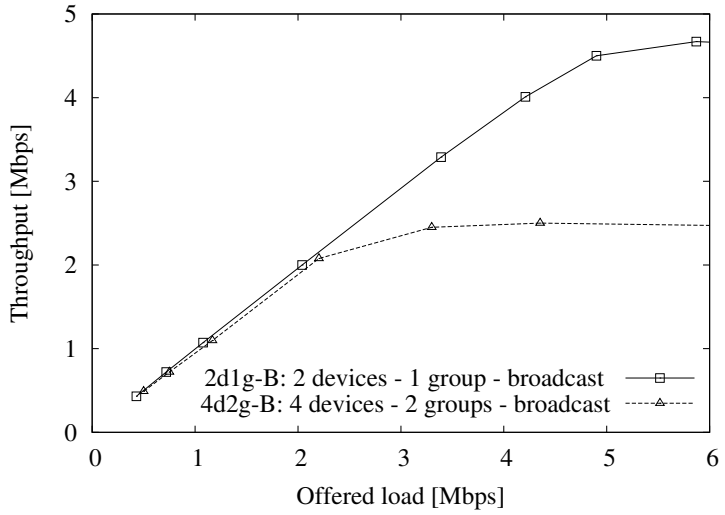


Fig. 4.7 Throughput at application layer as a function of the offered traffic load, for packet transfers involving also broadcast transmissions.

composing the network. While the single collision domain increasingly affects the performance as the number of active transmitters grows, having the hop number independent of the group size improves scalability. Additionally, the impact of the single collision domain lessens as the network gets larger: when transmitters are far away from each other, some degree of spatial diversity is possible and interference

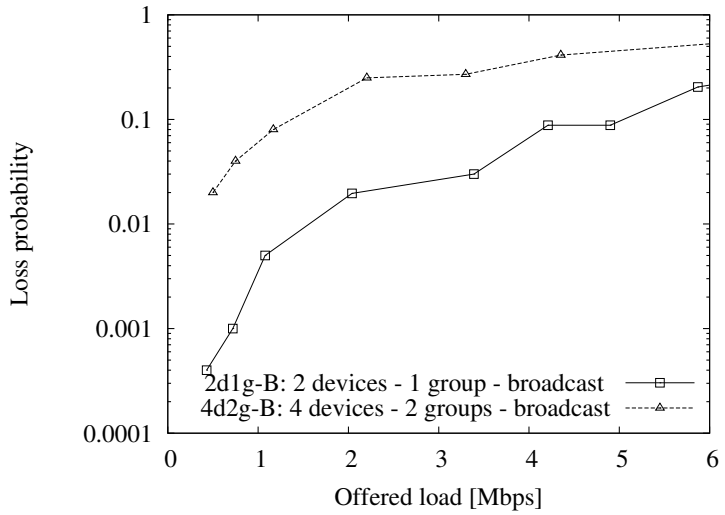


Fig. 4.8 Packet loss at IP layer vs. offered traffic load, for packet transfers involving also broadcast transmissions.

among parallel transmissions greatly reduces. It follows that the network throughput decreases less than proportionally to the number of hops.

Fig. 4.6 shows the overall packet loss probability. Note that packet losses are almost negligible in the 2d1g scenario. They become noticeable for the 3d1g case (0.4% for the lowest load) but still have little impact on the throughput. Under the 4d2g scenario, instead, packet losses are more significant (2.4% for the lowest load), since the transmissions over the three hops that every message has to undergo at the MAC layer interfere with each other.

Figs. 4.7 and 4.8 depict throughput and packet losses, respectively, for the last two scenarios, 2d1g-B and 4d2g-B, both implying one broadcast transmission by GO2. The maximum throughput is 4.6 Mbit/s for 2d1d-B and 2.5 Mbit/s for 4d2g-B. Such numbers are much smaller than in the first three scenarios, since, at MAC layer, 802.11 broadcast packets are transmitted at the minimum data rate (6 Mbit/s), whereas much higher rates are used for unicast transmissions (up to 54 Mbit/s). Note also that, even if three hops are involved in 4d2g-B, the first two hops occur through unicast transmissions (hence at much higher data rate than broadcast transmissions) and, thus, they mildly affect the throughput. Looking at Fig. 4.8, it can be seen that the loss probability is higher in the two scenarios with broadcast transmissions than in previous cases. This behavior is also expected: in case of failure, broadcast packets

are never retransmitted at the MAC layer, thus the reliability of the communication from the GO to its relay client is severely reduced.

In summary, the performance of the communication backbone is strongly affected by the traffic flow direction. The two different relay schemes, adopted within a group to work around the constraints imposed by Wi-Fi Direct, show significantly different performance. The main bottleneck is represented by broadcast communications from the GOs to their relay clients.

Comparison with Bluetooth

The Android devices we used in the previous experimentation are equipped with Bluetooth 3.0, which also supports multi-hop communications. For this reason, we have chosen to compare the performance of multi-group communications in Wi-Fi Direct and in Bluetooth under similar scenarios.

Under the same testbed setup shown in Fig. 4.4, we run a logical topology in Bluetooth, shown in Fig. 4.9, to mimic exactly the Wi-Fi Direct multi-group topology, considered in the previous sections. We set up two piconets, P1 and P2. P1 consists of three devices: one master (M1) and two associated slaves (S1A, S1B). P2 consists of two devices: one Master (M2) and one associated Slave (S2A). To enable bridging capabilities among the two piconets, M2 is also connected to M1 as a slave. We developed a single Android application (*not requiring to root the devices*) that can generate traffic, relay packets between the two piconets and record performance metrics. We run this application on each device, manually configuring the role of each device (source/destination/gateway).

We focused mainly on the maximum achievable throughput in different scenarios, when changing the source-destination pairs. We adopted 990 bytes as the application-layer packet size, to avoid packet fragmentation. The throughput is always measured at the receiver's application layer.

To faithfully mimic the scenarios considered in Section 3.3, we define each piconet as a "group" and consider the following cases:

1. "2 devices - 1 group" (2d1g), in which the source is S1A and the destination is M1. The packets are directly sent from S1A to M1 at the MAC layer without any relay.

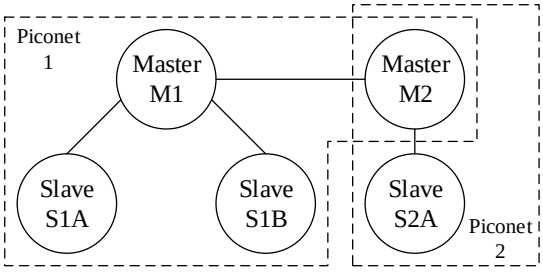


Fig. 4.9 Logical topology for the scenario based on Bluetooth communications, comprising two piconets.

2. "3 devices - 1 group" (3d1g), in which the source is S1A and the destination is S1B. The traffic generated by S1A is first sent to the M1, i.e. the piconet master. Each packet is processed by M1 at application layer and then relayed to S1B. The overall communication involves two hops at both application and MAC layer (S1A $\rightarrow$ M1 $\rightarrow$ S1B).

3. "4 devices - 2 groups" (4d2g), in which the source is S2A and the destination is S1B. The traffic traverses P1 and P2 thanks to the two application-layer relays operated by the two masters, M2 and M1. The overall communication involves 3 hops both at application and MAC layer (S2A $\rightarrow$ M2 $\rightarrow$ M1 $\rightarrow$ S1B).

Note that the Bluetooth does not support broadcast, hence we do not consider the last two scenarios in Section 3.3 involving broadcast communications.

We now compare the maximum throughput between Wi-Fi Direct and Bluetooth. Table 4.1 provides the maximum throughput achieved in each scenario, measured at application layer. The throughput is expressed in terms of absolute value and normalized value, as described below.

For Bluetooth, the maximum absolute throughput with 2 devices in direct communication (2d1g) is around 1.9 Mbit/s, which is consistent with the maximum net data rate of 2.1 Mbit/s. For 2-hop communications (3d1g), the throughput decreases by a factor of two (0.9 Mbit/s); this is expected, since the communication slots used by master M1 are divided in two: one to receive the data from one slave (S1A) and one to send the data to the other slave (S1B). In the case of two piconets (4d2g), the

Table 4.1 Experimental maximum throughput of Wi-Fi Direct and Bluetooth, measured at application layer

Technology	Throughput Mbit/s			Normalized throughput		
	2d1g 1 hop	3d1g 2 hops	4d2g 3 hops	2d1g 1 hop	3d1g 2 hops	4d2g 3 hops
Bluetooth	1.92	0.94	0.77	64%	31%	26%
Wi-Fi Direct	22.9	10.6	6.2	35%	16%	9.5%

maximum throughput is slightly less than 3d1g, since simultaneous transmissions (S1A→M1 and M2→S2A) can occur in the two different piconets.

Table 4.1 reports also the throughput achievable by Wi-Fi Direct, which is much higher with respect to Bluetooth in absolute terms, thanks to the higher data rates. For a fair comparison, we also report the normalized throughput obtained by dividing the throughput by the actual physical data rate adopted during the communication. In Wi-Fi Direct the rate must adapt to the channel conditions, and in this case we observed, most of the times, packets sent at the maximum data rate (65 Mbit/s) thanks to the small physical distance (always less than 40 cm) between the devices. For Bluetooth, the physical data rate adopted for the normalization is 3 Mbit/s, which is the data rate for Bluetooth 3.0 operating in the devices.

By considering only the normalized throughput in Table 4.1, Wi-Fi Direct achieves about 35%, 16% and 9.5% of the maximum throughput, respectively for increasing number of transmission hops, while Bluetooth achieves about 64%, 31% and 26% of it. The lower efficiency of Wi-Fi Direct is due to the contention-based protocol that regulates the access to the same radio channel. As already observed, the throughput decreases proportionally to the number of transmission hops. Instead, in Bluetooth the efficiency is larger thanks to the slotted time version of the protocol, whose access is coordinated by the Master.

Chapter 5

Formation of Multi-group Network and Logical Topology

5.1 Smart Group Formation Design

As confirmed by the above results, the Wi-Fi Direct network topology plays a crucial role in the performance achieved by inter-group data transfers. Thus, in this section we propose a fully-distributed group formation mechanism that (i) meets all the technology requirements, (ii) accounts for the devices physical resources and power consumption, and (iii) generates topologies that facilitate an effective inter-group communication.

Before introducing the mechanism, we highlight the main principles (P1-P4) that guide our design.

5.1.1 Design principles

(P1) When creating the topology, the number of groups should be minimized while keeping the interconnectivity. Indeed, as shown in Section 4.5, the throughput between two peers degrades with the number of hops the packets traverse. Properly choosing the GOs allows the control of the virtual topology on which data is routed and reduce the number of traversed hops. Note that involving more peers in a single communication also creates unnecessary power consumption of the relay peers along the path.

(P2) The choice of the GO should be optimized not only by taking into account its position within the topology, but also selecting the node that best meets critical requirements, such as high residual energy or unused computing/storage resources. As highlighted in the previous sections, GOs play an important role in each Group: forwarding packets at the MAC layer, keeping track of the content with associated peers, managing inner- and inter-group content exchange, etc. As a consequence, GO devices consume more power than Clients, due to higher computation and communication requirements. Moreover, once a GO leaves, all the nodes in the Group become disconnected. Therefore, properly assigning the GOs can make groups last longer and provide better performance. Besides, devices that can provide other resources (e.g., localization, Bluetooth availability, Internet connection) could be also preferred candidates to act as GOs.

Another crucial role in the network is played by Relay Clients, since they handle all the inter-group traffic and, hence, their selection must be optimized as well, with the same spirit of choosing GOs.

Finally, the multi-group network should involve as many as devices as possible. As introduced in Sec. 3.1, devices discover each other during the Device Discovery phase. Once started, Device Discovery remains active until a device initiates a P2P connection and establishes a P2P Group with others, or until the phase is stopped explicitly. After a device finishes discovering, the phase will never start again unless it is manually triggered by the user. As a consequence, some devices may not have the chance to find each other, leaving some devices disconnected. While setting up the network, we aim at maximizing the number of connected devices in order to maximize the amount of shared content in the whole network.

We therefore devise a procedure that accounts for the above requirements and lets each Wi-Fi Direct device coordinate with its neighbors so as to effectively decide its own role in the network topology (i.e., GO, Client or Relay Client), and, connect to other devices according to the selected role. For ease of presentation, we describe the procedure by referring to a set of devices that have not established any wireless link yet, and, following the Wi-Fi Direct specifications, they perform the Device Discovery procedure to acquire information on their neighborhood. We therefore start by introducing the information that each device needs to collect about its neighborhood (Sec. 5.1.2) and then we detail the steps of the network formation procedure (Sec. 5.1.3).

5.1.2 Neighborhood information

Unconnected network devices execute the Wi-Fi Direct Device Discovery procedure till they become part of a group. Through such procedure, devices can advertise their own presence and information about themselves to neighbor nodes, as well as receive other devices' advertisements. The advertised information consists of the following main pieces:

- *Suitability to be a GO.* The GO Ability Index (GOAI) expresses quantitatively the overall level of available resources at the advertising device. It corresponds to the weighted sum of the level of available resources (e.g., battery level, availability of Internet connection, CPU maximum frequency, amount of RAM and of non-volatile storage, etc.). Note that it also plays an important role in the selection of the Relay Client.
- *List of neighbors.* This is the list of neighbor nodes of the advertising node. The *neighbor* of a device is defined as a node that can be can seen with the Device Discovery procedure.
- *Current state in the topology formation.* As the topology formation progresses, this indicates the state of the procedure reached by the advertising device.

Note that, according to the above scheme, each device can acquire only a local view of the surrounding nodes, based on the list of its 1-hop neighbors with their GOAI and on the list of its 2-hop neighbors. Indeed, enabling each device to learn the entire network topology would require a large exchange of information, which may lead to network congestion and increased latency to form the group.

5.1.3 Topology formation

We devise a distributed algorithm running at each device to decide its role (GO/Client /Relay Client). The aims of the algorithm are manyfold. First, a device with higher GOAI should have higher probability to become a GO. Second, the GOs should form a connected backbone in order to construct a multi-group network. Third, as explained at the beginning of this section, the number of groups must be minimized.

We consider the undirected connectivity graph among the nodes, according to which each vertex corresponds to a device and an edge exists among two nodes if the corresponding devices are able to communicate directly.

The topology formation can be modelled as a problem of Minimum Connected Dominating Set (MCDS) on the connectivity graph. A *dominating set* (DS) is a set of nodes such that all the nodes outside this set can reach the nodes in the DS within one hop. Each node in the DS is called the *dominant node* and corresponds to a GO in our system. If all the nodes in the DS can be connected to each other through only nodes in DS (i.e. DS induces a connected subgraph in the connectivity graph), then the set is a Connected Dominant Set (CDS). The CDS represents the communication backbone among GOs, which enables multi-group communications. In order to minimize the number of groups, the number of nodes in the CDS must be minimized.

It is well known that the MCDS problem is NP hard. Several distributed, approximated algorithms [22–24] were proposed. In [22] a node selects its role based on the choices of other nodes, which requires costly synchronization and coordination among the nodes, differently from our proposed approach. In [23] nodes take decisions dynamically, while in our case the nodes only decide their roles at the beginning. Besides, the algorithm [23] requires the geographical information of the neighbors, which may not be feasible in our scenario.

Our approach is instead based on Dai and Wu's (DW) algorithm [24], described in the following. Each node is assigned with a unique ID, which is a value representing the node's priority to become dominant. The DW algorithm runs in two synchronous steps. Initially, during the *marking step*, each node marks itself as a dominant if it has two neighbors that are not directly connected. Clearly, this marking process generates a large number of dominant nodes. In the subsequent *self-pruning step*, the set of nodes in the DS is reduced. Each marked node applies a self-pruning rule, denoted as *Rule-k*, to *unmark* itself, where k is a positive integer. According to this rule, a marked node, say node u, unmarks itself if all of its neighbors can be covered[1] by k nodes that (i) have an higher ID than its one, (ii) are marked, and (iii) induce a connected subgraph. The last condition assures that such set of k dominant nodes is eligible to build the GOs backbone and requires the full knowledge of the

[1] A node set A is covered by node set B if and only if each node in A is the neighbor of at least one node in B.

(a) Marking step (b) Self-pruning step

Fig. 5.1 An example of distributed selection of dominant nodes according to DW algorithm. Marked nodes are colored gray

connectivity among nodes that are arbitrary far from u. At the end of the step, the marked nodes elect themselves dominant. Note that the decisions taken during the self-pruning step are based just on the state at the end of the marking step.

Fig. 5.1 gives an example of electing the dominant nodes using DW algorithm. As shown in Fig. 5.1a, during the marking step, node A and C mark themselves as the dominant nodes since their neighbors B and D are not directly connected. In the self-pruning step, as shown in Fig. 5.1b, when Rule-1 is applied, node A unmarks itself as it finds that there exists the marked node C with a higher ID which is neighbor of B and D. On the contrary, even if the topology is symmetric, C does not unmark itself since its ID is higher than A. At the end, only node C elects itself as the dominant node and all the other three nodes can reach it in one hop.

The DW algorithm is tailored to wireless ad-hoc networks where nodes are already connected and is used to address the routing problem, i.e. only dominant nodes are allowed to route messages. As a result, it does not consider the full mesh topology where all the nodes are neighbors of each other. In this case, applying the DW algorithm, no node can mark itself in the marking step since no node can find two neighbors that are not connected. In our scenario, however, we aim at selecting proper dominant devices and forming groups that lead to efficient data transfers across the network.

Our algorithm, denoted as *Smart-Group-Formation* (SGF), is fully distributed and runs at each node independently, following a procedure based on three subsequent phases, as shown in Fig. 5.2. Thus, some time coordination is required to make sure that all the nodes are running the same phase contemporary, and one possible implementation will be discussed in Section 5.2. In the following we explain each phase focusing on a given node M, and we will refer to its GOAI as GOAI(M).

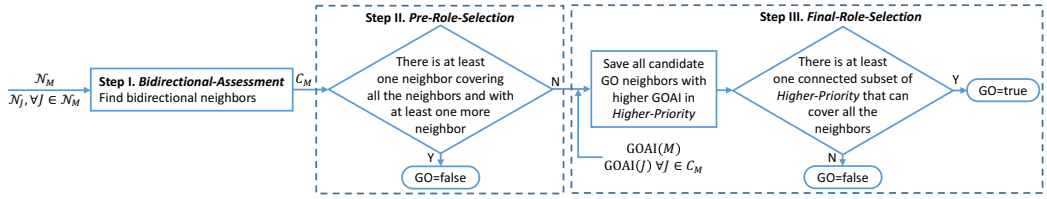


Fig. 5.2 The three steps of the role selection algorithm run by node M.

Algorithm 1 BIDIRECTIONAL-ASSESSMENT at node M
Input: $\mathcal{N}_M, \mathcal{N}_J$ for all $J \in \mathcal{N}_M$
Output: $\mathbf{C}_M$ ▷ List of M's bidirectional neighbors

1: **initialize** $\mathbf{C}_M = \{\}$
2: **for** every neighbor device $J \in \mathcal{N}_M$ **do**
3: **if** $M \in \mathcal{N}_J$ **then** ▷ if M's neighbor sees M
4: **add** J to $\mathbf{C}_M$ ▷ then, bidirectional connectivity
5: **end if**
6: **end for**

Step I: Bidirectional-Assessment. Due to the unreliability of the radio communication and the Device Discovery protocol, the visibility between any pair of nodes may not be bidirectional, and thus a node may not appear in the neighbor list of its neighbors. In this step, each nodes finds the neighbors for which bidirectional visibility is assured, namely, its *bidirectional neighbors*. Only such nodes are considered "valid" neighbors for the following steps.

Let $\mathcal{N}_M$ be the set of devices discovered by M, and let $\mathbf{C}_M \subseteq \mathcal{N}_M$ be the set of devices with which M has bidirectional visibility. Algorithm 1 reports the pseudocode for BIDIRECTIONAL-ASSESSEMENT of SGF, aimed at obtaining the bidirectional-visibility neighbor list $\mathbf{C}_M$. For the ease of description, we define the neighbors with bidirectional visibility of a device as its *bidirectional neighbors*. Specifically, for each neighbor, node M checks the bidirectional visibility by verifying if it appears in the neighbor list of each its neighbors.

Step II: Pre-Role-Selection. This step is described in Algorithm 2: each node makes a preliminary decision on whether to become a GO or not, by checking if there are two unconnected neighbors, similarly to the marking step in DW algorithm. To solve the issue that no device would be marked in a full mesh topology under the DW algorithm, we modify the DW marking step as shown in lines 2-4: a device M does not become a GO if there exists at least one neighbor that can cover all the

Algorithm 2 PRE-ROLE-ELECTION at node M
Input: $\mathbf{C}_M$
Input: $\mathcal{N}_M$, $\mathcal{N}_J$ for all $J \in \mathbf{C}_M$
Output: GO(M) ▷ Boolean flag

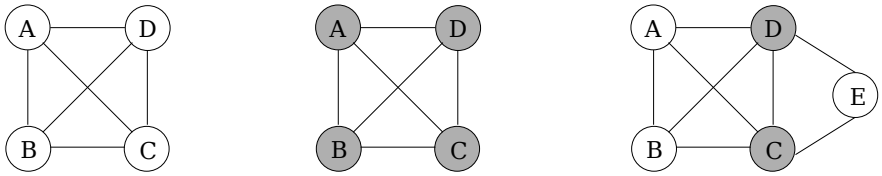
for every device J in $\mathbf{C}_M$ **do**
  if $(\mathcal{N}_M \cup \{M\}) \subset (\mathcal{N}_J \cup \{J\})$ **then**
    GO(M)=false ▷ M is not GO
    return
  end if
end for
GO(M)=true ▷ M is a candidate GO

Algorithm 3 FINAL-ROLE-ELECTION at a candidate GO node M
Input: $\mathcal{N}_M$, GOAI(M), GOAI(J) for all $J \in \mathbf{C}_M$
Output: GO(M) ▷ Boolean flag

// Find all candidate neighbors with higher GOAI than M
Initialize Higher-Priority={} ▷ Neighbor candidate GOs with higher GOAI
for every device J in $\mathbf{C}_M$ **do**
  if (GOAI(J) > GOAI(M)) **and** (GO(J)=true) **then**
    add J to Higher-Priority
  end if
end for
// Decide the role of M
for each connected subset $\Omega \subseteq$ Higher-Priority **do**
  if $\mathcal{N}_M \subseteq \mathcal{N}_\Omega$ **then**
    GO(M)=false ▷ M is not GO
    return
  end if
end for
GO(M)=true ▷ M becomes GO

neighbors of M and has at least one more neighbor than M; otherwise, the node becomes a candidate GO.

Fig. 5.3 illustrates the reasoning behind the different marking approach adopted by SGF compared to DW. Fig. 5.3a shows a full mesh topology in which no node can become a GO according to DW algorithm, since no one in the graph has unconnected neighbors. On the contrary, as shown in Fig. 5.3b, by applying SGF, as for each node no neighbors have larger coverage than it, all the nodes become candidate GOs. Fig. 5.3c shows a different topology in which, according to DW marking step, node



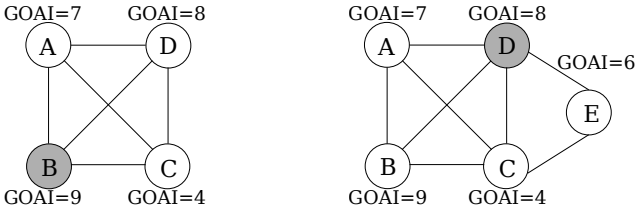
(a) DW marking. (b) PRE marking. (c) DW/PRE mark.

Fig. 5.3 Effect of different marking rules (DW and PRE-ROLE-ELECTION) for two different topologies. Marked nodes are colored gray and represent candidate GOs.

C and *D* would become candidate GOs since they both have a neighbor *E* that is not connected with their other neighbors. According to our approach, we get the same result. Specifically, for node *A* and *B*, their neighbors, *C* and *D*, have a distinct neighbor, node *E*, while covering all the neighbors of node *A* and node *B*. Thus node *A*, *B* cannot become the GOs and the role election phase is finished for them. For a similar reason, node *E* does not become a GO too. In the meanwhile, none of node *A*, *B* and *E* can fully cover the neighbors of node *C* and *D*. As a consequence, only node *C* and *D* are marked as candidate GOs.

Step III: Final-Role-Selection. It is executed only by those nodes that marked themselves as candidate GOs at the end of the previous step (i.e. GO(*M*)=true). As discussed in Sec. 5.1.2, thanks to the advertised GOAI, each candidate node can build a set, named Higher-Priority, of bidirectional neighbors that are candidate GOs and have larger GOAI than itself (see Algorithm 3, ln. 3-7). To select the role of *M*, we consider a restricted version of Rule-*k*: node *M* unmarks itself if there exists a subset of 1-hop neighbors that (i) appear in the Higher-Priority set and (ii) are directly fully connected among them. Thus, in lines 9-14 of Algorithm 3, if at least one subset in the Higher-Priority set meets requirement (ii), then *M* unmarks itself and does not become GO. Note that in the pseudocode, given a generic subset Ω of Higher-Priority, $\mathcal{N}_\Omega$ denotes the union of the set of neighbors discovered by each node in Ω: $\mathcal{N}_\Omega = \cup_{X\in\Omega}\mathcal{N}_X$.

Fig. 5.4 shows the results of running FINAL-ROLE-ELECTION based on their results of PRE-ROLE-ELECTION in the topologies shown in Fig. 5.3b and Fig. 5.3c respectively. In the full mesh topology, as shown in Fig. 5.3b, all the nodes become candidate GO after finishing Algorithm 2. When running Algorithm 3, nodes *A*, *C* and *D* only add node *B* to their Higher-Priority sets, since *B* is the neighbor candidate GO with the highest GOAI. When checking all the possible subsets of Higher-Priority lists, which is actually just $\{B\}$, all the three nodes find that this subset can cover



(a) Full connected graph (b) Irregular graph

Fig. 5.4 Results of running FINAL-ROLE-ELECTION on the candidate GOs obtained in PRE-ROLE-ELECTION for the cases shown in Fig. 5.3b and Fig. 5.3c respectively. Gray nodes are the final GOs.

their neighbors. As a result, all these three nodes renounce to become GO. Instead, node *B* finds that none have a higher GOAI than itself, and thus its Higher-Priority set is empty; as a result, *B* becomes GO. In the topology shown in Fig. 5.3c, node *C* and *D* are candidate GOs at the end of Algorithm 2 and they are the only ones to run Algorithm 3. Node *D* has an empty Higher-Priority set, as the only neighbor candidate GO *C* has a lower GOAI; thus *D* becomes GO. On the contrary, node *C* adds node *D* to its Higher-Priority; since *D* can cover all its neighbors, node *C* renounces to become a GO.

5.2 Implementation of Smart Group Formation

We implemented the SGF mechanism into five sequential phases: (i) *Neighborhood Information Collection (NIC)*, when devices gather information about other devices, (ii) *Neighborhood Advertisement (NA)*, when each device advertises its 1-hop neighbors, (iii) *Role Selection (RS)*, when devices decide their roles according to the distributed approach described in Sec. 5.1.3, (iv) *Connection (CO)*, when devices setup the fully connected, multi-group network, and, after a group becomes consolidated, (v) *Relay Client Selection*, the GO selects one Client as Relay Client. These implementation phases are summarized in Fig. 5.5. Notably, during the first 4 phase the device are not connected at layer 2 and interact only exploiting the advertised device name. Only after phase (iv), the devices are connected at layer 2.

Recall that the Device Discovery procedure in Wi-Fi Direct allows a device to discover other nodes by acquiring their MAC address and device name, which is a human readable string of ASCII characters. Also, a device can be discovered

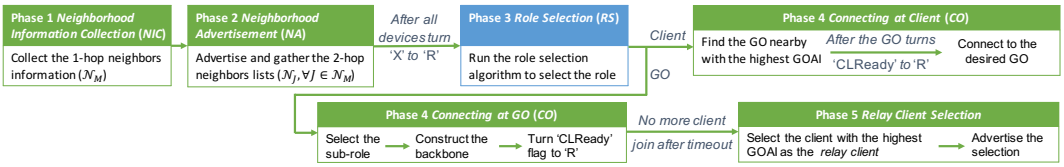


Fig. 5.5 The implementation phases of SGF for a device M to construct the multi-group communication network.

by others while performing the discovery procedure. In our SGF implementation, we encode all information that devices need to advertise at the various stages of the topology formation process, in the device name field of the messages that are transmitted during the Device Discovery procedure.

5.2.1 Neighborhood Information Collection (NIC) phase

Upon starting, each device computes the GOAI and encodes it into human-readable ASCII characters in the interval $[32, 127]$, which appears explicitly in the device name advertised by the node, as `SGF_ID-GOAI`. In more details, the `SGF` string identifies all the nodes running our SGF approach, and the following ASCII characters identify the device ID, chosen as the last 2 bytes of the MAC address of the interface, which are represented by 4 ASCII characters in hexadecimal expression, in order to minimize ID collisions. During the initial NIC, with Device Discovery, each device discovers its neighbor nodes in terms of MAC address and device name. Notably, we avoided to use the GO Intent field – an integer in the interval $[0, 15]$ included in the GO Negotiation Request [25] to set up a connection – for two reasons: (i) without rooting the device, it is not possible to modify the GO Intent used to compare with the requester at the receiver, and (ii) because of the limited number of bits (just 4) to encode the GOAI.

Due to the asynchronous nature of the Device Discovery process, the vision of the neighborhood of each node becomes coherent with that of all other nodes only after some transient time. Thus, it is important to set properly the maximum time allowed for this initial phase in order to find the best compromise between the consistency of the devices' view and the duration of the network topology formation. According to [26], during P2P Discovery, most of the devices tend to be found firstly within the initial 2 seconds and secondly around 15 seconds; these results provide some guidelines to design the timing for the discovery. We propose the

Table 5.1 Minimum number of found devices required in each time interval to end the P2P Discovery phase, with an example of a neighborhood of 4 devices. We assume $N_1 \geq N_2 \geq N_3 \geq N_4$

Time Interval $[s]$	Minimum number of devices	Example values
$[0,2]$	N_1	3
$[0,5]$	N_2	2
$[0,10]$	N_3	1
$[0,15]$	N_4	1

threshold-based timing shown in Table 5.1, according to which it is necessary to see at least a minimum number of neighbor devices in a given interval to conclude the NIC phase. For growing time intervals, we decrease the minimum number of neighbor devices. If a device has found no devices within 15 seconds, it restarts P2P Discovery and resets the timer. The list of discovered neighbors is stored in the local *neighbor list*.

5.2.2 Neighborhood Advertisement (NA) phase

At the end of the NIC phase, each device starts to advertise the list of its neighbors in the device name, which will be acquired by other devices, according to the following string:

`SGF_ID-GOAI-ID1#ID2# ··· #IDk-X`

which is an extension of the name adopted during the NIC phase. After the GOAI string, coded as in the NIC phase, a device includes the list of all the neighbor IDs separated by '#'. The final character X is the final string delimiter and indicates whether the advertising device is ready ('R') or not ('N') for the following phase. This field is initially set to 'N'. Each device monitors its neighbors by checking the neighbor list and the `X` state in their device names. Once a device receives the advertisements from all the discovered neighbors (i.e., those that appear in the list it advertises), it will set `X` equal to 'R'. As soon as all the neighbors of a device have turned their `X` flag to 'R', the device will enter the Role Election phase. Notably, this procedure does not require bidirectional visibility since the final 'R' decision is taken at a node independently from the fact that the neighbor nodes are able to see it.

5.2.3 Role Selection (RS) phase

During this phase, a device follows the topology formation approach described in Sec. 5.1.3 in order to set its role. At the outset, each device runs BIDIRECTIONAL-ASSESSMENT to compute the list of neighbor nodes with bidirectional visibility, using the neighbor information advertised by others. Then PRE-ROLE-SELECTION makes a preliminary decision on becoming candidate GO. Whenever the decision is taken, the result is reported in the advertised device name, according to the following format: `SGF_ID-GOAI-Y`, where `Y` can be either the string `Client` when the device declines to become a GO or `Marked` whenever the device becomes a candidate GO. When all neighbors have advertised their decision, the node enters the FINAL-ROLE-SELECTION. At the end, each device advertises its final decision in the device name with the following format: `SGF_ID-GOAI-Role`, in which the `Role` can be either `Client` or `GO`. After setting its role, each Client waits for all of its neighbors to finish the SGF procedures by checking their device names, and then it enters the next phase directly. The GOs, after all its neighbors have finished their role selection procedures, add a numerical `nGOs` value in the device name field, as follows: `SGF_ID-GOAI-Role-nGOs`. This field is used to advertise the number of neighbor GOs of each GO; a GO enters the next phase only after it has advertised its `nGOs` value.

5.2.4 Connection (CO)

In this phase, devices setup connections among each other so as to establish the final multi-group network. The main idea is to exploit the number of neighbor GOs as a priority to determine how to build the network backbone that will be used to route the traffic between the groups. The behavior of a device in this phase depends on the role selected in the previous one. We therefore describe the behavior of the devices depending on whether they are GO or Clients.

Connection at the GO

Consider a generic node x selected as GO. If x is associated at layer-2 to another GO, x is tagged with a sub-role denoted as *Legacy Client*. If other GOs have a layer-2 association to x, then x is tagged with a sub-role denoted as *Root*. Depending on

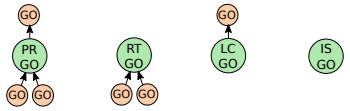


Fig. 5.6 All the possible cases for the GO sub-roles when/whether connected to other GOs.

how/whether a GO is connected to other GOs, *x* will take on one of the four sub-roles depicted in Fig. 5.6: a *LC-GO* acts just as Legacy Client; an *RT-GO* acts only as a Root; a *PR-GO* acts as a parent GO, i.e., *x* is as a Legacy Client of another GO and there is a GO that is connected to *x* as a Legacy Client; an *IS-GO* is an isolated GO, i.e., not connected to any other GO.

A generic GO *x* decides its sub-role as follows. As a first step, *x* looks for neighbor GOs. If there is none, *x* becomes an IS-GO. Otherwise, it checks the `nGOs` values advertised by all the neighbor GOs and compares them against its own. GO *x* becomes an RT-GO if it has the highest `nGOs`. Else, *x* tags itself as an LC-GO and connects to the neighbor GO with the maximum `nGOs` (ties are solved based on the GOAI and, if needed, on the MAC address). Such node is referred to as *target* of *x*; next, *x* adds a subfield, called `RT_Target`, to its device name and sets it to MAC address of its target node so that it can be advertised to its neighbors. The device name advertised through the Device Discovery procedure now has the following format: `SGF_ID-GOAI-Role-RT_Target/Accepting-nGOs -CLReady`. Then *x* searches among all its neighbor GOs for a device whose name contains *x*'s MAC address in the `RT_Target` field. If such a GO exists, *x* becomes a PR-GO.

After having selected their sub-roles, RT-GOs, LC-GOs and PR-GOs cooperate to construct the routing backbone across different groups. Specifically, each RT-GO firstly sets up a group, defines a *service* encoding the credentials of its own group, and disseminates this information through the standard P2P Service Discovery. Recall that, as described in Sec. 3.1, a device can acquire the information of a service even if it is not connected to the service provider. After broadcasting its credentials, each RT-GO sets the `Accepting` field in the device name to 'R' (i.e., Ready) to notify the availability to accept incoming connection requests. Only after such notification, the LC-GOs and PR-GOs start connecting to their GO targets. Once established a legacy connection with its target GO, a PR-GO follows the same steps taken by an RT-GO: it creates a group, broadcasts the group credentials and sets its `Accepting` field to 'R'.

In order to let the client devices join the group, a GO turns the `CLReady` field to 'R'. IS-GOs, being not involved in the backbone establishment, set their `CLReady` field to 'R' as soon as the group has been established. Instead, an RT-GO or a PR-GO does so only when all LC-GOs and PR-GOs, for which it is a target, have connected.

Connection at the Clients

We now describe the behavior of a client node after the Role Election phase. Each client ranks the neighbors based on the GOAI and tries to connect towards the neighbor GO with the highest GOAI. Nevertheless, a client has to wait for the target to turn the `CLReady` field to be 'R'. To find a reasoable tradeoff between choosing the best GO and the time required to join a group, after a time-out the client restarts the association procedure with the GO ranked just after the previous target GO.

As the clients runs asynchronized, concurrent connection requests may flood a GO. Note that the connection between two devices takes time to setup. During the establishment of a connection, other connection requests fail due to Wi-Fi Direct protocol. This causes many of the clients to be isolated. To reduce the probability that two connection requests collide, we have implemented an adaptive back-off mechanism to disperse the connection requests.

5.2.5 Relay Client selection

In the Android Wi-Fi Direct implementation, a GO is notified when a new client joins the group. The GO, after no more client join the group, selects the relay client among the associated clients. Note that waiting for the group to become stable gives the GO a comprehensive knowledge of the clients, which can be selected more carefully. The GO chooses the client with the highest GOAI as Relay client. To notify its decision, the GO transmits a *relay client appointment* message at application level containing the MAC address of the selected relay client; the message is sent in broadcast to the entire group through a UDP packet. Note that we exploit a broadcast UDP packet because in Android the GO does not hold the IP layer information of the clients but just their device names and their MAC addresses. Thus the GO cannot send a unicast UDP packet to the selected client.

When receiving the relay client appointment message, only the client matching the MAC address acknowledges the GO and starts to act as relay client. The GO keeps sending the appointment message until it is acknowledged. In case of too many of unacknowledged appointments, the GO selects another relay client based on the GOAI ranking.

5.3 Performance of Smart Group Formation

We study the performance of the smart group formation through both numerical simulations and experiments on a real testbed.

5.3.1 Numerical evaluation

We developed a MATLAB simulator to generate random geometric graphs and to implement the PRE-ROLE-ELECTION and FINAL-ROLE-ELECTION phases described in Section 5.1.3. Each vertex in the graph represents a portable device and an edge exists between two vertices if and only if both of them reside in the transmission ranges of each other. To simulate the GOAI, each vertex is also assigned with a random integer in $[32, 127]$. The vertices are distributed in a two-dimensional space where the x and y coordinates are chosen uniformly at random in the range $[-40, 40]$ m, thus the maximum distance is around 113 m. We vary the total number of nodes and the transmission range, assumed fixed among all the nodes. This is coherent with a scenario in which all the devices use the same technology, i.e., Wi-Fi.

We focus on the *GO-ratio*, defined as the ratio between the final number of GOs generated with the role election algorithm and the total number of nodes in the network. We vary the scenario by changing the number of nodes and the transmission range. We discarded all the sample graphs with node partitions. We run 30 simulations for each scenario, each run with a different graph and assigned GOAIs.

Fig. 5.7 shows the GO-ratio with respect to the transmission range for 8 and 16 nodes. When increasing the transmission range, the graph becomes more connected and thus less GO are required to build a connected backbone; this explains the decreasing behavior of both curves. When the transmission range is enough large,

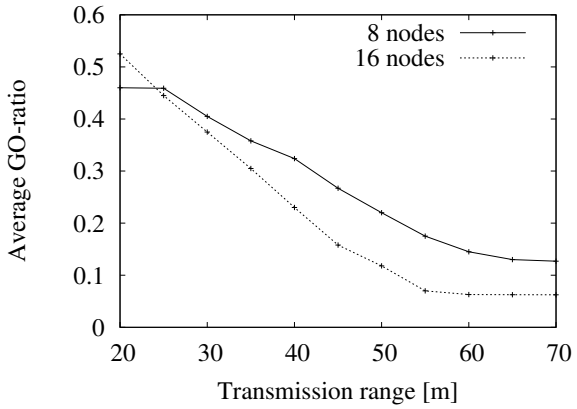


Fig. 5.7 GO-ratio on random geometric graphs and random values of GOAI.

the graph become full connected and thus just one GO is required to cover all nodes, and the GO-ratio become $1/8 = 0.125$ and $1/16 = 0.0625$, respectively. When the transmission range is too small, instead, the graph is sparse and the degree is low and thus the choice of GOs is very limited. This explains why we observed a constant GO-ratio for 8 node and transmission range ≤ 25 m.

5.3.2 Experimental evaluation

We adopted the same testbed described in Section 4.5.1 to test the actual implementation proposed in Section 5.2. Specifically, we setup different topologies composed of four devices and for each topology we tested the time required by each of the four phases (i.e. NIC, NA, PE, CO) that consist the SGF procedure.

We addressed many experimental issues. Firstly, note that it is difficult to setup a dedicated topology where a device can only hear certain neighbors, which requires to precisely maintain the relative positions of the devices. Therefore, we forced the devices to neglect certain neighbors according to the desired topology. Secondly, we did not force the devices to start the formation procedure at exactly the same time. Recall that devices are synchronized by the `X` flag when transiting into the RE phase at the end of the NIC and NA phase. Thus the formation starting time only have an impact on RE and NA phases. Thirdly, note that during the CO phase, the GOs and Clients have distinct behaviors. For a GO, we considered the phase finishing time when the flag `CLReady` is turned to 'R', i.e., when the local backbone has been

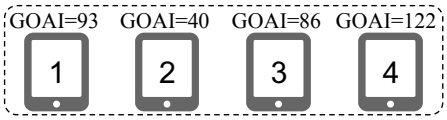


Fig. 5.8 A full-connected topology with one area of proximity.

Table 5.2 Experimental results for the full-connected topology of Fig. 5.8.

Phase	Duration $[s]$			
	Device 1	Device 2	Device 3	Device 4
NIC	2.0	2.0	2.0	2.0
NA	18.1	17.9	17.9	18.0
RE	6.8	5.8	5.8	6.1
CO	12.1	11.3	13.5	0.0
Total	39.0	37.0	39.2	26.1

established. For a Client, we consider the finishing time when Client's target GO sets the `CLReady` to 'R'.

We mainly focused on three different topologies and for each topology, we carried out 30 independent experiments. The time required for each phase was measured at application level. All the average results were obtained with an accuracy $\leq 1\%$, evaluated as the relative width of the 95% confidence interval.

We firstly consider a full-connected topology shown in Fig. 5.8 where all the four devices can hear each other. Running the role election algorithm, device 4 has the largest GOAI and becomes a GO and all the other three devices choose to be the Clients and connect to the GO, i.e., device 4. Table 5.2 shows the corresponding results of the time consumed at each phase and the total time. Recall that a device concludes the NIC phase and enters the NA phase as soon as it discovers enough neighbors within given intervals. During this test, all the devices have found the others in 2 seconds. Since all the devices start the formation almost at the same time, the duration of the NA phase of the four devices are similar. Notably, NA is the longest phase, due to the temporal thresholds adopted in the process. Recall that a device finishes the RE phase only when all its neighbors have selected their roles, which also causes the synchronization among the devices at the end of this phase. Thus the devices except Device 2 experience almost the same duration of RE phase. Since Device 4 is the only GO in this phase, it becomes a normal GO with an immediate decision and it is the first to end the SGF procedure, after 26 s. The duration of the final CO phase is not negligible due to time needed by the adopted

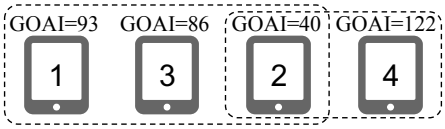


Fig. 5.9 Asymmetric topology with two areas of proximities.

Table 5.3 Experimental results for the asymmetric topology of Fig. 5.9.

Phase	Duration $[s]$			
	Device 1	Device 2	Device 3	Device 4
NIC	2.0	2.0	2.0	2.0
NA	18.8	18.3	18.4	18.0
RE	4.6	6.8	4.5	4.1
CO	16.5	0.0	16.5	17.5
Total	41.9	27.1	41.4	41.6

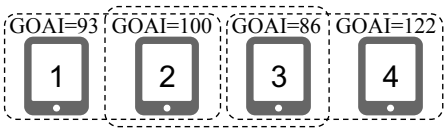


Fig. 5.10 A linear topology with three areas of proximities.

Table 5.4 Experimental results for the linear topology of Fig. 5.10.

Phase	Duration $[s]$			
	Device 1	Device 2	Device 3	Device 4
NIC	2.0	2.0	2.0	2.0
NA	16.8	20.4	17.7	17.5
RE	3.3	14.0	11.3	4.9
CO	50.4	19.9	27.7	74.8
Total	72.5	56.3	58.7	99.2

protocol and required for the device name update and advertisement in the Android implementation of Wi-Fi Direct. Thus, all the other Clients end after 39 s.

The second topology is asymmetric and shown in Fig. 5.9, where devices 1, 2 and 3 can hear each other, whereas device 4 is in radio proximity of device 2 only. As expected, according to our SGF, during the RS phase device 2 becomes IS-GO because of its centrality, despite its smaller GOAI, and all the other devices become Clients connected to it. Table 5.3 presents similar results to the full-connected scenario in Table 5.2, since all the Clients take around 42 s and the IS-GO around 27 s. As in the previous case, the Clients takes longer due to the CO phase.

The last topology is linear and shown in Fig. 5.10. Each device sees just a subset of neighbor nodes. The experimental results are reported in Table 5.4. Now both devices 2 and 3 become GOs since they are directly connected and have different neighborhoods. At this point, the two GOs must coordinate to create the communication backbone, hence the duration of the CO phase is no longer negligible. Since node 2 has a higher GOAI than node 3, device 2 includes the credentials of its group in the message used to advertise its service to the nodes in its proximity. When device 3 receives this information, it uses such credentials to connect to device 2 (which has a higher GOAI than itself) as a Legacy Client. As final result, device 2 is tagged as RT-GO and device 3 as LC-GO. Device 1 and 4 become Clients and wait until the backbone is established. Note that this scenario is representative of a large wireless network, in which the nodes are very far from each other and our proposed multi-group scheme is the only viable approach to let the devices form a fully connected network.

Part II

Transparent Bandwidth Aggregation for Residential Access Networks

Chapter 6

Cooperative Bandwidth Aggregation

6.1 Background

The growing popularity of high-speed Wi-Fi technologies (as IEEE 802.11n) deployed in residential networks has exacerbated the inequality between the bandwidth available in local domestic networks and the bandwidth available to access the Internet Service Provider (ISP) network. Notwithstanding EU plans for fast broadband foresee 100% coverage of 20 Mbps (or more) access connections for EU citizens by 2020, the majority of member states are still on the way to support basic broadband access connections, based on ADSL technology. Basic ADSL provides a download bandwidth (around 1-10 Mbps) and an upload bandwidth much smaller (around 0.1-1 Mbps), typically lower than local area networks based on Ethernet and Wi-Fi. Thus, ADSL link constitutes often the performance bottleneck for residential users accessing Internet-based applications in which upload traffic is dominant (as cloud storage and computing). Consequently, several methods have been recently proposed to increase the performance perceived by the domestic users through aggregating the available bandwidth of neighboring Wi-Fi Access Points (APs).

This approach is practically enabled by the high density of APs in residential areas, which provides overlapping radio coverage. A recent study [27] showed that, in a metropolitan area, a generic AP can see up to 52 neighbors, with a median value around 7. Furthermore, despite the strong correlation of the residential daily Internet traffic at the aggregated level, which follows nicely a day-night sinusoidal traffic shape, the traffic correlation among neighboring users is typically small due

to different habits and behaviors of each individual person. Thus the instantaneous traffic of neighboring ADSL connections is often uncorrelated, enabling statistical multiplexing of the traffic among neighboring users. State-of-art solutions for access bandwidth aggregation require modifications at client side, such as custom drivers or specific applications to be installed on users' devices. This makes their deployment not commercially viable due to chipsets variety, to operating systems diversity, and to additional constraints imposed by smartphones and tablets development environments.

In recent years, many works have been focused on the bandwidth aggregation problem in the context of the access network. FatVAP [28] is one of the first works on aggregating the access broadband bandwidth of multiple APs. FatVAP has two main contributions: an 802.11 driver that enables a client to connect to multiple APs using a single radio wireless chip; a scheduler that enables a client to decide to which APs to connect to maximize the throughput. Unfortunately, the proposed approach is not suitable for domestic users of ISPs, as targeted by our work, since it is not transparent for the user, who is required to install a new driver that is not integrated in an off-the-shelf operating system and maybe not supported by the available chipset of the wireless card. In addition to that, [28] does not involve an authentication procedure when connecting to an AP and thus cannot be adopted in a residential access network. The work in [29] extends [28] and addresses specifically the security issue when connecting to multiple APs. It proposes a fast authentication mechanism, but it is not compatible with legacy 802.11 security protocols. Furthermore, clients connect directly to each other using a virtual interface connected in ad-hoc mode, limiting severely the portability of the approach. Finally, whenever a client shuts down, it cannot share anymore the backhaul bandwidth to others, even if its access gateway is still working. In [30] a new scheduling scheme is proposed that guarantees a fair access among multiple clients, which overcomes the drawback of FatVAP that only maximizes the performance of a single client, neglecting the potential unfairness among all the clients. Finally, [31] proposes to aggregate the access bandwidth by exploiting the transmission on overlapping channels, but it relies completely on a non-standard communication technology. In [32] a new opportunistic approach is proposed to aggregate the backhaul bandwidth. The main idea is that the client sends a packet in broadcast towards all the APs and each AP runs a scheduler that decides whether to forward this packet. As a result, it requires a strong cooperation among the clients, APs, and the servers, only achievable with customized devices. Furthermore,

broadcast communications in 802.11 are inefficient in terms of bandwidth, since occurring always at the minimum data rate.

One of the most relevant work to ours is SmartAP [33]. Evolving from [30], SmartAP develops an AP-based scheduling algorithm that tries to maximize the overall throughput of all the clients. SmartAP is transparent for the user, since it does not require modification on the applications, the wireless card driver, or the operating system. An optimization algorithm coordinates the traffic flows among neighboring APs in order to maximize the bandwidth. To achieve some transparent behavior for the local user of an AP, each AP must estimate its own available bandwidth and communicate to a central controller running the optimization algorithm. Due to the latencies for the bandwidth estimation and the centralized control, the approach slowly reacts to fast varying loads. Unlike SmartAP, our approach is completely distributed, since each Access Gateway runs a traffic control scheme independently from the others; in addition, our scheme preserves the bandwidth available for the local traffic. Another relevant work is 3GOL [34], which shares the same motivation of our work, i.e. boosting the speed of ADSL users. Instead of aggregating the bandwidth of multiple APs, [34] proposes to exploit a parallel cellular connection to increase the speed of a home user. The proposed approach is not transparent for the user, which must install a dedicated scheduler in her access devices.

In this thesis we propose Beyond One's Bandwidth (BOB), a distributed gateway-centric system exploiting the collaboration between multiple Access Gateways (AGs) to provide a higher Internet connection speed to residential users without any software or hardware modification at the client side. The AG is a standard access device that integrates a broadband modem, a network-layer router, and a Wi-Fi AP. Residential devices such as laptops, smart phones and TVs can access the Internet by associating to the AP or by connecting through Ethernet. We design BOB to meet the transparency requirements of a real commercial deployment. Each AG in BOB is responsible of constructing a dedicated wireless topology with other nearby AGs, forwarding portion of traffic to neighbors while guaranteeing that each user is able to fully exploit his own broadband connection bandwidth, without observing any meaningful performance degradation due to the cooperative sharing scheme.

We provide the following contributions. We propose BOB architecture, based on integrating a novel flow-balancing scheme with a traffic scheduler, both amenable for implementation in Linux networking stack. We develop a fluid model to estimate

the achievable performance in terms of throughput and fairness based on the traffic distribution policy adopted in BOB. This model allows to compute the optimal parameters of the architecture to maximize the performance, given the knowledge of the user traffic rate. Finally, we implement and test BOB in an operational scenario with several typical clients. The results show that BOB can provide a substantial increase of the throughput with negligible overhead under synthetic traffic and real applications. Note that a preliminary version of our work was published in [35].

This part of the thesis is organized as follows. In Section 6.2 we describe the proposed architecture, whose detailed implementation is discussed in Section 6.3. We also integrate Multi-Path TCP (MPTCP) into our BOB system in Section 6.4. In Chapter 7 we present the fluid model. Specifically, in Section 7.1 we provide the formulation of different scenarios with the model. This model is evaluated numerically and validated experimentally in Section 7.2.

6.2 Architecture

We consider a residential access network of an ISP, in which each household is equipped with an AG. The AG is connected through an ADSL line to the ISP's POP and provides connectivity to users' devices through an 802.11 interface. Fig. 6.1 shows a basic example of 3 AGs that cooperate to implement our proposed BOB bandwidth aggregation scheme. The wireless interface of each AG is connected to the *local* devices but also to the *neighboring* AGs. In the example, AG2 is connected to both AG1 and AG3. BOB is designed to exploit the unused ADSL bandwidth of the neighboring AGs to boost the performance of the local devices. Fig. 6.2 shows an example in which the local devices of AG1 and AG3 are currently exploiting 60% and 30% of their own ADSL upload bandwidth, respectively. Thanks to the cooperation of the two neighboring AGs, the local devices of AG2 exploit not only their local ADSL bandwidth, but also the unused ADSL bandwidth of AG2 and AG3 to/from the POP. Assuming all the ADSL link rates being the same, a $2.1\times$ gain would be experienced on the overall uplink bandwidth.

We designed the cooperative bandwidth aggregation system by considering the following constraints. *(1) Transparent performance*: the users' QoS should not be affected negatively by the sharing scheme. This means that the local user should be able to fully exploit his local ADSL bandwidth, independently from the neighbors'

behavior. If some performance degradation is experienced, it should be negligible. *(2) Transparent deployment*: the system should not require any modification of the applications and of the Wi-Fi interface drivers at the users' devices, and should be compatible with the most common existing Internet transport protocols and with standard IP routing. *(3) Single wireless interface*: to reduce costs, the AG should exploit a single Wi-Fi physical interface to connect both the local users and the neighboring AGs. *(4) Self-configuring scheme*: the cooperation scheme must be enabled in a distributed way without the need of a central control. All the previous constraints are dictated by an ISP who wishes to scale the approach to a large population of users, by providing a proprietary low-cost AG to its subscribers.

To meet all the previous design constraints, we combined many techniques. First, to achieve transparent performance, a traffic scheduler running in the AG regulates the traffic flows contending for the ADSL bandwidth. This guarantees that most of the ADSL bandwidth is devoted to the local devices, whereas a small bandwidth (negligible for the local user) is given to the neighboring AGs to avoid the starvation of active TCP flows. Section 6.2.3 will be devoted to describe the details of such traffic scheduler. Second, to achieve transparent deployment, the AG is entirely responsible to route the traffic flows. An internal flow-based load balancing scheme, described in Section 6.2.2, route part of the incoming TCP/UDP flows to the neighboring AGs, seamlessly for the local devices and for the whole ISP network. Thus, neither hardware or software modification is required in the users' devices and the bandwidth sharing scheme can work with all the different 802.11 standards currently available. We wish to emphasize that such *transparent deployment* is

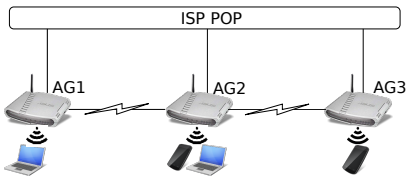


Fig. 6.1 Example of a scenario with three cooperating AGs running BOB

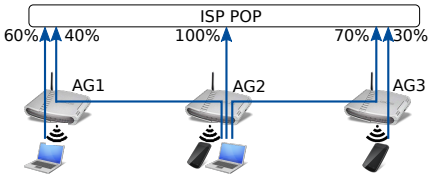


Fig. 6.2 Example of a bandwidth sharing scenario achieved by BOB

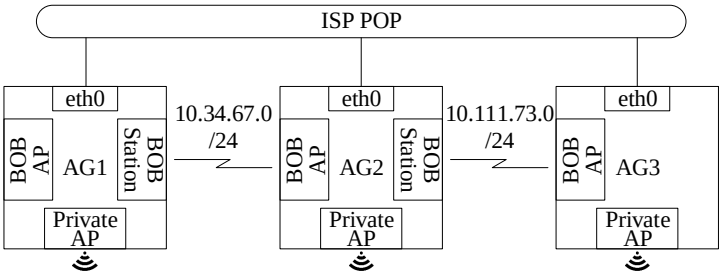


Fig. 6.3 Layer-2 topology highlighting the role of each kind of virtual interface.

the main difference of our approach with respect to the state of art [28–30]. The constraint about the single wireless interface poses some natural limitation regarding the scalability of the approach, since a common channel must be shared across all the cooperating AGs. Finally, to allow the cooperation among the AGs, we build a logical interconnection topology among the neighboring AGs, as detailed in the following Section 6.2.1.

6.2.1 Communication topology

The topology connecting the AGs and their local devices requires multiple connections at MAC layer for each AG. Since the network interface is single, this configuration can be achieved by defining many virtual interfaces on the Wi-Fi physical interface. One virtual interface, referred as "*private AP*", is devoted to the local user (more precisely, to all the domestic Wi-Fi devices) and acts as a standard 802.11 AP establishing a BSS for the local devices. Another virtual interface, denoted as "*public BOB AP*", provides the connectivity to the neighboring AGs acting as access point. Finally, one or more virtual interfaces, referred as "*BOB station*", allows the AG to connect to the public BOB AP interfaces of other cooperating AGs. Each virtual interface is configured with a MAC address chosen automatically in increasing order with respect to the original MAC of the physical interface, to avoid conflicts at MAC layer. Fig. 6.3 shows an example of a topology achievable by the three virtual interfaces present in each AG.

The formation of the layer-two topology is achieved by a simple distributed algorithm, as follows. Each AG periodically scans for public BOB AP interfaces. Whenever it finds a new one, if not yet connected to it, it associates to it and

establishes the bidirectional cooperation among the two AGs. To enable such process, each public BOB AP is identified by a BSSID composed by combining the unique identifier of the AG (obtained by the native public AP BSSID) and the string "BOB". This allows to understand whether one AG is already associated to another AG and to avoid double associations (as when a single AG acts as both public AP and station towards another AG). After establishing the layer-two connectivity among the AGs, the public AP runs a DHCP server to provide the IP address to the BOB-station interfaces. To avoid conflicts of IP network addresses, the public AP is set in the range 10.X.Y.0/24, where X.Y are chosen according to a 16-bit hash function applied to the string of native AG BSSID. In addition to the above association scheme, the cooperation between two AGs is actually established only if the RSSI is above a given threshold. This guarantees that only neighboring AGs cooperate when they have a reasonable good connectivity.

6.2.2 Flow-level balancing

All the incoming traffic on the private AP interface is processed by a flow balancer, to eventually distribute the traffic across different neighboring AGs. The balancer works at transport layer and identifies the traffic based on the standard pair of transport ports and network addresses. When the first packet of a new flow reaches the balancer, a load balancing algorithm selects the local destination, that is either the local interface of the ADSL connection, or the BOB interfaces (BOB-station or BOB-AP). Note that, since all the traffic is routed through a NAT to reach Internet, the first packet of a flow is always generated by a local device. Furthermore, given that an unique public IP address is associated to each AG, all the following packets of the same flow need to be forwarded along the same path of the initial packet. Note that this choice also avoids out-of-sequence packets within a flow, which are very poorly tolerated by TCP/UDP protocols, and it is compatible with standard IP routing also for the backward path.

We propose two possible flow-balancing schemes. The first scheme is based on a classical *Weighted Round Robin* (WRR) in which the number of flows sent on each BOB-interface and to the ADSL connection is proportional to the maximum bandwidth available in each interface. Such value can be estimated approximatively by the association data rate of each wireless interface and by the sync rate of the ADSL link. The AG must maintain an Interface Rate Table (IRT) with the

updated data rates of each interface. The second scheme is denoted as *Pending Flow Balancing* (PFB) and is designed to distribute each new flow to the specific local interface (either BOB or ADSL) with the minimum number of pending active flows. A TCP flow is defined as "pending active" during the period between the initial SYN handshake and the final FIN handshake. An UDP flow is similarly defined after the initial packet and until a timeout expiration after the last observed packet. The AG must maintain a Pending Flow Table (PFT) that keeps track of the numbers of active flows on each interface.

If we assume equal flow sizes, WRR tends to balance the load across the BOB interfaces obliviously of the actual bandwidth that different paths would experience, which depends on the local congestion in each neighboring AG. In a worst-case scenario, all elephant flows will be directed to lower bandwidth paths whereas mice flows to the higher bandwidth paths, with severe throughput degradation. Instead, PFB self-adapts the load on each path according to its actual available bandwidth, since it concentrates the flows on the paths with the higher throughput, on which the number of pending flows tends to decrease faster. Whenever a sudden reduction of the available bandwidth is experienced (due for traffic fluctuations and to the implemented traffic scheduler), the corresponding flows will start to experience some temporary starvation and the number of pending flows along the path will not decrease, thus raising the probability that new flows will be routed along alternative paths with better available bandwidth. Thus, PFB is also preferred for asymmetric links, as it balances the load according to the actual available bandwidth of each link. In Section 8.2 we will show an experimental comparison between the performance of the two algorithms, and discuss the performance for asymmetric links.

Note that a flow-level balancing could be inefficient in the case of very few concurrent flows. This is true in general, but in practice the number of active flows by a user is quite large, since many bandwidth-hungry applications open more than one flows, as described later in the evaluation chapter of this part (see Chapter 8). An alternative solution to optimize the routing would be to balance the traffic at packet level. In this case, the only option would be to implement a MPTCP [36] proxy within the AG, as shown in Section 6.4.

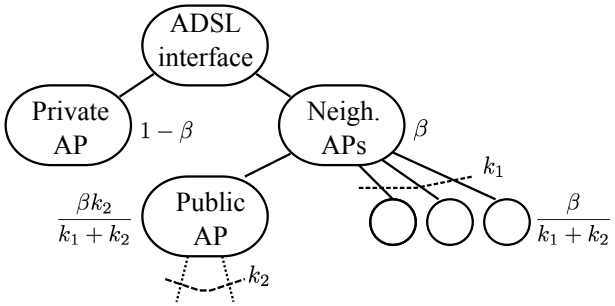


Fig. 6.4 Traffic classes in HTB scheduler with corresponding minimum rates $r_{\min}$, normalized to the ADSL link speed

6.2.3 Traffic scheduling

A work-conserving traffic scheduler regulates the access to the uplink ADSL interface, and manage the set of output queues associated to such interface. This scheme is crucial to guarantee transparent performance for the local devices of an AG. The flows are scheduled at a fine granularity, i.e. at packet level, according to a Hierarchical Token Bucket (HTB) [37, 38]. HTB can be described by a multilayer tree, in which each node corresponds to a traffic class and in particular the root node corresponds to the main interface towards which all the traffic is sent. Each class is controlled by an internal token bucket and the multilayer topology is used to aggregate multiple classes (i.e. each node defines a new class aggregating all the children classes) and to specify detailed scheduling rules on such aggregation. More in details, each class is assigned with a pair of parameters $(r_{\min}, r_{\max})$, where $r_{\min}$ is the minimum average rate and $r_{\max}$ is the maximum rate that cannot be exceed by the traffic within the class. In a nutshell, when the corresponding queues are backlogged, the class traffic will receive at least $r_{\min}$ bandwidth, but no more than $r_{\max}$. The hierarchy allows one child class to increase its rate by borrowing the unused bandwidth from the ancestor class, providing high flexibility to set the minimum and maximum rates for single and groups of traffic classes.

Fig. 6.4 shows the hierarchy among the traffic classes defined in BOB. Consider a generic AG to which $k_1 + k_2$ neighboring AGs are associated; k_1 are associated as stations to the public BOB-AP interface, whereas k_2 are associated through multiple BOB-station interfaces. We set $r_{\max} = 1$ (normalized to the ADSL link speed) for all the traffic classes, in order to exploit all the available bandwidth: the scheduler is indeed fully work-conserving. This choice guarantees also that the unused bandwidth

of the local ADSL connection can be fully exploited by the neighboring nodes. In addition to this, we impose that $1-\beta$ (with a small value of $\beta > 0$) fraction of the local ADSL bandwidth must be guaranteed to the local devices, whenever they are actively sending packets. A small β fraction of the ADSL bandwidth is instead devoted to guarantee that the flows from the neighboring nodes will not starve. This minimum bandwidth is divided evenly across all the $k_1 + k_2$ neighboring AGs. Note that this allocation provides a reasonable level of fairness among neighboring AGs that use different data rates to connect wirelessly to the local AG.

6.3 Implementation

We implemented BOB on Linux and tested it on several laptops with several kernels, namely from 2.6.38 to 3.16. The wireless card was an Atheros AR9285 for which Linux provides a built-in driver (ath9k). We also successfully imported BOB into Arduino YUN with RTL8188CUS wireless card with the default driver provided by Realtek. In the following sections, we describe some relevant design issues for the Linux implementation.

6.3.1 Communication topology

We implemented the topology formation using a Python script that uses the subprocess module to call external Linux commands. In particular, to create multiple virtual interfaces we used `iw` tool.

6.3.2 Flow-level balancing

We implemented the algorithms for the flow-level balancer in Python and run it as a daemon. It chooses the path for each flow, according to the WRR or the PFB scheme. When using the PFB scheme, we use the `conntrack` tool to get the information of the active pending flows.

The main implementation issue to address is how to route in a transparent way the packets according to the flow-balancer decision. To solve this, we adopt the sequence of operations described in Fig. 6.5. Whenever a packet is received by the

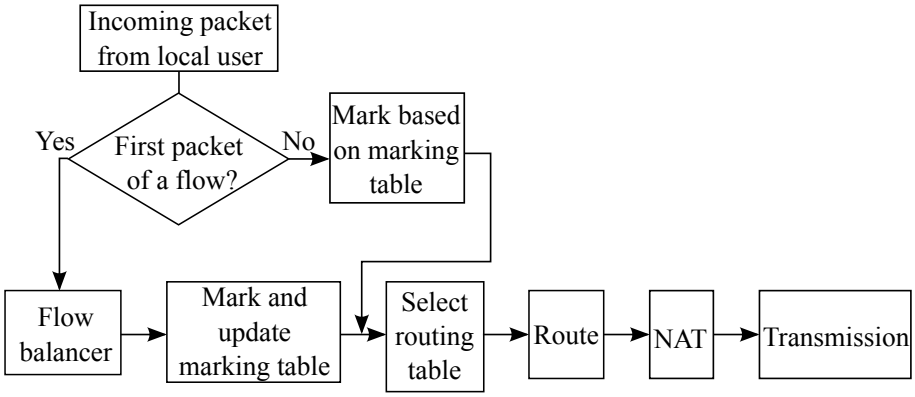


Fig. 6.5 Procedure followed by the flow-level balancer.

AG through the local private AP interface, we have two cases. (1) If the packet is the first of a TCP/UDP flow, the flow-level balancer chooses the path (i.e. the local broadband connection or one of the neighboring AG) to route the packet. The packet is marked through an appropriate `iptables` rule based on the chosen path and this is stored in a marking table. The marking allows to use a specific routing table when exploiting a neighboring AG, according to which the default gateway has been set equal to the IP address of the BOB interface present in the neighboring AG. In addition to this, NAT modifies the source IP address to support the routing back on the reverse direction. (2) If the packet is not the first one of a flow, it is marked according to the marking table to choose the same path of the first packet of the flow, and thus the packet is processed by the desired routing table.

6.3.3 Traffic scheduler

We implemented the traffic scheduler using the native HTB scheduler available in the traffic control `tc` tool [39] available in Linux. Referring to Fig. 6.4, we set $\beta = 0.01$ to reserve just 1% of the bandwidth for the flows arriving from the neighboring AGs. To be able to classify correctly the traffic entering the scheduler, we exploited the same marking capabilities described before to mark a packet based on the incoming interface.

6.4 MPTCP Integration

In the flow-level balancing, when there is only a single TCP connection, it is impossible to split the connection over two different paths due to the NAT on the AGs and to the identification of a TCP connection which is the pair of the IP addresses and ports of the two endpoints. Indeed, when there are few connections generated by the client, as the connections are usually asymmetric, it may happen that the connections are unbalanced over the paths in terms of the load. Therefore, to distribute the load of a single connection on different paths, we introduce MultiPath TCP (MPTCP) in the BOB system.

The integration of MPTCP functions in the AG allows to achieve a balancing of the traffic on the neighboring AGs occurring at packet level instead of flow level. A relevant scenario is an application running at the local user device that opens only one TCP connection towards a server. The practical relevance of such scenario depends on the application: for instance, as we will show in Section 8.2, Google Drive opens multiple TCP connections in parallel, so that access bandwidth aggregation can be performed efficiently without MPTCP. However, when the server, or even intermediate network nodes along the path between the client and the server do not support parallel TCP connections, then MPTCP is a solution that can be implemented by the access network provider as depicted in Figure 6.6. This is possible even if the server is not MPTCP capable, by making use of a concentrator at the core/aggregation network level.

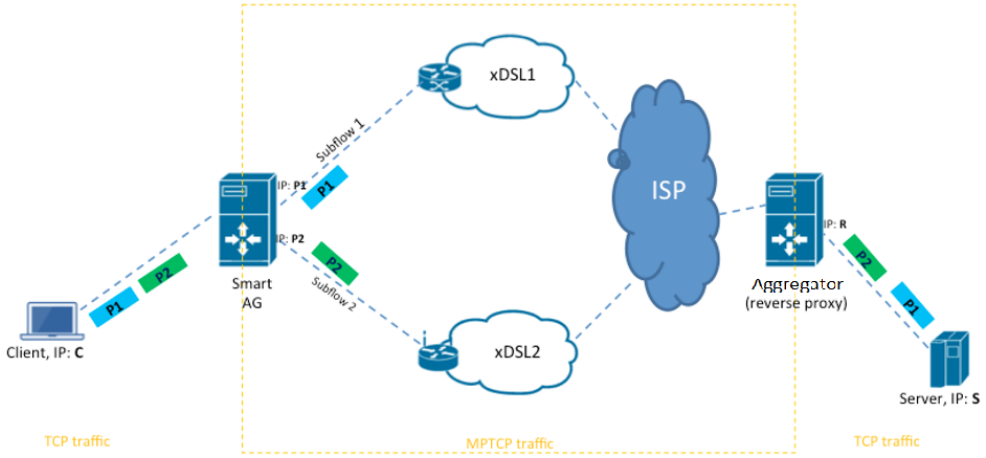


Fig. 6.6 A scenario of MPTCP-based access bandwidth aggregation

In more details, the client opens a standard TCP connection toward a standard TCP server. The traffic reaches the Smart AG that translates it into an MPTCP connection. It is worth mentioning that the multiple subflows created by MPTCP carry standard TCP packets, identified with specific TCP options, so that the probability of being blocked by intermediate network nodes is minimized. It is also possible for the AG to send one subflow to one access network line and the other subflow to another access network link (i.e., passing by a Wi-Fi interface linking to another ADSL access point, in typical residential scenarios). Note that the number of MPTCP subflows and access links can be higher than 2, despite the common reference use-case described where two links are assumed. The MPTCP traffic then reaches a "Concentrator/Aggregator" that terminates the MPTCP connection and retranslates it back to a standard TCP connection directed to the TCP server. In summary, the integration of MPTCP appears completely transparent for the end-to-end application and user.

Based on these high-level specifications, the bandwidth aggregation over multiple access links can be proved to be efficient if the following cases hold:

- when using the standard MPTCP schedulers (using two possible policies: least RTT subflow first, or round-robin), it is required that the differential RTT, i.e., the difference between the RTTs of the longest and shortest path should be minimal;
- when using an advanced MPTCP scheduler, adequately weighting and estimating the travel latency on each path when deciding over which subflow/path to send each packet (as for instance proposed in [40]), the above requirement on the differential RTT is not anymore relevant;
- the system-processing overhead at both the AG and the Aggregator should have no or very low impact on the throughput of the connection. This assumption is supposed to hold practically as the envisaged bit-rate at the AG level is in the order of the 20-30 Mbit/s, while at the Aggregator the switched traffic is in the order of Gbit/s, considering that the dimensioning of an Aggregator is done based on the average number of active users per DSLAM, typically considered to be 500 in modern Ethernet-IP DSLAM/BRAS architectures.

Chapter 7

A Fluid Formulation of the Bandwidth Sharing Scheme

7.1 The Fluid Formulation

In order to assess the performance gain achievable by the proposed scheme, we consider a distributed upload scenario in which the amount of traffic that must be forwarded to the neighboring nodes must be optimally chosen. Notably, forwarding the local traffic to the neighbor increases also the radio contention for the channel and thus reduces the transmission opportunities for the local user. We will show that it is possible to find the optimal fraction of traffic to be forwarded from the local AG to the neighboring ones.

In the following, we describe a fluid queueing model to compute the optimal fraction of traffic to be forwarded to the neighboring nodes. The model can be numerically solved in an efficient way, and the results apply to a broad family of ingress traffic, since the traffic rate is required just to satisfy the law of large numbers (i.e. an average rate can be defined). The results are coarser than a standard queuing model based on Markov chains, but actually we will show its great accuracy in Section 7.2.2 where we will validate the model in an experimental scenario.

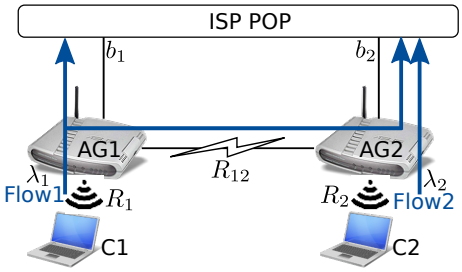


Fig. 7.1 The topology of the single-hop scenario with 2 AGs.

7.1.1 Single-hop formulation

We start by formulating the model of a basic scenario with two AGs, as shown in Fig. 7.1, coherent with the network topology described in Section 6.2.1. Let AG1 and AG2 denote the two AGs and let C1 and C2 be the two local users. The ADSL uplink bandwidth of the two AGs are denoted by b_1 and b_2.

We model all the traffic in this scenario with a first-order fluid model. We assume that the users send the traffic at constant rates, λ_1 and λ_2 respectively, towards the POP. Without loss of generality, we consider the case in which C1 is allowed to exploit the ADSL bandwidth of AG2 for the upload traffic. To model the transmission contention on the shared wireless channel, we define three transmission queues, Q_1, Q_{12} and Q_2, which are associated to C1, the interface towards AG2 at AG1, and C2 respectively, as shown in Fig. 7.2. The traffic generated by C1 firstly enters Q_1. As we allow C1 to use the available bandwidth of its neighbor, part of the traffic departed from Q_1 enters Q_{12}, traverses Q_{12} and goes out through the ADSL link of AG2, while the rest goes out through AG1 ADSL link directly. Since C2 does not exploit the ADSL bandwidth of its neighbor, the traffic generated at C2 just passes Q_2 and exits through its own ADSL link. In our model we do not consider the queues associated to the ADSL uplinks since not affecting the contention for the wireless channel.

According to the 802.11b/g/n protocol, the wireless interfaces adapts the transmission data rate based on the channel condition [41]. Let R_1 and R_2 denote the data rate used from C1 to send traffic to AG1 and from C2 to AG2, respectively. Let R_{12} be the data rate between the two AGs. Notably, the data rate (between 6 and 54 Mbit/s) is not the actual achievable throughput, because of the time sharing among the different transmitters.

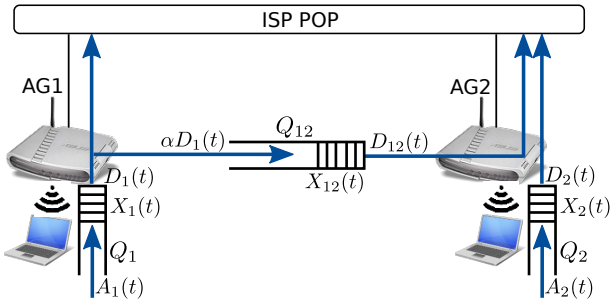


Fig. 7.2 The queueing for the fluid model for the single-hop scenario of Fig. 7.1.

We denote the *forwarding factor* as α, with $\alpha \in [0,1]$, which indicates the fraction of the traffic generated by C1 that AG1 forwards to AG2. We assume α fixed and not changing with the time, since it can be shown that this choice allows optimal bandwidth allocation in the stationary traffic conditions considered in our fluid model.

Consider now queue Q_i. Let $A_i(t)$ denote the cumulative amount of traffic entered the queue from time 0 to time t and $D_i(t)$ the cumulative amount of traffic transmitted from time 0 to time t. Let $X_i(t)$ denote the queue length of Q_i at time t. AG1 forwards a fraction α of the traffic that exits from Q_1, thus we can impose that $A_{12}(t) = \alpha D_1(t)$, for any t. We assume initial null conditions: $A_i(0) = D_i(0) = X_i(0) = 0$, for $i \in \{1,12,2\}$. The evolution of the occupancy of the 3 transmission queues can be described by the standard Lyndley's equations:

$$X_1(t) = A_1(t) - D_1(t) \tag{7.1}$$

$$X_{12}(t) = \alpha D_1(t) - D_{12}(t) \tag{7.2}$$

$$X_2(t) = A_2(t) - D_2(t) \tag{7.3}$$

We now compute the first-order derivative of (7.1)-(7.3). Since the arrival rate is constant, thus $\dot{A_1}(t)$ and $\dot{A_2}(t)$ are the average offered load of C1 (i.e. λ_1) and of C2 (i.e. λ_2), respectively. Thus, we can claim:

$$\dot{X_1}(t) = \lambda_1 - \dot{D_1}(t) \tag{7.4}$$

$$\dot{X_{12}}(t) = \alpha \dot{D_1}(t) - \dot{D_{12}}(t) \tag{7.5}$$

$$\dot{X_2}(t) = \lambda_2 - \dot{D_2}(t) \tag{7.6}$$

We now characterize the departure rates $\dot{D}_1(t)$, $\dot{D_{12}}(t)$ and $\dot{D}_3(t)$ based on a standard and basic model for 802.11b [42], which can be adopted also for 802.11g [43, 44]. Even if the model is very simple, in Section 7.2.2 we will show to provide accurate results also in experimental scenarios. All the Wi-Fi interfaces contend for the same channel, but we assume that the relevant traffic is just from C1 to AG1 and from C2 to AG2. All the traffic in the reverse direction (e.g. TCP ACKs) is assumed to be negligible. Thus, we can consider the radio contention occurring from the traffic from C1 to AG1, from C2 to AG2 and from AG1 to AG2 (which is a given fraction of the traffic generated from C1).

Observe that $\dot{D}_1(t)$, $\dot{D_{12}}(t)$ and $\dot{D}_3(t)$ corresponds to the instantaneous service rate of the corresponding interface, which depends on the actual number of contending interfaces on the same channel. Note that an interface is *active* and competes for the channel if and only if its transmission queue is not empty. For the sake of simplicity, we assume that all the hosts transmit frames of length L. The transmission time t_i of a frame at interface i operating at rate R_i is, by definition, L/R_i. We assume an ideal MAC protocol that would allow a perfect fair access to the channel, i.e. a perfect round-robin for the transmissions among all the interfaces with non-empty transmission queues (i.e. for all j such that $X_j(t) > 0$). We can now approximate the duration $T_{\text{tot}}(t)$ of a cycle of transmissions (according to a round-robin) among all the active interfaces at time t as follows:

$$T_{\text{tot}}(t) = \sum_{j \in \{1,12,2\}} \frac{L}{R_j} \mathbb{1}_{X_j(t)>0} \tag{7.7}$$

where $\mathbb{1}_A$ is the standard binary indicator function, i.e. equal to one iff event A holds. Hence, the instantaneous throughput $\dot{D}_i(t)$ for an active interface at time t corresponds to one frame transmission over a cycle duration T_{tot}:

$$\dot{D}_i(t) = L/T_{\text{tot}}(t) \tag{7.8}$$

By combining (7.7) with (7.8), we can claim

$$\dot{D}_i(t) = \begin{cases} \dfrac{1}{\sum_{j \in \{1,12,2\}} [\mathbb{1}_{X_j>0}/R_j]} & \text{if } X_i(t) > 0 \\ 0 & \text{if } X_i(t) = 0 \end{cases} \tag{7.9}$$

In summary, the evolution of the fluid model describing the dynamics of all the queues involved in the wireless channel contention is obtained by solving the set of equations: (7.4)-(7.6) (for the queue evolutions) and (7.9) (for the instantaneous throughput at each queue).

Given the dynamics of the queues described above, assuming a long enough observation time τ, the average throughput T_1 of AG1 and T_2 of AG2 on the ADSL uplinks are the following:

$$T_1 = \min\left\{b_1, \frac{(1-\alpha)D_1(\tau)}{\tau}\right\} \tag{7.10}$$

$$T_2 = \min\left\{b_2, \frac{D_{12}(\tau)+D_2(\tau)}{\tau}\right\} \tag{7.11}$$

Indeed, by referring to Fig. 7.2, AG1's ADSL throughput (7.10) is obtained by considering just the fraction of local traffic sent from Q_1 to the ADSL link, but clipped to the maximum ADSL rate b_1. Similarly, the AG2's throughput (7.11) is obtained by summing the fraction of throughput arriving from AG1 plus the local user, and finally clipped to b_2. Notably, the actual throughput of all the queues depends on the channel contention level, captured by (7.9).

The aim of the proposed traffic balancing scheme is to find the optimal forwarding factor α^* to maximize the throughput. Formally:

$$\alpha^* = \max_{\alpha\in[0,1]} T_1 + T_2$$

Given the throughput observed at each ADSL link, it is possible to compute the throughput per each user, taking into account that each local user will get highest priority in the traffic to access the ADSL link, as forced by the traffic scheduler implemented in BOB.

Let T_i^u be the throughput obtained by user i associated at its local AGi. Because of the priority of the local user at AG2:

$$T_2^u = \min\left\{b_2, \frac{D_2(\tau)}{\tau}\right\} \tag{7.12}$$

The remaining ADSL bandwidth $b_2 - T_2^u$ at AG2 is exploited by user 1 and thus his throughput is:

$$T_1^u = T_1 + \min\{b_2 - T_2^u, \alpha D_1(\tau)\} \tag{7.13}$$

Later in Section 7.2.1 we will solve numerically the fluid model for the single-hop scenario to get the optimal forwarding factor.

7.1.2 Multi-hop formulation

We extend the single-hop fluid model to the scenario in which 3 AGs cooperate and thus the traffic may traverse up to 2 hops before reaching the ISP POP. In addition to identifying the optimal way to distribute the traffic in order to maximize the throughput, we wish to investigate the achieved fairness among different users.

We consider an access network with three AGs, denoted by AG1, AG2 and AG3 respectively, as shown in Fig. 7.3. Each AG is associated with a single user (C1, C2 and C3, respectively) generating upstream traffic. Furthermore, we have AG1 connected to AG2, and AG2 connected to AG3 through their Wi-Fi interfaces. Finally, as before, all the three AGs are connected to the ISP through ADSL links of which the uplink bandwidth are b_1, b_2 and b_3 respectively. For simplicity, in this scenario C1 exploits the ADSL bandwidth of AG1, AG2 and/or AG3; C2 exploits the one of AG2 and/or AG3; C3 exploits the bandwidth of only its own AG (i.e. AG3). Furthermore, the available bandwidth of an ADSL link is shared equally among the neighboring users (i.e. C1 and C2 have the same priority to access AG3's ADSL bandwidth), coherently with the traffic scheduler adopted in BOB. The users generate the traffic at constant rates, equal to λ_1, λ_2, and λ_3. Using a notation similar to the single-hop scenario, let R_1, R_2 and R_3 denote the Wi-Fi data rate between each user and its corresponding AG respectively. We also have R_{12} denote the Wi-Fi data rate from AG1 to AG2, and R_{23} from AG2 to AG3.

Similarly to the single-hop scenario, we have five transmission queues defined for the wireless interfaces, as shown in Fig. 7.4: Q_1, the queue at C1; Q_{12}, the queue associated to the interface of AG1 towards AG2; Q_2, the queue at C2; Q_{23}, the queue at AG2 on the interface towards AG3; Q_3, the queue at C3. The traffic at rate λ_1 generated by C1 leaves its transmission queue Q_1. A fraction α of it (with $\alpha \in [0,1]$) is forwarded by AG1 to AG2, passing through Q_{12}. The rest just goes out through the ADSL link of AG1 directly. When the traffic from AG1 reaches AG2, a β_1 fraction

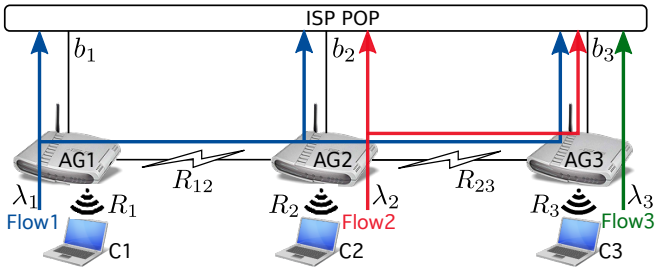


Fig. 7.3 The topology of the multi-hop scenario with 3 AGs.

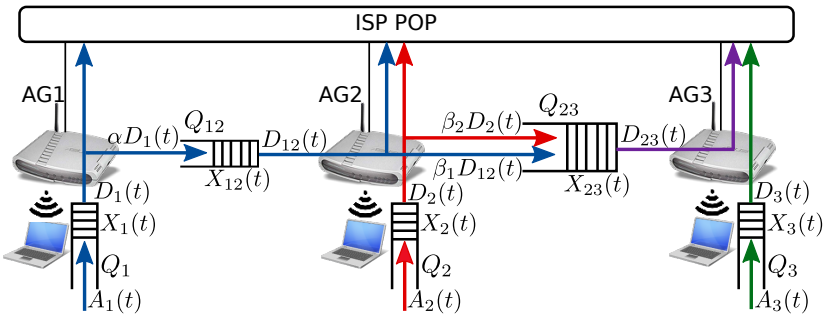


Fig. 7.4 The queueing in the fluid model for the multi-hop scenario of Fig. 7.3

of it (with $\beta_1 \in [0,1]$) is sent to AG3 through Q_{23}, while the rest is sent to the ADSL link. AG2 forwards also a fraction β_2 of the traffic generated by C2 (with $\beta_2 \in [0,1]$), to AG3 through Q_{23}. Finally, AG3 gathers the traffic from Q_{23} and the traffic sent from Q_3 (generated by C3), and sends them out through its own ADSL link.

We start to formulate this scenario by modeling the temporal evolution of the queues. Similarly to the single-hop scenario, $A_{12}(t) = \alpha D_1(t)$ and $A_{23}(t) = \beta_1 D_{12}(t) + \beta_2 D_{23}(t)$ because of the forwarding behavior of AG1 and AG2. Assuming $A_i(0) = D_i(0) = X_i(0) = 0$, $\forall i$, we have:

$$X_1(t) = A_1(t) - D_1(t) \tag{7.14}$$

$$X_{12}(t) = \alpha D_1(t) - D_{12}(t) \tag{7.15}$$

$$X_2(t) = A_2(t) - D_2(t) \tag{7.16}$$

$$X_{23}(t) = \beta_1 D_{12}(t) + \beta_2 D_2(t) - D_{23}(t) \tag{7.17}$$

$$X_3(t) = A_3(t) - D_3(t) \tag{7.18}$$

By taking the first-order derivative of (7.14)-(7.18), we have:

$$\dot{X_1}(t) = \lambda_1 - \dot{D_1}(t) \tag{7.19}$$

$$\dot{X_{12}}(t) = \alpha \dot{D_1}(t) - \dot{D_{12}}(t) \tag{7.20}$$

$$\dot{X_2}(t) = \lambda_2 - \dot{D_2}(t) \tag{7.21}$$

$$\dot{X_{23}}(t) = \beta_1 \dot{D_{12}}(t) + \beta_2 \dot{D_2}(t) - \dot{D_{23}}(t) \tag{7.22}$$

$$\dot{X_3}(t) = \lambda_3 - \dot{D_3}(t) \tag{7.23}$$

Similarly to (7.9), the instantaneous service rate can be evaluated as follows:

$$\dot{D_i}(t) = \begin{cases} \dfrac{1}{\sum_{j \in \{1,12,2,23,3\}} [\mathbb{1}_{X_j > 0} / R_j]} & \text{if } X_i(t) > 0 \\ 0 & \text{if } X_i(t) = 0 \end{cases} \tag{7.24}$$

where $i \in \{1, 12, 2, 23, 3\}$.

By solving the equation set (7.19)-(7.24), we can obtain the evolution of all the queues in our fluid model and compute the overall throughput and the throughput of each flow. Let τ be a long enough observation time; the corresponding throughput T_i of the ADSL link at AGi is:

$$T_1 = \min \left\{ b_1, \frac{(1-\alpha) D_1(\tau)}{\tau} \right\} \tag{7.25}$$

$$T_2 = \min \left\{ b_2, \frac{(1-\beta_1) D_{12}(\tau) + (1-\beta_2) D_2(\tau)}{\tau} \right\} \tag{7.26}$$

$$T_3 = \min \left\{ b_3, \frac{D_{23}(\tau) + D_3(\tau)}{\tau} \right\} \tag{7.27}$$

Indeed, in (7.25) the throughput on AG1 ADSL link of AG1 is a fraction of the traffic from Q_1. In (7.26), the throughput on AG2 ADSL link is the sum of two components: the traffic from Q_{12} which is not further forwarded to AG3 and the traffic from Q_2 which is not redirected to AG3. Similarly, in (7.27) the total traffic arrived at AG3 ADSL link is composed of two parts: the traffic generated by C1, forwarded by AG1 and AG2 to AG3, plus the traffic generated by C2 and redirected by AG2 to AG3; the traffic generated by C3.

Our object is to find the optimal forwarding factors α^*, β_1^* and β_2^* that maximize the overall throughput:

$$(\alpha^*, \beta_1^*, \beta_2^*) = \max_{\alpha,\beta_1,\beta_2 \in [0,1]} T_1 + T_2 + T_3$$

To evaluate the fairness achieved among users, we can compute the throughput for each user, obtained by summing the throughput of all his traffic flows and sent out through all the ADSL links. Let T'_{ji} denote the throughput of user j obtained on the ADSL link of AGi. As the bandwidth of AG1's ADSL link is only exploited by C1:

$$T'_{11} = T_1 \tag{7.28}$$

Based on the adopted traffic scheduler in BOB, on AG2 ADSL link, the throughput contribution for C1 and C2 are the following:

$$T'_{22} = \min\left\{ b_2, \frac{(1-\beta_2)D_2(\tau)}{\tau} \right\} \tag{7.29}$$

$$T'_{12} = \min\left\{ \frac{(1-\beta_1)D_{12}(\tau)}{\tau}, b_2 - T'_{22} \right\} \tag{7.30}$$

Note that (7.30) depends on the fact that C1 has lower priority than C2 in AG2 ADSL link and thus C1 can use only the available bandwidth, if any.

When considering AG3, since the traffic of C3 has the highest priority on the ADSL link:

$$T'_{33} = \min\left\{ b_3, \frac{D_3(\tau)}{\tau} \right\} \tag{7.31}$$

To compute the contribution of the throughput due to C1 and C2 on AG3 ADSL, we must recall the scheduler behavior of BOB. Due to the round-robin selection for neighboring APs (see Fig. 6.4), AG2 will serve the flows directed to AG3 (i.e. traversing Q_{23}) proportionally to the incoming traffic, i.e. β_1 fraction of C1 traffic and β_2 fraction of C2 traffic to AG3. Let λ'_{13} and λ'_{23} be the offered load of C1 and C2 for AG3 ADSL link. Note that from the point of view of Q_{23}, λ'_{13} and λ'_{23} are the

throughput of C1 and C2. Hence we can obtain

$$\lambda'_{13} = \frac{\beta_1 D_{12}(\tau)}{\beta_1 D_{12}(\tau) + \beta_2 D_2(\tau)} \frac{D_{23}(\tau)}{\tau} \tag{7.32}$$

$$\lambda'_{23} = \frac{\beta_2 D_2(\tau)}{\beta_1 D_{12}(\tau) + \beta_2 D_2(\tau)} \frac{D_{23}(\tau)}{\tau} \tag{7.33}$$

According to BOB scheduler, C1 and C2 have the same priority to access the available bandwidth of AG3 ADSL link, i.e., $b_3 - T'_{33}$. Thus let $b'_3 = (b_3 - T'_{33})/2$, we have four cases for the combination of T'_{13} and T'_{23}:

$$(T'_{13}, T'_{23}) = \begin{cases} (\lambda'_{13}, \lambda'_{23}) & \text{if } \lambda'_{13} \leq b'_3, \lambda'_{23} \leq b'_3 \\ (\lambda'_{13}, b'_3 - \lambda'_{13}) & \text{if } \lambda'_{13} \leq b'_3, \lambda'_{23} > b'_3 \\ (b'_3 - \lambda'_{23}, \lambda'_{23}) & \text{if } \lambda'_{13} > b'_3, \lambda'_{23} \leq b'_3 \\ (b'_3, b'_3) & \text{if } \lambda'_{13} > b'_3, \lambda'_{23} > b'_3 \end{cases}$$

Finally, we obtain the throughput of each user by summing the throughput contributed by each ADSL link:

$$\begin{aligned} T_1^u &= T'_{11} + T'_{12} + T'_{13} \\ T_2^u &= T'_{22} + T'_{23} \\ T_3^u &= T'_{33} \end{aligned}$$

7.2 Evaluation and Validation of the Fluid Model

In this section we will discuss the numerical results obtained by solving the fluid model and then we will validate our theoretical findings in an experimental testbed.

7.2.1 Numerical results

We developed a C-language solver for the set of equations describing the fluid model presented in Section 7.1.1 and Section 7.1.2, to study the effect of the different input parameters on the model.

	Wi-Fi			ADSL		User
	R_1	R_{12}	R_2	b_1	b_2	λ_2
Rate [Mbit/s]	54	6	54	2	2	0.5

Table 7.1 Parameter settings for the single-hop scenario.

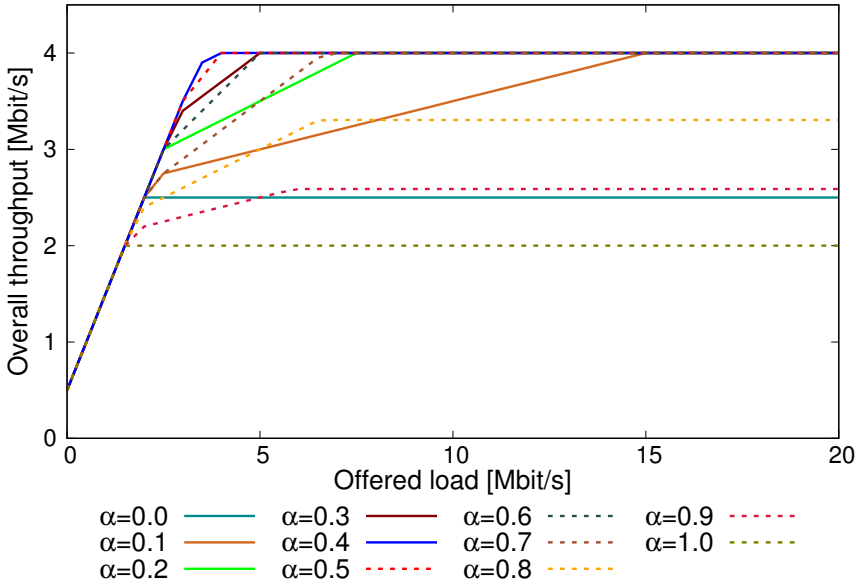


Fig. 7.5 Throughput in the single-hop scenario for different λ_1

Single-hop scenario

We start by considering a single-hop scenario, as shown in Fig. 7.2. The parameters for the scenario are shown in Table 7.1. The data rates of users C1 and C2 are assumed to be 54 Mbit/s which is the maximum data rate that can be achieved in 802.11g. Data rate of AG1 is assumed to be 6 Mbit/s, i.e. the minimum allowed in 802.11g, to describe a realistic scenario in which neighboring APs adopt low data rates due to the obstacles and the high distance. In order to investigate scenarios with available bandwidth to share, we set the offered load of C2 to 0.5 Mbit/s.

Fig. 7.5 shows the overall throughput when varying the offered load of C1 and the forwarding factor α, respectively. When the offered load increases, the overall throughput increases and then saturates at around 4 Mbit/s, which is the maximum achievable in this setting. We now consider the effect of α. We notice that for any load it is possible to find at least one value of α to maximize the performance.

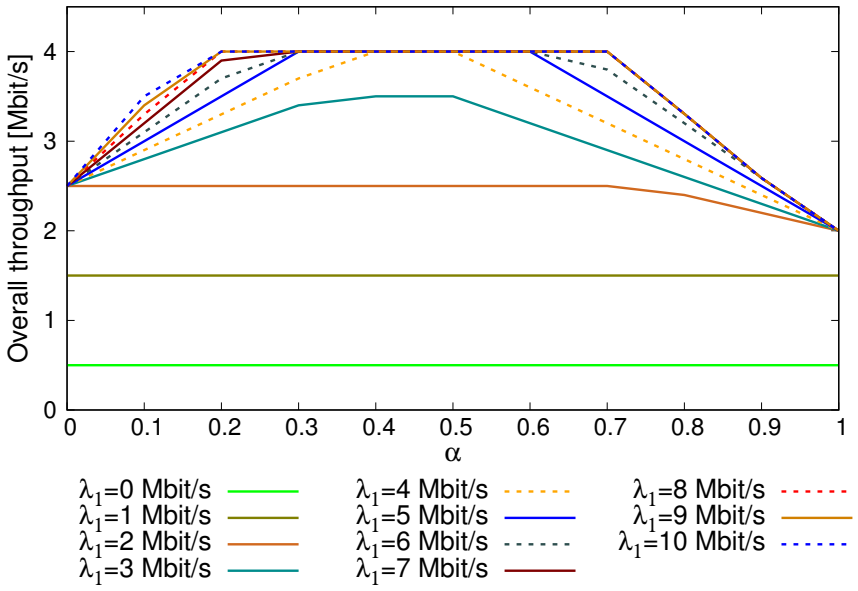


Fig. 7.6 Throughput in the single-hop scenario for different α

Interestingly, the performance degrades when α is too large, since too much traffic of C1 is redirected to AG2 and the ADSL link of AG1 is not fully utilized, which prevents the overall throughput from being maximum. To better highlight the effect of α, Fig. 7.6 shows the performance in function of α. When the offered load is small, i.e. $\lambda_1 \leq 1$ Mbit/s, α has no effect on the overall throughput, since the bandwidth provided by a single ADSL link is already enough to sustain the offered load. When the offered load is large, we have many choices for the optimal value of α, all of them guaranteeing that the ADSL links are saturated. Note that the throughput is a concave function of α and this allows local gradient-based searching algorithms to get the optimal solution for the optimization problem.

Multi-hop scenario

After investigating the single-hop scenario, we now consider the multi-hop case shown in Fig. 7.4, with the parameter setting of Table 7.2. We set $\lambda_3 = 0.5$ Mbit/s in order to leave some spare bandwidth on its ADSL link and thus to enable multi-hop communications.

In the first scenario MH-1, we set $\lambda_1 = 6$ Mbit/s and $\lambda_2 = 2$ Mbit/s. This configuration allows to saturate the local ADSL links of AG1 and AG2. We set

Parameter		Scenario			
		MH-1	MH-2	MH-3	MH-4
Wi-Fi data rates [Mbit/s]	R_1, R_2, R_3	54	54	54	54
	R_{12}, R_{23}	6	6	6	6
ADSL bandwidth [Mbit/s]	b_1, b_2, b_3	2	2	2	2
Forwarding factors	α	[0,1]	0.4	0.4	0.2
	β_1	[0,1]	[0,1]	[0,1]	1.0
	β_2	0	0.5	0.2	[0,1]
User offered load [Mbit/s]	λ_1	6	6	6	2.5
	λ_2	2	4	2.5	6
	λ_3	0.5	0.5	0.5	0.5

Table 7.2 Parameter settting for the multi-hop scenarios.

$\beta_2 = 0$ since C2 must exploit only its ADSL link and saturate it. Thus, only the traffic of C1 is sent eventually to AG3. Thus we can easily observe the multi-hop behavior for C1 traffic. We varied α and β_1 in the full interval $[0,1]$ with steps equal to 0.1 to get all the possible combinations of these two factors. In all the possible settings, the throughput of C2 and C3 are maximum: $T_2^u = \lambda_2$ and $T_3^u = \lambda_3$, thus the ADSL bandwidth is always guaranteed to the local user, as expected. Fig. 7.7 shows the result of the overall throughput $T_1^u + T_2^u + T_3^u$. As in the single-hop scenario, many combinations of α and β_1 allow to achieve the maximum overall throughput. It is also worth noticing that when β_1 is less than 0.4, C1 traffic is not able to fully exploit the link between AG2 and AG3 and thus the throughput is not maximum. Interestingly, the throughput of C1 is maximized when exploiting also AG3 ADSL (occurring when $\beta_1 > 0$). Fig. 7.8 shows the overall throughput when varying α. When $\alpha < 0.3$, the overall throughput is not maximum, since the traffic forwarded to AG2 is too small, which further makes it impossible to fully exploit the ADSL link of AG3. Similarly, when $\alpha > 0.6$, AG1 forwards too much traffic to the other AGs and thus its own ADSL is not fully utilized. Observing both Figs. 7.7 and 7.8, the overall throughput appears as a concave function of β_1 and α, thus enabling, also in this case, local gradient-based algorithms to find the optimal parameters.

We now consider MH-2 scenario, in which, differently from MH-1, we set $\lambda_2 = 4$ Mbit/s to exceed the bandwidth of the local ADSL link. We evaluated the results for all the combination of the parameters, but for the sake of space we report only the results for $\beta_2 = 0.5$ and $\alpha = 0.4$. Fig. 7.9 shows the throughput of C1 (i.e. T_1^u) and of C2 (i.e. T_2^u) and the overall throughput when varying β_1. When

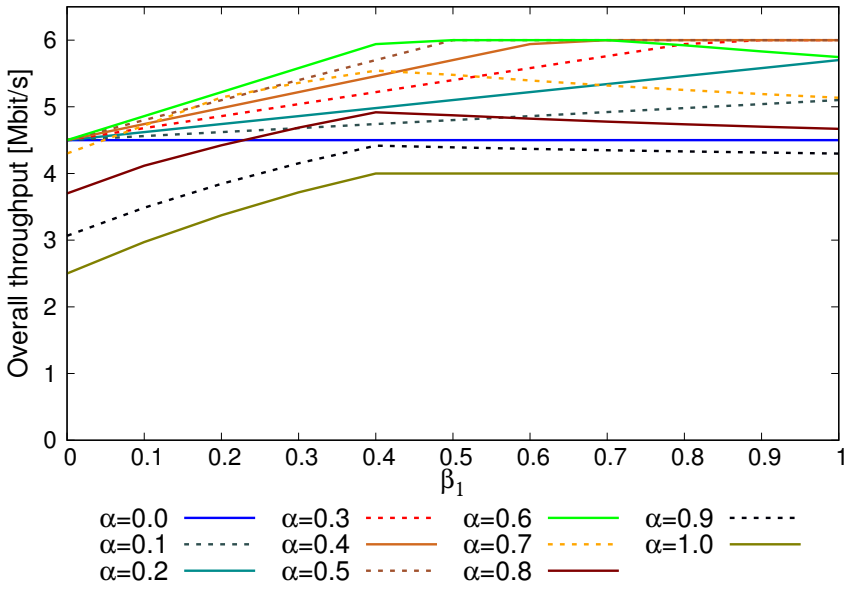


Fig. 7.7 Overall throughput in MH-1 scenario in function of β_1.

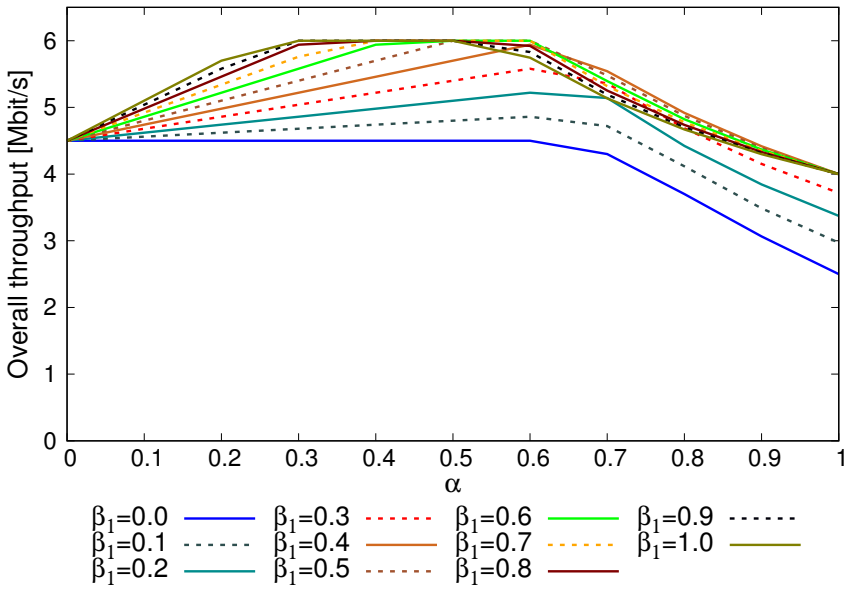


Fig. 7.8 Overall throughput in MH-1 scenario in function of α.

β_1 increases, while the overall throughput is the maximum, the throughput of C1 increases accordingly. This is reasonable as larger β_1 increases the utilization of the bandwidth of C3 by C1. This result proves again that it is worth exploiting multi-hop cooperation in terms of the performance of C1, while not affecting the overall throughput. As the total available bandwidth of AG3 is fixed, the throughput of C2

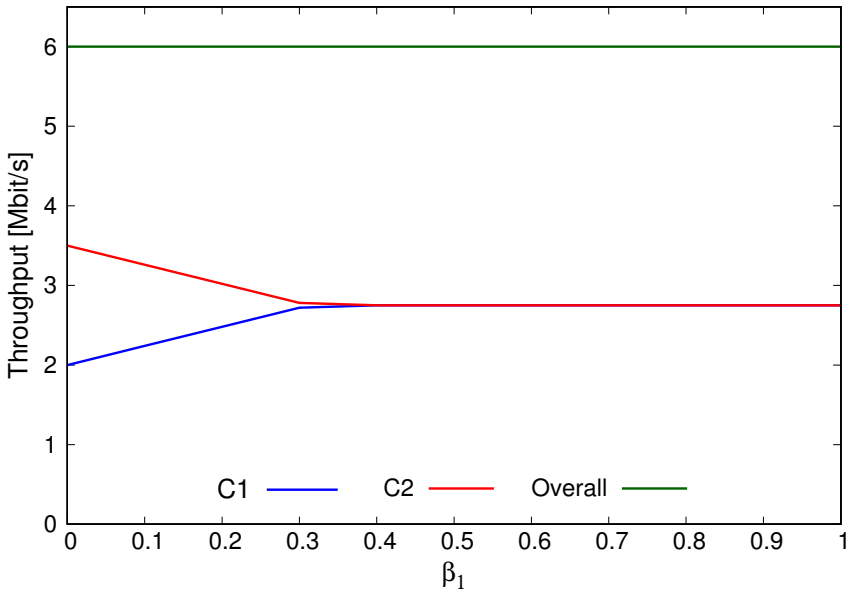


Fig. 7.9 Throughput in MH-2 scenario.

decreases accordingly. For $\beta_1 = 0.3$, C1 and C2 throughputs reach the same value and then remain the same for $\beta_1 > 0.3$. This is due to the scheduling mechanism which provides the same priority to the two traffic flows when accessing AG3 ADSL link, and this confirms fairness of the adopted scheme.

As a comparison, we consider the scenario MH-3, where we set $\lambda_2 = 2.5$ Mbit/s and $\beta_2 = 0.2$. The results are shown in Fig. 7.10. When β_1 increases, i.e., more traffic of C1 is forwarded to AG3, the throughput of C1 grows. However it does not affect the performance of C2 even when β_1 is extremely large. This implies that we can preserve the performance of the flow with the smallest offered load, thus guaranteeing max-min fairness among the contending flows.

In the last scenario MH-4, we set $\lambda_1 = 2.5$ Mbit/s and $\lambda_2 = 6$ Mbit/s. We run the simulations with all the combinations of the three forwarding factors and for the sake of space we report here only the results for $\alpha = 0.2$ and $\beta_1 = 1.0$. In Fig. 7.11 we observe that when $\beta_2 \in [0.2, 0.6]$, the overall throughput is maximum corresponding to the cases in which C2 fully exploits the available bandwidth of C3. As β_2 increases, C2 takes higher bandwidth on AG3 ADSL link, which is also used by C1, and thus the throughput of C1 does not degrade. This is achieved by the scheduling mechanism on the AG3 ADSL link where we assign the same priority

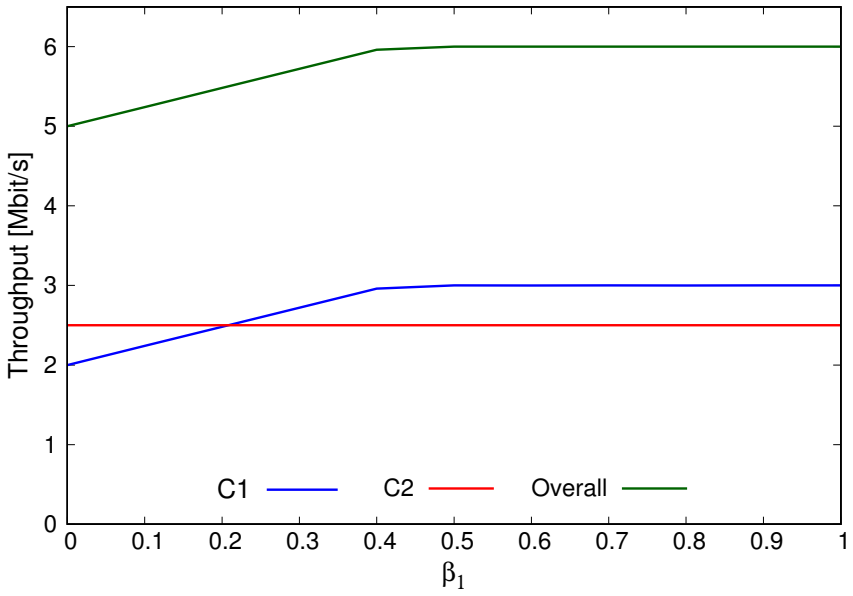


Fig. 7.10 Throughput in MH-3 scenario.

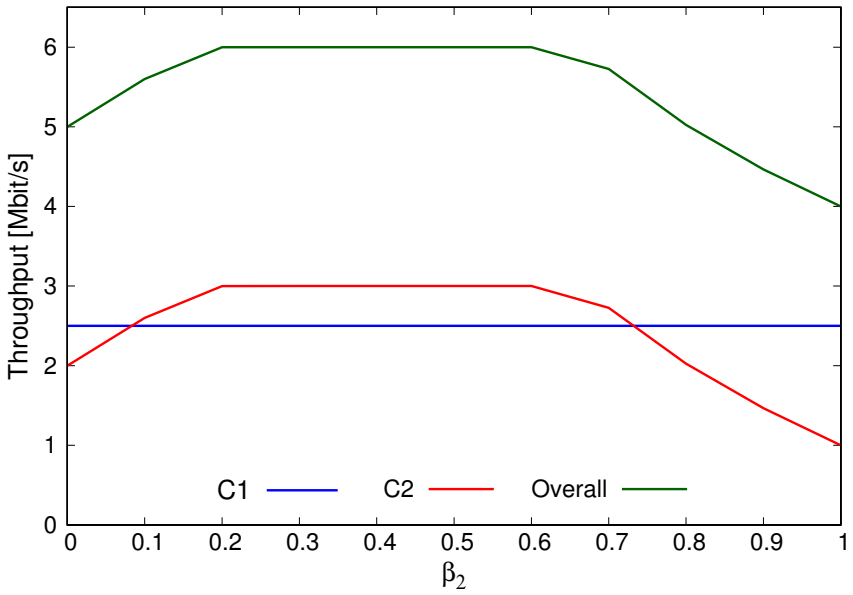


Fig. 7.11 Throughput in MH-4 scenario.

to the traffic coming from C1 and C2. Note that for $\beta > 0.8$, C2 is not able to fully exploit its own ADSL link.

As a summary, our results show that it is beneficial to exploit the multi-hop bandwidth sharing approach, by setting the forwarding factors properly. Moreover,

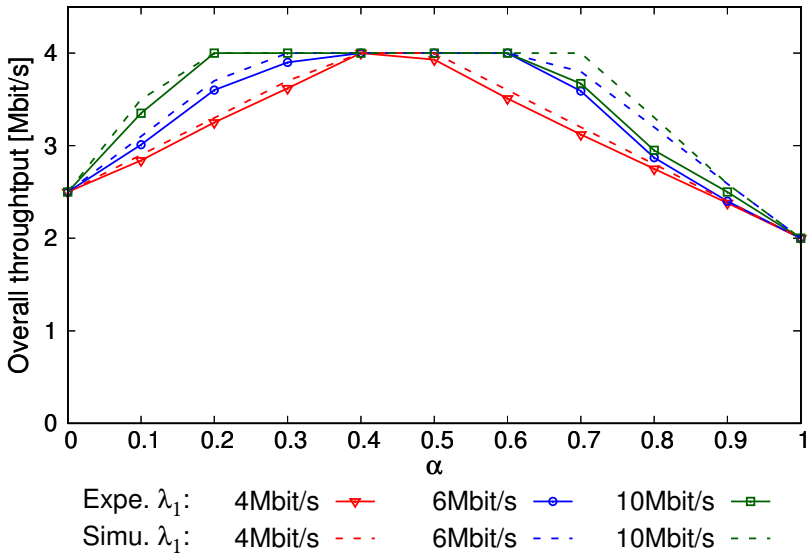


Fig. 7.12 Validation of fluid model in the single-hop scenario. Solid lines refer to experimental results and dotted lines refer to the fluid model results.

our scheme guarantees higher priority to the local user at an AG and achieves max-min fairness among all the traffic flows coming from the neighboring AGs. Furthermore, the throughput is a concave function of α, β_1 and β_2, and this enables the use of simple gradient-based approaches to find the optimal parameter settings.

7.2.2 Experimental validation

We run several tests to validate our fluid formulation model with real single-hop testbed with 2 AGs. Each AG is connected to an independent ADSL modem. We used two Linux laptops to act as AG and two Raspberry Pi with Edimax USB wireless card (with RTL8188 chipset) emulating the users and to setup the topology in Fig. 7.1. In order to emulate different ADSL scenarios, we rate-limited the traffic from the AG to each modem to 2 Mbit/s, through `qdisc` tool provided by Linux Traffic Control [39]. We used `iwconfig` command to fix the data rates of all the wireless interfaces. We generated UDP traffic at constant bit rates for each user. We run all tests at night in order to minimize the interference from campus Wi-Fi and other networks.

To validate the model, we adopted the same parameters for the single-hop scenario of Table 7.1. Fig. 7.12 shows the experimental results obtained in the real testbed and the numerical results obtained by solving the fluid model, for different values of λ_1. By comparing the two curves, the throughput obtained by solving the fluid model appears to be very accurate. This behavior is interesting, given the level of approximation of the model, which is oblivious of the packet nature of the traffic and does not take into account some specific temporal overheads of the Wi-Fi protocol and the collision on the wireless channel. The relative error of the model is always below 10%, with the maximum error achieved when α is large. This is expected, since for small α actually just two interfaces contend for the channel (i.e. just the two users) with negligible collision probability. Instead, for larger values of α, we have 3 interfaces contending and then the collision probability is not anymore negligible and thus the error of the model (oblivious of the collisions) is larger.

Note also that the throughput obtained in the real testbed shows the concave behavior expected according to the fluid model, and thus confirms the validity of adopting a gradient-based approach to find the optimal parameters (α, β_1 and β_2) to distribute the traffic among the cooperating AGs.

Chapter 8

Experimental Evaluation

We adopted the same testbed as the one adopted in Section 7.2.2, but now with 3 AGs. The only difference is that we used a desktop PC with an Edimax USB wireless card (with RTL8188 chipset) to emulate each user. The evaluation mainly focus on: (i) evaluating the maximum performance gain achieved by BOB with real applications, (ii) comparing the two algorithms for flow-level balancing described in Section 6.2.2, (iii) and assessing the performance of HTB scheduler described in Section 6.2.3.

8.1 Performance for cloud storage applications

We select Google Drive [45], OneDrive [46] and Dropbox [47], which are some of the most popular Cloud Storage services, as test applications. To evaluate the performance, we run several tests varying the number of activated AGs and the application. We set the rate of all the ADSL links to 2 Mbit/s. In each experiment, we upload 10 files through a cloud storage application, with an average file size of 50 MB. We capture the traffic at the ADSL interface of the local AG and of neighboring AGs, considering just the packets directed to the specific IP addresses of the servers adopted by the applications. The list of IP addresses of the servers of a specific applications were obtained with `netstat` tool. Since all the considered applications adopt TCP as transport protocol, we were able to evaluate the upload throughput by considering the acked sequence numbers at TCP level.

Table 8.1 shows the *throughput gain*, defined as the ratio between the actual bandwidth obtained when BOB is enabled and the one when BOB is not enabled.

Table 8.1 Performance achieved by BOB during file upload of cloud storage services for WRR and PFB flow balancing

Scenario	Throughput gain		
	GoogleDrive	OneDrive	Dropbox
2 AGs	1.98	1.99	1.00
3 AGs	2.96	2.97	1.00

Table 8.2 Total upload time for 100 files under different load balancing schemes

Scenario	Upload time [s]		
	Minimum	PFB	WRR
Balanced	235.1	264.4	283.2
Unbalanced	235.1	272.3	357.7

The results for WRR and PFB flow-balancing are exactly the same. For GoogleDrive and OneDrive, BOB increases the throughput by a factor equal to the number of cooperative AGs. This large gain is due to the fact that GoogleDrive and OneDrive open multiple TCP connections [48] while uploading files and thus the flow-level balancer is able to exploit fully the available aggregate bandwidth. On the contrary, Dropbox opens only one TCP connection to upload files [49]. Since BOB balances the traffic on per-flow basis, in this specific case it cannot exploit the available bandwidth at the neighboring AG. This highlights the main weakness of our flow-level balancer, which is effective only for many concurrent data flows.

8.2 Performance of flow-level balancing

We utilize two AGs to compare the performance of WRR and PFB algorithms described in Section 6.2.2. In this test, the user sends 100 files to a server and we repeat the experiment 10 times. We set WRR weights to balance equally the traffic across the two possible paths. The size of each file is uniformly distributed between 250 kB and 2.5 MB, and the user opens one TCP connection per file. The user starts to send each file according to a Poisson process at rate 0.4 file/s. We consider two ADSL scenarios. In the first one, denoted as *balanced*, the two ADSL links have the same bandwidth, equal to 2 Mbit/s, while in the second one, denoted as *unbalanced*, the rate of one link is set to 1.5 Mbit/s and the other one is set to 2.5 Mbit/s.

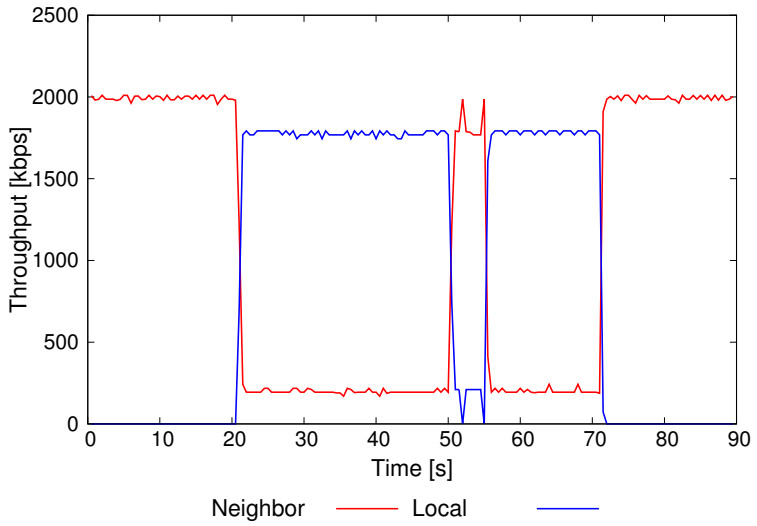


Fig. 8.1 The throughout for the local and neighboring user measured at the Local AG.

Table 8.2 shows the total time to upload the 100 files with WRR and PFB for the two scenarios. The minimum value is calculated theoretically by dividing the aggregate file size by the sum of the two ADSL capacities and provides a lower bound to the upload time. In the balanced scenario, PFB and WRR perform almost the same. WRR assigns to each link the same number of files and the performance degradation with respect to PFB is due to the randomness of the file sizes, that do not allow a perfect load balancing. On the contrary, in the unbalanced scenario, PFB greatly outperforms WRR due to the ability to adapt to the available bandwidth, by considering the number of pending flows during the flow allocation. Since the available bandwidth of a real BOB system is expected to frequently change, PFB is expected to be the best choice to adopt in a real system.

8.3 Performance of the traffic scheduler

We test the behavior of HTB scheduler with 2 AGs. We let one user to associate to each AG. We set both ADSL capacities equal to 2 Mbit/s. We set β in HTB equal to 0.1, thus assuring 1.8 Mbit/s to the local user and 0.2 Mbit/s to the neighboring user. We let the neighbor user to continuously upload multiple files using OneDrive. As described in Section 8.1, OneDrive opens multiple TCP connections and BOB is able to fully exploit the ADSL link of the local AG.

Fig. 8.1 shows the throughput of the two users measured at the local AG. Initially, the AG correctly provides all its ADSL capacity to the neighbor user, since the local user is inactive. At time 20s, the local user starts a short upload using Dropbox and he is able to instantaneously get the allocated bandwidth of 1.8 Mbit/s, while the throughput of the neighboring user simultaneously drops to 200 kbit/s. After the local user finishes the upload, the neighbor fully regains the available bandwidth of the local AG. The result shows that the bandwidth is allocated as expected, i.e., according to the HTB settings. Besides this, it shows that HTB reacts very fast when the local user starts to generate traffic. This clearly shows that the traffic scheduler is able to guarantee a transparent behavior for the local user, which is affected by the neighboring users by a small throughput degradation, that can be easily controlled by the β parameter.

8.4 Smart AG MPTCP functions test and validation

We have deployed MPTCP Proxy features described in the previous section on our Smart AG prototype and carried out two sets of experiments to evaluate the performance of MPTCP in BOB application scenario. As shown in Fig. 8.2 and Fig. 8.3, the two sets of experiments share the same basic BOB architecture with two AGs denoted by AG1 and AG2 respectively. AG1 is implemented on a ASUS 1015B PC with Atheros AR9285 wireless chipset while AG2 is implemented on an Arduino YUN prototype. Each AG is connected to the ISP DSLAM with VDSL connections that are able to deliver a maximum rate of 3 Mbps for uplink and 30 Mbps for downlink.

We exploit a single client associated to AG1 which generates TCP traffic towards a dedicated server. An MPTCP proxy is implemented in the AG1, translating the legacy TCP connections generated by the Client into MPTCP connections. For each legacy connection, two MPTCP subflows are created: one is routed through the local xDSL interface and the other through the wireless interface towards the neighbor AG, i.e., AG2 and finally over the xDSL interface of AG2.

As for the server part, in the first case, we exploit an MPTCP server a public MPTCP Server running on the Internet [50]. This could correspond to the cases with provider-delivered intra-domain services, or to a case where the Internet server is indeed MPTCP capable (some commercial services start having such a feature

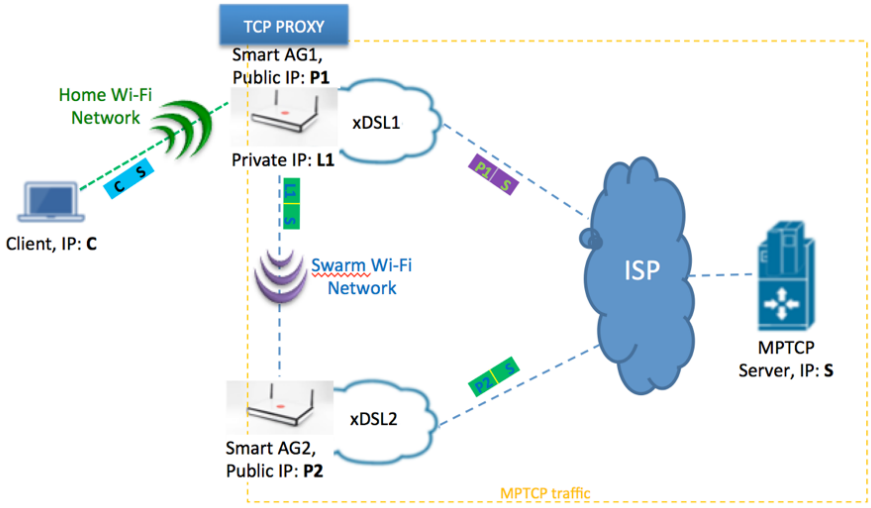


Fig. 8.2 The topology of the test where a TCP client connected to a local VDSL AG opens multiple TCP connections towards an MPTCP server over the Internet.

indeed). In the second case, we exploit a legacy TCP server over the DSLAM, ahead of which an MPTCP Aggregator is deployed, aiming at simulating the common scenario where the server is not MPTCP-capable. In order to simplify the test, the Aggregator is co-located with the TCP server on the same machine.

We utilize iperf2 [51] to open TCP connections at the client towards the dedicated servers. We evaluate the behavior of the MPTCP proxy by monitoring the traffic on the two network interfaces of AG1 and capturing the throughput with Wireshark 2.0 [52]. We measure the RTT from the client to the servers with the `ping` command on Linux.

In each case of experiments, we run tests in three scenarios in which we manipulate the uplink bandwidth of the two VDSL links, as shown in table 8.3. In the first two scenarios, we limit the bandwidth to specific values with the traffic control `tc` tool available in Linux. In the third scenario, we do not limit the uplink bandwidth. Given the characteristics of our VDSL link, the maximum uplink bandwidth is around 3 Mbit/s. Note that we do not simulate neighbor link perturbation for these tests.

Fig. 8.4 shows the throughput of each subflow at AG1 in the first sets of experiments where the client opens TCP connections towards the public MPTCP server over the Internet. The RTT between the client and the server of the local VDSL

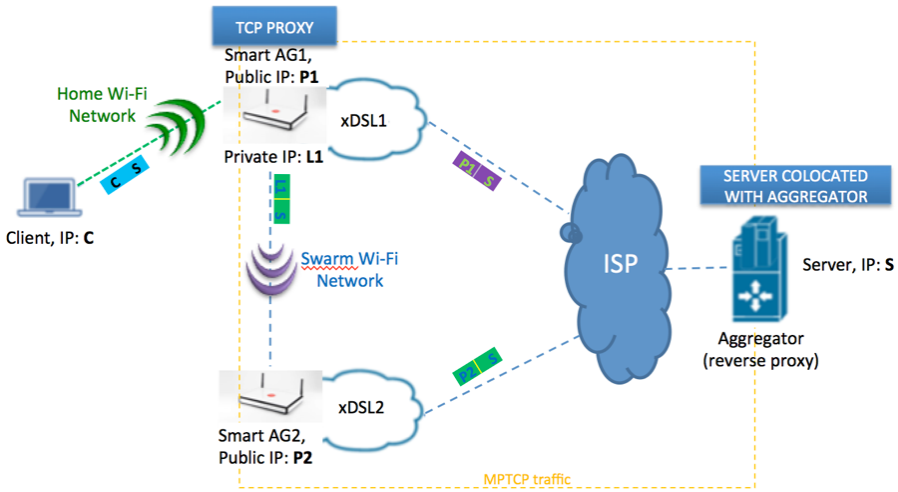
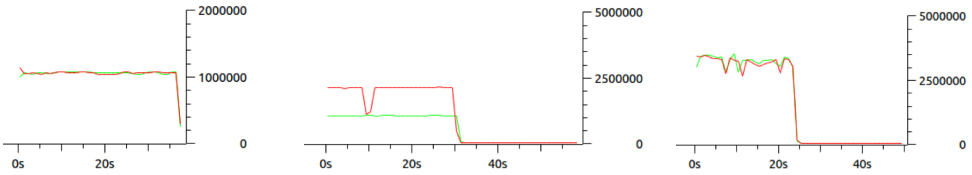


Fig. 8.3 The topology of the test where a legacy TCP server is exploited. The legacy TCP server is behind an MPTCP Aggregator that also serves as a reverse MPTCP proxy. The TCP client connected to a local VDSL AG opens multiple TCP connections which are aggregated by the aggregator to a single TCP connection.

Table 8.3 Different uplink bandwidth test scenarios.

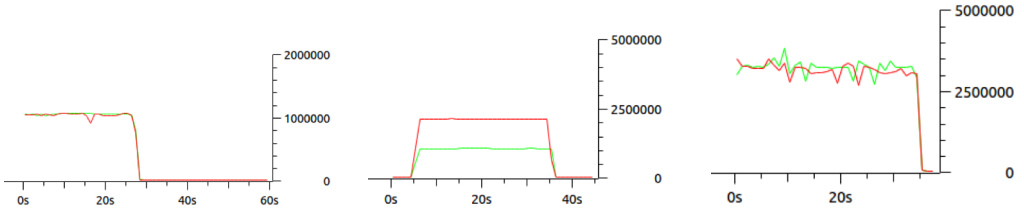
Scenarios	Uplink Bandwidth (Mbps)	
	VDSL 1	VDSL 2
1 Symmetric	1	1
2 Asymmetric	2	1
3 Unlimited	~ 3	~ 3



(a) Symmetric scenario (b) Asymmetric scenario (c) Unlimited symmetric scenario

Fig. 8.4 The result of the tests in which the client opens TCP connections towards a public MPTCP server over the Internet.

path is around 49 ms and that of the neighboring path is 58 ms. In the figures, the throughput of the local VDSL interface is reported with the red line and the wireless interface towards the neighbor with the green line. In the first two scenario, both the two subflows of the MPTCP connection can achieve the maximum limited bandwidth and the overall throughput can be boosted to 2 times and 1.5 times of the local VDSL



(a) Symmetric scenario (b) Asymmetric scenario (c) Unlimited Symmetric scenario

Fig. 8.5 The result of the tests where a legacy TCP server is utilized which is behind a TCP aggregator.

bandwidth respectively. Note that although the two subflows have different RTT, the neighboring subflow can still obtain the maximum bandwidth, in both the symmetric and asymmetric scenarios, which proves the feasibility of introducing MPTCP in the BOB system. In the third scenario, since the bandwidth is not limited to a specific value, given the feature of the DSL, the throughput vibrates with time.

The throughput captured in the second sets of experiments is shown in Fig. 8.5. Measured RTT from client to server on each of the two paths are around 24 ms and 33 ms respectively. Similarly to the one in the previous experiments, the result demonstrates the feasibility of deploying the MPTCP proxy in the BOB system and the MPTCP Aggregator at the edge of the ISP network.

Chapter 9

Conclusion

In this thesis, we provided the cooperative data transfer approaches as part of the 5G technology aiming at a significant performance gain in terms of link reliability, energy consumption, system capacity, spectral efficiency and system coverage.

As the out-door solution, we considered the D2D communication. We implemented bidirectional, multi-group communication in Android devices supporting the recent Wi-Fi Direct protocol. This allowed us to extend the achievable communication range for a protocol whose current implementation in off-the-shelf, unrooted Android devices has been tailored just to single group D2D communication.

In particular, we proposed a solution to overcome the limitations of the physical Wi-Fi Direct network topology and of its addressing plan, and we built a logical topology that enables bidirectional inter-group data transfers. The logical topology we devised is based on a cooperative traffic relaying scheme among adjacent groups and, through transport-layer tunnels, leads to the formation of a network backbone that provides full network connectivity. We also devised a content-centric routing scheme, which properly exploits the above backbone and allows content advertisement, discovery and retrieval in arbitrary D2D network topologies. We implemented our solution in Android and validated it by developing a testbed comprising a heterogenous set of devices. To our best knowledge, our work is the first one enabling a content-centric network with multi-group communication on legacy smartphones, without the facilitation of existing infrastructure.

We proposed a smart group formation mechanism according to which devices can establish a physical multi-group network in a fully distributed manner, focusing

on the performance of the built network: group sustainability, content transfer throughput and network coverage. In addition, in order to establish the logical topology, our mechanism also includes a procedure to select the relay clients. Finally, we implemented our smart group formation mechanism by a four-phase operation on unrooted Android devices, exploiting only the procedures defined in Wi-Fi Direct and the existing Wi-Fi Direct functionalities on Android OS.

Several practical scenarios can benefit from our approaches: resource sharing, context-aware applications, public safety data dissemination and exchange in natural disasters, etc.

Our work opens up several future research directions. Firstly, an in-depth study could be carried out to determine the system scalability with the number of network devices. Such study could also factor in the choice of nodes to be elected as relay clients (their number and typology) as well as the techniques to efficiently manage the consequences of node churning. Secondly, bidirectional, inter-group communication can be the basis for disruptive cooperative applications and service models. Finally, the device association in Wi-Fi Direct requires certain security credentials. Further studies are needed to understand how to distribute the security credentials seamlessly among the devices.

For the in-door usages, we proposed BOB, i.e. Beyond One's Bandwidth, a distributed system composed of cooperative Access Gateways (AGs), which can improve the Internet upload speed for ADSL residential users. Differently from previous solutions, our system is totally transparent to the users. We presented the basic architecture and the communication topology of the system. We designed a load balancer that distributes the traffic at flow level according to two schemes: Weighted Round Robin (WRR) and Pending Flow Balancing (PFB). We also designed a work-conserving traffic scheduler that exploits a Hierarchical Token Bucket (HTB) scheduler to achieve a transparent behavior for the local users of the AG.

We developed a fluid model to investigate the effect of the traffic balancing scheme on the throughput experienced by each user. The model was shown to be accurate in a real scenario, even if based on many simplifying assumptions. This model also opens the door to have the AGs make decisions on the forwarding in a distributed manner according to the level of utility and certain incentives, provided that the information of neighboring AGs can be shared.

To validate our design, we implemented a prototype of BOB on Linux machines and tested the performance of our approach with real applications. Notably, we showed that with some standard cloud storage services, the available bandwidth of the neighboring AGs is fully exploited with a beneficial effect on the file upload time. We also proved the performance gain achievable by our proposed PFB scheduler when the available bandwidth is not evenly distributed across the cooperative AGs. Finally, we showed the fast reactivity of the proposed scheme in preserving the local bandwidth to the local user of an AG.

References

[1] The total number of LTE subscriptions will reach 1.3 billion by the end of 2018. http://www.cellular-news.com/story/Reports/63061.php, 2017.

[2] Bluetooth low energy. http://www.bluetooth.com/Pages/low-energy-tech-info.aspx, 2014.

[3] Wi-Fi Direct Alliance. http://www.wi-fi.org/, 2014.

[4] 3GPP RP-122009. Study on LTE device to device proximity services. 3GPP TSG RAN Meeting #58, 2012.

[5] A. Asadi and V. Mancuso. WiFi Direct and LTE D2D in action. In *IFIP Wireless Days (WD)*, pages 1–8, Valencia, Spain, 2013.

[6] S. Andreev, A. Pyattaev, K. Johnsson, O. Galinina, and Y. Koucheryavy. Cellular traffic offloading onto network-assisted device-to-device connections. *IEEE Communications Magazine*, 52(4):20–31, 2014.

[7] D. Camps-Mur, X. Perez-Costa, and S. Sallent-Ribes. Designing energy efficient access points with Wi-Fi Direct. *Computer Networks*, 55(13):2838–2855, 2011.

[8] Keun-Woo Lim, Woo-Sung Jung, H. Kim, J. Han, and Young-Bae Ko. Enhanced power management for Wi-Fi Direct. In *Wireless Communications and Networking Conference (WCNC)*, pages 123–128, 2013.

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

[10] M. Conti, F. Delmastro, G. Minutiello, and R. Paris. Experimenting opportunistic networks with WiFi Direct. In *IFIP Wireless Days (WD)*, pages 1–6, Valencia, Spain, 2013.

[11] Y. Duan, C. Borgiattino, C. Casetti, C.F. Chiasserini, P. Giaccone, M. Ricca, F. Malabocchia, and M. Turolla. Wi-Fi Direct multi-group data dissemination for public safety. In *World Telecommunications Congress (WTC)*, Berlin, Germany, June 2014.

[12] A. Pyattaev, K. Johnsson, A. Surak, R. Florea, S. Andreev, and Y. Koucheryavy. Network-assisted D2D communications: Implementing a technology prototype for cellular traffic offloading. In *IEEE Wireless Communications and Networking Conference (WCNC)*, Istanbul, Turkey, April 2014.

[13] M. Fiore, C. Casetti, and C.F. Chiasserini. Information density estimation for content retrieval in MANETs. *IEEE Transactions on Mobile Computing*, 8(3):289–303, 2009.

[14] P. Meroni, E. Pagani, G.P. Rossi, and L. Valerio. An opportunistic platform for Android-based mobile devices. In *Proceedings of the Second International Workshop on Mobile Opportunistic Networking*, MobiOpp, pages 191–193. ACM, 2010.

[15] P. Tiago, N. Kotilainen, and M. Vapa. Mobile search - social network search using mobile devices demonstration. In *IEEE Consumer Communications and Networking Conference (CCNC)*, pages 1245–1245, January 2008.

[16] T.N. Duong, N.-T. Dinh, and Y. Kim. Content sharing using P2PSIP protocol in Wi-Fi Direct networks. In *International Conference on Communications and Electronics (ICCE)*, pages 114–118, 2012.

[17] I.M. Lombera, L.E. Moser, P.M. Melliar-Smith, and Y.-T. Chuang. Peer management for iTrust over Wi-Fi Direct. In *International Symposium on Wireless Personal Multimedia Communications (WPMC)*, pages 1–5, 2013.

[18] I.M. Lombera, L.E. Moser, P.M. Melliar-Smith, and Y.T. Chuang. Mobile decentralized search and retrieval using SMS and HTTP. *Mobile Networks and Applications*, 18(1):22–41, March 2013.

[19] Sacha Trifunovic, Bernhard Distl, Dominik Schatzmann, and Franck Legendre. Wifi-opp: ad-hoc-less opportunistic networking. In *Proceedings of the 6th ACM workshop on Challenged networks*, pages 37–42. ACM, 2011.

[20] LTE Direct. https://ltedirect.qualcomm.com/, 2015.

[21] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.

[22] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless networks*, 8(5):481–494, 2002.

[23] Ivan Stojmenovic, Mahtab Seddigh, and Jovisa Zunic. Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Transactions on parallel and distributed systems*, 13(1):14–25, 2002.

[24] Fei Dai and Jie Wu. Distributed dominant pruning in ad hoc networks. In *Communications, 2003. ICC'03. IEEE International Conference on*, volume 1, pages 353–357. IEEE, 2003.

[25] Wi-Fi certified Passpoint. http://www.wi-fi.org/discover-and-learn/, 2013.

[26] How android wifi state machine works. http://jhshi.me/2014/04/25/how-android-wifi-state-machine-works/, 2014.

[27] Lucas DiCioccio, Renata Teixeira, and Catherine Rosenberg. Measuring and characterizing home networks. *SIGMETRICS Performance Evaluation Reviews*, 40(1):383–384, June 2012.

[28] T. Badirkhanli S. Kandula, K.C.J. Lin and D. Katabi. FatVAP: Aggregating AP backhaul capacity to maximize throughput. In *NSDI*, volume 8, pages 89–104, 2008.

[29] Xinyu Xing, Shivakant Mishra, and Xue Liu. ARBOR: hang together rather than hang separately in 802.11 WiFi networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[30] D. Giustiniano, E. Goma, A. Lopez Toledo, I. Dangerfield, J. Morillo, and P. Rodriguez. Fair WLAN backhaul aggregation. In *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, pages 269–280. ACM, 2010.

[31] Eugene Chai and Kang G Shin. Sidekick: AP aggregation over partially overlapping channels. In *Network Protocols (ICNP), 2011 19th IEEE International Conference on*, pages 301–310. IEEE, 2011.

[32] S. Jakubczak, D.G. Andersen, M. Kaminsky, and S. Seshan K. Papagiannaki. Link-alike: using wireless to share network resources in a neighborhood. *ACM SIGMOBILE Mobile Computing and Communications Review*, 12(4):1–14, 2009.

[33] Eduard Goma and Domenico Giustiniano. SmartAP: Practical WLAN backhaul aggregation. In *Wireless Days (WD), 2013 IFIP*, pages 1–7. IEEE, 2013.

[34] C. Rossi, N. Vallina-Rodriguez, V. Erramilli, Y. Grunenberger, L. Gyarmati, N. Laoutaris, R. Stanojevic, K. Papagiannaki, and P. Rodriguez. 3GOL: Power-boosting ADSL using 3G OnLoading. In *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, pages 187–198. ACM, 2013.

[35] Y. Duan, P. Giaccone, P. Castrogiovanni, D. Mana, C. Borean, and C. Rossi. Transparent bandwidth aggregation for residential access networks. In *IEEE GLOBECOM*, 2015.

[36] H. Han, S. Shakkottai, C. V. Hollot, R. Srikant, and D. Towsley. Multi-path TCP: A joint congestion control and routing scheme to exploit path diversity in the Internet. *IEEE/ACM Transactions on Networking*, 14(6):1260–1271, 2006.

[37] Hierarchical Token Bucket theory. http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm, 2016.

[38] R.G. Garroppo, S.Giordano, S. Lucetti, and E. Valori. The wireless hierarchical token bucket: a channel aware scheduler for 802.11 networks. In *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, pages 231–239. IEEE, 2005.

[39] Queueing Disciplines for Bandwidth Management. http://www.lartc.org/howto/lartc.qdisc.html, 2016.

[40] Matthieu Coudron, Stefano Secci, and Guy Pujolle. Differentiated pacing on multiple paths to improve one-way delay estimations. In *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, pages 672–678. IEEE, 2015.

[41] M. Lacage, M.H. Manshaei, and T. Turletti. IEEE 802.11 rate adaptation: a practical approach. In *Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 126–134. ACM, 2004.

[42] M. Heusse, F. Rousseau, G. Berger-Sabbatel, and A. Duda. Performance anomaly of 802.11b. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 836–843. IEEE, 2003.

[43] D. Vassis, G. Kormentzas, A. Rouskas, and I. Maglogiannis. The IEEE 802.11g standard for high data rate WLANs. *IEEE Network*, 19(3):21–26, 2005.

[44] M. Loiacono, J. Rosca, and W. Trappe. The snowball effect: Detailing performance anomalies of 802.11 rate adaptation. In *Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE*, pages 5117–5122. IEEE, 2007.

[45] Google drive. https://www.google.com/drive/, 2016.

[46] Onedrive. https://onedrive.live.com/, 2016.

[47] Dropbox. https://www.dropbox.com/, 2016.

[48] I. Drago, E. Bocchi, M. Mellia, H. Slatman, and A. Pras. Benchmarking personal cloud storage. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 205–212. ACM, 2013.

[49] I. Drago, M. Mellia, M.M. Munafo, A. Sperotto, R. Sadre, , and A. Pras. Inside Dropbox: understanding personal cloud storage services. In *Proceedings of the 2012 ACM conference on Internet measurement conference*, pages 481–494. ACM, 2012.

[50] MultiPath TCP - Linux Kernel implementation. http://multipath-tcp.org/pmwiki.php/Users/UseMPTCP, 2017.

[51] iPerf - The ultimate speed test tool for TCP, UDP and SCTP. https://iperf.fr/, 2017.

[52] Wireshark. Go Deep. https://www.wireshark.org/, 2017.