Dynamic Clustering in WiFi Direct Technology *

Urbano Botrel Menegato
iMobilis
DECOM / UFOP
Ouro Preto, MG - Brasil
urbanobm@gmail.com

Leonardo de S. Cimino
iMobilis
DECOM / UFOP
Ouro Preto, MG - Brasil
leonardocimino@gmail.com

Saul Delabrida
iMobilis
DECOM / UFOP
Ouro Preto, MG - Brasil
saul@sdelabrida.com

Fernando A. Medeiros Silva
fernandoaugusto@gmail.com

Joubert de Castro Lima
HPC Lab
DECOM / UFOP
Ouro Preto, MG - Brasil
joubertlima@gmail.com

Ricardo A. R. Oliveira
iMobilis
DECOM / UFOP
Ouro Preto, MG - Brasil
rrabelo@gmail.com

ABSTRACT

WiFi Direct is a new technology supported by WiFi Alliance. Devices can establish connections using an Access Point (network leader), chosen automatically by the system. Unfortunately, there are no measurements for discovering the best device to be a network leader. In this paper, we propose a dynamic election of leaders for Wifi Direct Technology. In our approach, devices are exposing clustering service information used in the election of leaders. This way, leaders can be replaced based on any clustering strategy, such as battery, speed, direction and many others. We implement classical clustering algorithms to prove the usability of our proposal. The results demonstrate that our proposal is extensible to support cluster election algorithms.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design — *Wireless communication*

General Terms

Algorithms and Experimentation and Standardization

Keywords

WiFi Direct and Cluster and Architeture.

1. INTRODUCTION

WiFi Direct can be considered a new technology, being little explored by the academic community, so many research problems are still open that enable contributions to the technology.

In [1], presented is an overview of the features defined in WiFi Direct and also an evaluation of the expected performance in real scenarios. Authors measured both expected delays for finding other devices and the time to establish connections among them. Two computers were used in the experiments and no mobility was considered. The work of [1] does not address the diversity of scenarios that the technology can provide. An important issue is the election of leaders in WI-FI Direct. The connections among users can occur by the definition of an Access Point that is called the network leader, which is responsible for connecting other devices. The devices found so far in the market do not have any type of technology that allows devices to define the network leader dynamically.

Our work explored the service exposure functionality to publish information for the election of a leader. We focus on using information published by devices to implement and test clustering algorithms known in literature that are focused on mobile networks. Devices considered as cluster heads are similar to leaders and candidates to develop the role of Network Acces Point. A point we considered during our study was that these algorithms are usually tested on simulators that can cause some differences compared with the real scenario. When implementing algorithms, we implemented a programming template to give more flexibility for developing clustering algorithms. This will be presented and explored in this article.

Contributions

We consider that our work provides three main contributions:

1. Introduce the concept of clusterization in WiFi Direct devices networks;
2. Present a template to help future implementations of clustering algorithms;
3. Present an architectural model that provides an abstraction of the complexity of access to WiFi Direct features.

By experiments, we show that our proposal is a feasible alternative. We also present some opportunities and problems of the technology that can be explored in future years.

*The authors would like to thank to Federal University of Ouro Preto, CAPES, CNPq, Fapemig, and SEVA for financial support

MobiWac'14, September 21–26, 2014, Montreal, QC, Canada.

http://dx.doi.org/10.1145/2642668.2642682 .

2. WIFI PEER TO PEER

The growth of mobile and portable devices, such as smartphones, requires communication between two devices without notifications or agreements between users. WiFi P2P does not work the same way as the WiFi Ad Hoc mode does. WiFi P2P implements a peer-to-peer communication over 802.11 infra-structure mode, inheriting all advances obtained by this method. In the infrastructure mode, we have two distinct agents, the client and the access point (AP). The role of each one is well defined and static throughout the communication. In WiFi P2P, these roles are dynamically assigned and any of the devices must implement both functionalities and may even act simultaneously as both client and Access Point.

2.1 Architecture

Communication devices in WiFi P2P are called P2P Devices. So that communication can occur, WiFi P2P devices establish logical groups called P2P groups. These groups are functionally equivalent to traditional WiFi networks; the device that implements the AP (Access Point) function is called P2P Group Owner (P2P GO) while the other network devices are called P2P Clients.

WiFi P2P specification requires P2P GO to execute a Dynamic Host Configuration Protocol (DHCP) server in order to provide IP addresses to clients and only P2P GO can execute the gateway function of other networks, being forbidden the bridge at link level. Finally, P2P GO role cannot be transferred and P2P GO output leads to the end of the group.

2.2 Discovery

WiFi P2P discovery aims to quickly determine which device must be connected. This discovery consists of two phases.

1. *Scan* - Uses the standard process of IEEE 802.11 [2]. In this phase, P2P device discovers legacy WiFi networks and P2P GO.

2. *Find* - Aims to guarantee that two P2P devices searching for partners will be on the same channel so that they can communicate.

The Find phase starts and then an alternation between two modes begins: search, where the device sends Probe Requests in each of the Social Channels; and listening, where the listener device monitors its listener channel using Probe Requests and sends Probe Responses.

2.3 Group Formation

Two devices can establish a P2P group autonomously or negotiating. Camps-Mur [1] defines three kinds of negotiation that can be used: Standard, Autonomous, Persistent.

Group formation happens in two phases:

1. Definition of P2P GO;

   (a) Negotiation - Two devices negociate who will be the P2P GO based on the ability/intention to become a P2P GO.

   (b) Selection - P2P GO is previously and autonomously defined by choosing the application layer or decided in a previous negotiation.

2. Provision of P2P group.

   (a) Use WiFi Simple Configuration (WSC) for exchanging credentials

   (b) Establish P2P group with the correct credentials

At Standard Negotiation two devices discover each other and negociate who will be P2P GO. The devices must agree on who will be P2P GO and on group characteristics, such as operating channel and band. To define who will be P2P GO, devices send a 4 bit GO Intent numeric parameter that summarizes the intention of becoming P2P GO. The device with the highest value wins. For tiebreaker, a TieBreak bit is randomly activated at Group Owner Negotiation Request, that is defined inversely at Group Owner Negotiation Response. The device that sends the Tie Break wins. The value of GO Intent 15 must only be activated when the device requires to be Group Owner, i.e., if it will be the internet access gateway. In case of a tie with value 15, negotiation fails. After discovery and definition of roles, the devices establish a safe connection using WiFi Simple Configuration (WSC) and finally, we use the DHCP for the definition of addresses.

In the Autonomous form, the device automatically creates a P2P Group and becomes the P2P Group Owner. Other devices can meet and connect to the group using traditional methods for discovering WiFi and go straight to the Provision phase and DHCP. Since P2P Group Owner presence will be detected during the Scan phase, alternate search is not required.

In the Persistent form, P2P GO can claim the group is persistent during the formation phase. After doing this, the devices store network credentials and P2P Group Owner data to restart the group at other moments. Specifically, in the discovery phase, if a device detects it has already been part of a persistent group with the partner device, either device may request reinstatement of the group using a P2P Invitation Procedure.

The Invitation Procedure can be used in three cases:

1. A P2P GO invites a device to become a client of the group.

2. A P2P Client invites a device to become a group member of the group it belongs to.

3. A P2P Device requests the other one for the reinstitution of a persistent group of which both have been part and one of the devices is the P2P GO.

2.4 Service discovery

WiFi P2P enables discovery of services at the data link layer. Even before the establishment of groups, devices can exchange information about available services and, based on that, decide the formation of the group.

The discovery service is implemented using a protocol of high level service advertisement, such as the UPNP and the Bonjour [4], transported at the data link layer using the Generic Advertisement Protocol (GAS) specified in 802.11u [1]. GAS ia a data link layer protocol implemented using active public tables. It enables two unassociated 802.11 devices to exchange queries from a high-level protocol. GAS is implemented to work as a generic container, providing fragmentation, blocking and allowing the device that receives the query to identify the protocol being transported.

3. CLUSTERING ALGORITHMS

To conduct the election of the cluster head mobile devices exchange information with each other. The application of cluster leader election requests publication of some service. Another device in the surroundings finds this service. From the moment the devices have the information, the application can start the algorithm for cluster leader election. Local processing is done and the result is published. As soon as the cluster head is elected, all messages must pass through it. We will explain below the clustering algorithms used in our tests.

3.1 Major ID

In the Major ID election clustering algorithm, when the application starts, each node believes it is the leader and publishes this information to other network devices. When the device receives this message from another device, as both believe to be the leader, they do not agree with each other and it is necessary to elect one. And this is repeated for all network nodes. This process makes devices gradually come into a consensus on the cluster leader device, which stabilizes the process.

3.2 Shortest average distance - GEDIR

We implemented an algorithm for electing the cluster head based on the routing algorithm GEDIR [6] (GEographical DIstance Routing). The GEDIR algorithm uses the distance between nodes to choose the best route for a message to travel from one node to another cluster. The route is defined by always choosing the neighbor that has the shortest distance to the destination node. Our implementation first verifies which node has more devices in its range and this will be elected the cluster head. However, if there is a tie between two or more nodes, the device that has the shortest average distance between it and the other nodes around will be elected as the cluster head. We added one more step which is executed if there still is a tie between devices; it elects the one with the major ID as the cluster head.

3.3 MCFA

MCFA is a new algorithm based on mobility information suggested by Akbari [5]. MCFA divides itself between cluster formation and cluster maintenance. In the formation phase, the nodes are discovered and each one calculates the local relative mobility parameter, RM, which is used to define the leader. The nodes with less mobility in relationship with its neighbors are favored for election. In cluster maintenance, the loss of connection with the leader can be addressed by re-affiliation or by restarting the formation process.

4. WIFI DIRECT IMPLEMENTATION

The core of our implementation is changing the usage of API service publishing functionality in order to transfer data between different devices instead of exposing services.

As proof of concept, we implemented classical clustering algorithms which are detailed in section 3. These algorithms were implemented using the template, described in section 4.1. We specifically created this in order to fast prototype clustering algorithms. The applications developed in this work have been installed and tested on a set of devices configured with the operating system Android version 4.1.x.

4.1 Clustering algorithms template

To implement an algorithm for electing cluster head using WiFi Direct, the challenge is how to send and receive messages. The algorithms must usually be implemented taking into account only the peculiarities of WiFi Direct technology. Given this scenario, we created a "template" for implementing clustering algorithms to facilitate overcoming WiFi Direct particularities. The pseudo-code of algorithm 1 shows the template structure.

```
initialize data;

while true do

  publish my data (Ex: id, coordinate)

  collect other device data;

  if collected data changed then

    elect new leader { algorithm to be implemented }

  end if

end while
```

Algorithm 1: Pseudo code of the template

The "template" is divided into four steps:

1. Initialization (command "initialize data;"): some basic data can be configured, for example, in the algorithm of highest ID, the variable "myLeader" receives the name of the device name;

2. Publishing (command "publish my data"): devices publish their information;

3. Collecting data (command "collect other device data"): devices collect information from other devices;

4. Leader election (command "elect new leader"): it executes the election of the cluster leader.

The information published/collected refers to the clustering algorithm being implemented. For example, at the algorithm of highest ID, the device ID and the cluster leader are published. In the beginning, as there is still no defined leader, the election process needs to be executed. After, the algorithm verifies if the collected data were changed. If it is yes, the election process must be executed. The application runs in an infinite loop to ensure that even if the leader is out of range or turned off, the algorithm detects this event and a new election is executed, providing greater robustness and fault tolerance.

To send a message, it is necessary to add a local service so that the other devices can find it. It is important to highlight that when an information is published, it is exposed to any device.

To search for publications from other devices, i.e., receive messages, it is necessary to start a local process that will be listening to notifications from other devices.

After the end of election, to effectively have cluster formation, it is necessary to create a group. If the cluster head

is changed, it is necessary to delete the group and recreate it. The creation and deletion of the group must be made by the device that has been elected as cluster head.

4.2 Project architecture

During the development of our project we defined an architecture that clearly illustrates the complexity of using WiFi Direct and also to provide specific modules in order to develop a specific group of applications.

The project provides the necessary infrastructure for implementing and testing clustering algorithms. Some basic features have been added such as: publishing information, searching for information, list nearby devices and their services. They are the foundation for development of the project modules.

The project architecture central components are "Publisher", "Consumer" and "Information Manager". In addition to the architecture core components, the cluster head election application has the following components: "Producer", "Observer" and "Cluster Head Election Module". The Figure 1 shows these components and the interaction among them.

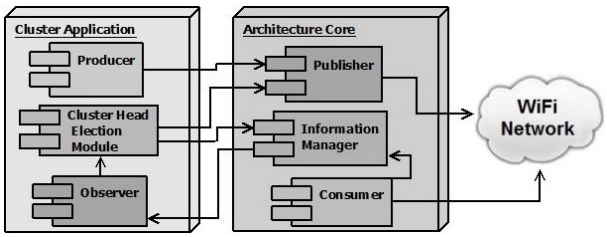

Figure 1: Cluster Architecture

The application requests publishing any service to the "Publisher" component. Another surrounding device finds this service using the "Consumer". The "Consumer" sends data to be stored by the "Information Manager". At this moment, the "Observer" receives a notification from the "Information Manager" with data from the new service or from the altered service. From the moment the "Observer" has the information received from the "Information Manager" , the application can initiate the required algorithm. After processing the algorithm, the result is published by the "Publisher" component. The parameters "information name" and "value of information" are needed in the publications.

To conduct a search for publications from other devices, i.e., to receive messages, it is necessary to use the "Observer" component. With it, we have access to devices found, values added to information and changes in the value of this information.

Another feature of the project is the access to the list of all devices known by a specific cluster node formed or to be formed. If no message from a device is received during a predefined period of time, we believe that this device went offline. Before, during and after the cluster head election, all devices send at least one ping message to inform that they are still active. This is one way to ensure fault tolerance in algorithm implementation.

In addition to the list of devices, it is possible to access all the services that have been published by all other devices. When we receive a service that has not been published yet, the service is included. When the device already has some information about a certain service, it is changed. If it is necessary to create or delete a group, WiFi-Direct API must be used.

4.3 Experiments

To validate the architecture and test the clustering template via WiFi Direct, we used the algorithms "Major ID", "GEDIR" and "MCFA" for leader election. The first tests were performed using an implementation of the algorithm "Major Id" installed in 4 devices.

Several executions for two types of scenarios were made and were illustrated in Figure 2.

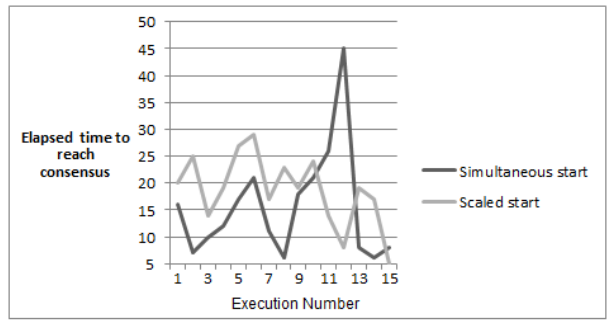

Figure 2: Time spent for all to reach a consensus

In the first scenario, named "Simultaneous Start", all devices initiated the application simultaneously. In the second scenario, called "Scaled Start", the node of highest ID was initiated only when the other devices already had a formed and stabilized cluster.

We could notice from the results that in both scenarios the election occurred correctly, i.e., all of them reached a consensus on the cluster leader and the most adequate device for the algorithm was elected. In the scenario "Scaled Start", devices were able to interpret the arrival of a new device with the highest ID and elect it as the new leader even after stabilization of the old cluster.

The election occurs in several steps. During application execution, creating more than a cluster, until all devices reached a consensus, was normal. This occurs because the devices may have received no message from the node that should really be the leader or because at some point, they understood that the leader was offline. On both cases, what happens is a communication problem, since there is no guarantee that all sent messages will be retrieved by all other devices. However, in the end they all agreed on the leader. In the tests with the MajorID algorithm in approximately 87.5% of the time, the correct device was kept as a leader with an oscillation in the remaining time.

In the scenario "Simultaneous Start" the average time spent until all devices came into a consensus on the leader, took about 15.12 seconds. In the scenario "Scaled Start" the average time for all nodes to notice that the leader should be changed, took 18.4 seconds. We realize that more time is spent to redefine another cluster from one already formed ("Scaled Start" scenario) than to start all of them simultaneously.

A comparative test among the three algorithms considering the time spent for each cluster leader election was also made. The graph in Figure 3 shows the percentage distribution of clusters generated until a specified time (Cumulative

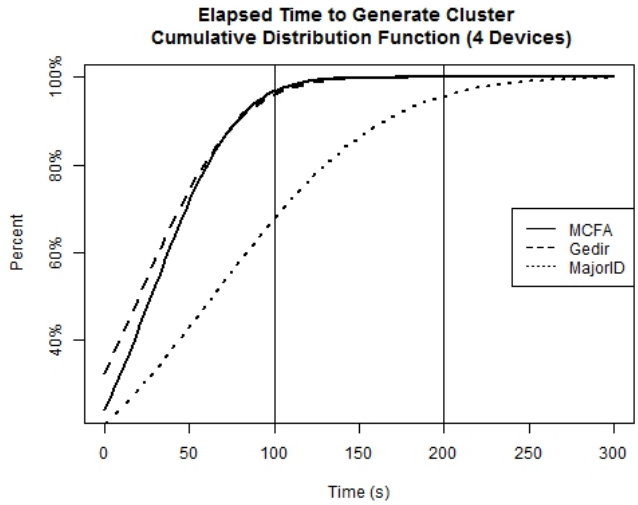

Figure 3: Elapsed Time to Generate Cluster - CDF

Distribution Function - CDF). We can observe that in algorithms "GeDir" and "MCFA", almost 100% of clusters are generated in less than 100 seconds, and the algorithm of "MajorID" gets close to 100% in 200 seconds.

The graph in Figure 4 shows how much time to generate a cluster. We can observe that in algorithms "GeDir" and "MCFA", the time for generating a new cluster is around 25 seconds, and in the algorithm "MajorID", it is approximately 60 seconds.

As expected, the algorithm "MajorID" is more stable since the time interval to exchange cluster heads is longer. This is due to the variation of position generated by the algorithms "Gedir" and "MCFA". As "Gedir" and "MCFA" use the device coordinates to verify the distance between each one; this problem may be related to the accuracy of GPS. This problem might not be identified if the implementation was tested in a simulator instead of a real scenario.

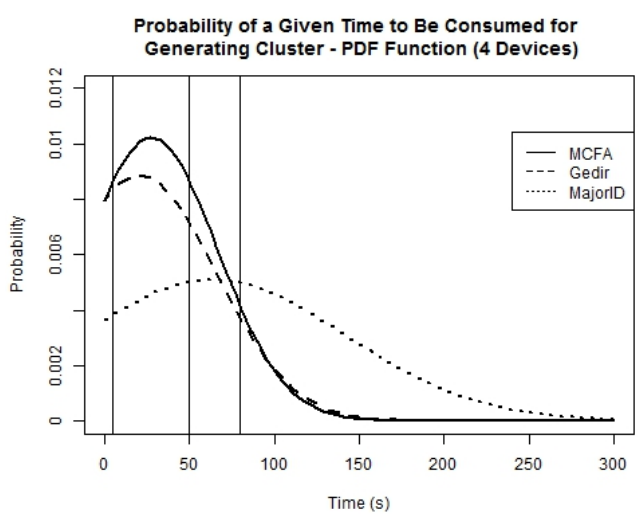

Figure 4: Probability of a Determined Time be Consumed to Generate Cluster - PDF

5. CONCLUSION AND FUTURE WORK

Our proposal has a relatively simple implementation and enables implementing and testing algorithms directly in devices and in field conditions allowing more realistic results when compared to those obtained currently via simulators. Our work can open up a range of possibilities for using a technology previously underused and underestimated.

During the project and implementation of our proposal for using WiFi Direct, we observed several important aspects of the technology. Among these aspects we can mention the little use of WiFi Direct and its potential to be explored.

We believe our work can positively influence the use of WiFi Direct in areas such as mobile computing, distributed systems, and other similar areas. As WiFi Direct technology has been still little explored, there are several opportunities for future work. Improvements can be made in our solution so that it will be possible to implement a wider range of algorithms. Studies can be made to propose improvements in WiFi Direct specification in order to provide greater information traffic between devices before they connect to each other. Testing algorithms can be made aiming at reducing the use of device resources such as battery, processor, and so on.

6. ACKNOWLEDGEMENTS

Acknowledgements to CAPES, CNPq, Fapemig, UFOP, SEVA.

7. REFERENCES

[1] D Camps-Mur; A Garcia-Saavedra; P Serrano, Device to device communications with WiFi Direct: overview and experimentation, IEEE Wireless Communications Magazine, 2012.

[2] Wi-Fi Alliance, P2P Technical Group, Wi-Fi Peer-to-Peer (P2P) Technical Specification v1, 2009.

[3] Fernando Augusto Medeiros Silva and Ricardo Augusto Rabelo Oliveira, GLUE: A mobility-based algorithm to smartphones cluster maintenance in tourism environments, The 11th ACM International Symposium on Mobility Management and Wireless Access (MobiWac 2013). November, 2013.

[4] Edwards, W Keith, Discovery systems in ubiquitous computing, Pervasive Computing, IEEE, 2006, vol. 5, number 2, pages 70 - 77.

[5] J. Akbari Torkestani and M. Meybodi, "A mobility-based cluster formation algorithm for wireless mobile ad-hoc networks," Cluster Computing, vol. 14, pp. 311-324, 2011, 10.1007/s10586-011-0161-z. [Online]. Available: http://dx.doi.org/10.1007/s10586-011-0161-z

[6] I. Stojmenovic and X. Lin, "GEDIR: Loop-Free Location Based Routing in Wireless Networks," Proc. IASTED Int'l Conf. Parallel and Distributed Computing and Systems, Nov. 1999.

[7] I. Stojmenovic and X. Lin "Loop-Free Hybrid Single path/Flooding Routing Algorithms with Guaranteed Delivery for Wireless Networks", IEEE Trans. Parallel and Distributed Systems, 2001.