A communication framework for ubiquitous systems *

Marcio E. F. Maia[1,2,3], Saulo Aguiar[3], Bruno Gois Mateus[3], Carlos Queiroz[3], Reinaldo B. Braga[3], Rute Nogueira[3], Fredrik Toorn[4], Rossana M. C. Andrade[2,3]

[1]Campus de Quixada – Universidade Federal do Ceara (UFC)
Estrada do Cedro - km 05. CEP 63900-000 - Quixadá - CE – Brazil

[2]Departmento de Ciencia da Computacao, Universidade Federal do Ceara (UFC)
Campus do Pici - Bloco 910 - Fortaleza - CE – Brazil

[3]Grupo de Redes de Computadores, Engenharia de Software e Sistemas
Universidade Federal do Ceara – Fortaleza, CE – Brazil

[4]Sony Mobile

{marcio,brung,sauloaguiar,carlosqueiroz,reinaldo,rutenp,rossana}@great.ufc.br,

Fredrik.Toorn@sonymobile.com

Abstract. *The widespread use of mobile devices is reshaping the way users interact, permitting applications to explore physical proximity of devices to exchange information. Altough the number of communication technologies that can be used is vast (e.g. Wi-Fi, Wi-Fi Direct, LTE, UMTS, Bluetooth, NFC), builiding an application from ground up considering communication and application requirements is not a trivial task. In order to help developers to implement communication primitives, this paper presents a communication framework for ubiquitous systems. Two important contributions are related to our framework: i) a set of technology-independent interfaces to handle communication requirements; and ii) a modular and extensible layer-based architecture that permits developers to implement new communication technologies and message dissemination strategies. The outcome of using our communication framework is a simpler construction of distributed applications that explore physical proximity.*

1. Introduction

The use of mobile devices and the proliferation of Location Based Services (LBS) on the Internet are fostering a whole new interaction paradigm. Acquaintances exchange messages, multimedia content and information in a near real-time basis [Mascolo 2010], anywhere and anytime.

Services based on the user location have been widely adopted as the key characteristic to provide enriched information for mobile applications in ubiquitous environments [Braga et al. 2012]. For instance, proximity has been used to diffuse context information with nearby devices, a concept known as participatory sensing [Mascolo 2010, MAIA et al. 2013]. Another possibility is to explore user mobility and proximity to carry-and-forward short messages, which is known as opportunistic computing [Conti et al. 2010].

*This work is funded by MCT/Informatics Law under project grant number 2951

In these scenarios, the interacting mobile devices may be distributed and pervaded throughout the environment. Therefore, communication is performed using direct communication technologies such as Bluetooth[1] and Wi-Fi Direct[2], allowing the communication without an existing infrastructure.

Although these technologies and LBS are useful to create ubiquitous applications, the design of a modular and extensible communication infrastructure is a challenging task. Nowadays, developers must consider basic issues such as neighbor discover and selection, message exchange, device disconnection and routing to develop ubiquitous applications [Drabkin et al. 2011]. However, to facilitate this development, this paper presents a communication framework that aids developers to create applications through the use of direct communication. The framework is divided into layers. The lower layers are related to the link discovering and communication technology (e.g., link and physical layers). These layers handle the creation and destruction of links between two devices, and offers to developers primitives such as neighbor discovery, message exchange and device disconnection. Hence, the framework is able to handle communication among different technologies, such as Bluetooth, Wi-Fi, and Wi-Fi Direct.

The independence of using different communication technologies is possible due to the network layer, which reuses primitives from the lower layers to send, forward and receive messages. This layer is conceived to allow the implementation of new routing algorithms. Reusing the primitives offered by the lower layers, the upper layers offer to applications communication primitives.

The remainder of this paper is divided as follows. Section 2 discusses important issues to develop ubiquitous applications. Section 3 presents the communication framework. Section 4 details how the framework can be used and extended. Section 5 presents related work and section 6 lists some conclusions.

2. Exploring device proximity

According to the authors of [Maia et al. 2009], communication among nearby devices is at the core of ubiquitous computing. Direct communication allows devices to exchange messages when a fixed infrastructure is unavailable. Taking into account the information about physical proximity, other devices can be used to carry and forward messages when direct exchange is not possible. Along this line, developers use technologies such as Bluetooth or Wi-Fi Direct to implement the communication processes of their applications.

Approaches that rely on direct communication like mobile ad-hoc networks (MANET) [Sarkar and Lol 2010], delay-tolerant networks (DTN) [Whitbeck and Conan 2010] and opportunistic computing (OC) [Conti et al. 2010] have been subject of study in the last decade. They allow the communication without a pre-defined infrastructure and can be efficiently implemented in ubiquitous environments. Besides that, other characteristics should be considered in ubiquitous environments, such as: no assumption about the topology of nodes should be made; nodes have to forward messages to other nodes in a collaborative way; and nodes may be mobile.

Although MANET, DTN and OC have been extensively studied, they are not thor-

[1] www.bluetooth.com

[2] http://www.wi-fi.org/discover-and-learn/wi-fi-direct

oughly explored in real applications. In order to diffuse their use, general purpose solutions to handle direct communication could provide valuable contributions.

3. Communication framework for ubiquitous computing

In order to foster the creation of ubiquitous applications relying on direct communication, this paper presents a ubiquitous communication framework. Its main contribution is two-fold: 1) it offers a set of technology-independent interfaces to handle communication requirements such as neighbor discovery, connection and disconnection, and message delivery; 2) it is based on a modular and extensible, layer-based, architecture that permits developers to plug in new communication technologies and message dissemination strategies, reusing much of the existing functionalities.

Our communication framework is based on the conceptual architecture introduced in [Conti et al. 2010]. We then bring these concepts to apply in real ubiquitous scenarios, facilitating the construction of applications that explore physical proximity. Up to this point, the framework has implemented communication using Bluetooth and Wi-Fi, and two well-known routing algorithms that can handle multi-hop, namely AODV and Flooding.

Implemented using Android[3], messages exchanged between devices are described using JSON (Java Script Object Notation) [4], a lightweight data exchange format to foster interoperability between interacting devices. Binaries and a series of documents describing its use are available at the following link: `http://r2d2.great.ufc.br/dokuwiki/doku.php?id=bluetoothnetwork:bluetooth_network`.

The framework is divided into three layers, trying to improve modularity and extensibility. Therefore, any layer can be adapted or modified acording to implementation requirements. The bottom layer handles the functionalities specific to the communication technologies. The middle layer uses the primitives from the bottom layer and implements routing functionalities. The top layer reuses the functionalities from the two previous layers, and lets applications send messages. Figure 1 shows the architecture of the framework, detailed as follows.

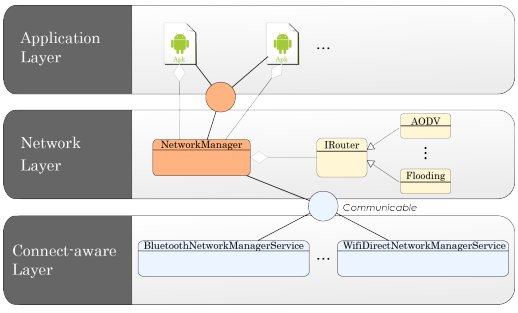

Figure 1. Architecture of the proposed framework

3.1. Connect-aware layer

This layer is responsible for handling the direct communication between two devices. The main features implemented in this layer are 1) Connectivity management, 2) Neighbor

[3]http://www.android.com/

[4]www.json.org

discovery and connection, 3) Message exchange and 4) Neighbor disconnection. These features are exposed to the upper layers as interfaces.

Since devices are mobile, they may enter and leave on the network unpredictably. The goal of the connectivity management is to keep the network connected, trying to maintain a path between any two devices.

Bluetooth. This technology has particularities that needed to be addressed. One is the restriction on the number of devices connected to each device. This restriction affects directly the connectivity management, since one device should consider carefully before connecting to a given neighbor, or it may run out of available connections. This problem has been analyzed before [Sharafeddine et al. 2012] and is called scatternet formation.

To minimize the load on each device, connectivity management is implemented using a simple scatternet formation strategy. Due to the restriction on the number of connections, this strategy instead of connecting to every known neighbor, and quickly saturating the number of connections, it uses the concept of redundant link.

A link is redudant when it can be removed and the network still remains fully connected. To define if a link to a new neighbor is redundant, once two devices meet, called device A and device B, they exchange their list of neighbors. If the list received by device A from device B contains at least one device already present in the list of neighbors from device A, the link between devices A and B is defined as a redundant link.

Before connecting to a device, the redundant link algorithm is used and the number of free connections that a device has is considered. If it has less than five connections, it will just connect to the new device. If it has five or more connections, it will check if there is any redundant link. If there is a redundant link, this link is removed and the new connection is added. If the number of connections is equal to seven, the new connection cannot be created. The number of five connections was defined analyzing the geometry created by devices communicating wirelessly.

Once a connected neighbor disconnects, one of two possible actions is taken. If the link was a redundant one, the redundant link list is updated to consider this disconnection. If the link was not a redundant one, the neighborhood is scanned looking for devices that could replace this neighbor that left.

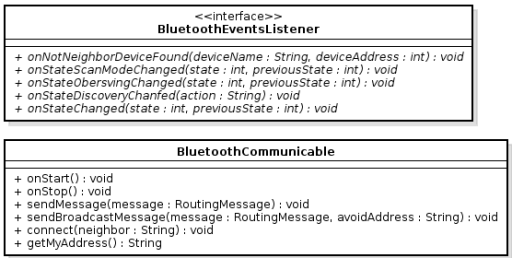

Figure 2. Interfaces exposed by the Bluetooth communication module

Figure 2 shows the interfaces that are exposed by the lower layer. There are two sets of functionalities. One set handles events occurring in the Bluetooth module, such as neighbor found or changes on the observing state. The second acts upon communication technology modules, responsible for device connection, disconnection, sending messages,

starting and stopping the module.

Using Wi-Fi and Wi-Fi Direct, this restriction on the number of connections can be relaxed. The framework already has an implementation using regurlar Wi-Fi. Wi-Fi Direct has details that deserve more attention and shall be considered in the future.

Wi-Fi. Unlike Bluetooth, that employs direct communication between neighbor devices, Wi-Fi permits that any two devices connected to an access point in the same subnetwork to communicate. The goal is to provide to the upper layers the same interfaces present in the Bluetooth module, but abstracting away what is specific from Wi-Fi. The Wi-Fi implementation relies on a network port monitor, where all events are received using that port (e.g. device connection and disconnection, message received). When a device needs to send a message, it wraps the message up on an UDP datagram and sends it to its network broadcast address, using the predefined network port. Other devices in the network receive the broadcasted message in the same predefined port, unwraps it pushs it to the upper layers throught network listeners and specific interfaces.

3.2. Network layer

Using the interfaces from the contact-aware layer, the network layer is responsible for routing messages. One important thing that must be highlighted is that the contact-aware layer only send 1-hop messages to nearby devices. The network layer routes messages using all technologies available from the contact-aware layer. The framework provides implementations for two routing algorithms, namely AODV and Flooding.

The network layer permits communication among devices that are not within communication range. Since no suposition about a centralized routing service should be made, messages are sent using multi-hop [Sarkar and Lol 2010, Conti et al. 2010]. Devices forming the network are responsible for forwarding messages.

Independently of the routing mechanism, each hop (device) decides how to send a message to the next hop based on the metric defined by the routing protocol. Thus, all a routing algorithm requires from the communication layer is the list of active neighbors, how to send a message using unicast or broadcast, and how to receive a message.

These are precisely the functionalities offered by the lower layer. Reusing them, the framework allows different routing algorithms to be implemented on top of the communication layer. Figure 3 shows the core interfaces and design of the network layer.

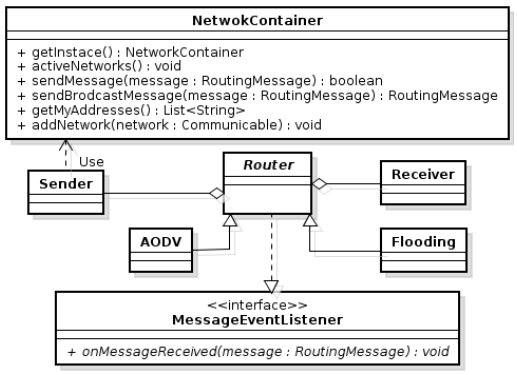

Figure 3. Interfaces exposed by the network Layer to the application layer

Messages coming from the contact-aware layer are delivered to the *Router* class by the *MessageEventListener* interface, using the *onMessageReceived(Message)* method.

The *Router* class implements this interface and it is the actual routing algorithm. This is the base class for implementing a new routing strategy. This framework offers to developers two implementations of well-known routing algorithms: Flooding [Viswanath and Obraczka 2006] and AODV [Perkins and Royer 1997].

If the routing algorithm decides to forward a message to a neighbor device, it queues the message using the *Sender* class by calling the *queueDataMessage(Message)* method, informing the next hop. Since the framework permits to route messages using different technologies, when the *NetworkContainer* class receives a message to be sent, it first decides, based on the destination of the message informed by the router, which communication technology should be used. The class *NetworkContainer* then selects the communication technology and sends the message using the *sendMessage* method.

To implement a different routing algorithm, the developer would only have to implement the *Router* class. All the functionalities regarding neighbor discovery and selection as well as message delivery process in the contact-aware layer are offered by the framework and can be reused. This approach reduces considerably the complexity to develop a new routing algorithm.

3.3. Application layer

One instance of the communication framework is deployed per device. To send a message, an application uses the interface provided by the framework, listed in figure 4, in the *NetworkManager* class. Additionally, when a message addressed to an application arrives, the listener class *NetworkEventListener* notifies the application. The application may be notified of other events, such as device found or device disconnected, as well.

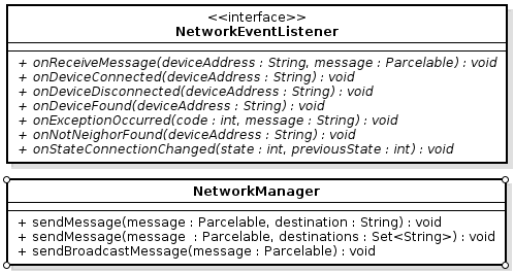

Figure 4. Interfaces exposed by the network layer to the application layer

The framework has two main contributions: 1) It provides primitives so developers can create novel solutions to improve the communication among nearby devices, without relying on a physical infrastructure; and 2) It offers a communication infrastructure implementing two existing communication technologies (e.g., Bluetooth and Wi-Fi) and two routing algorithms (e.g., Flooding and AODV).

The two first layers (contact-aware and network) present extension points which developers can explore. The contact-aware layer can be extended by adding new communication technologies, such as Wi-Fi Direct or 3G, or can be changed to offer different implementations of Bluetooth and Wi-Fi. The network layer can be extended by adding any existing routing approaches or by developing new solutions.

A different communication module can be implemented by extending the *Communicable* interface, similarly to what was done by the *BluetoothCommunicable* class. A similar class called *EventsListener* should be created and made available in the framework to the application developer. Please refer to figure 4 for more details. Finally, the new communication technology should handle device discovery and connection, unicast and broadcast message delivery, device disconnection, along with any other functionality offered by the *EventsListener* interface, according to the publish/subscribe pattern. Any new communication technology has to subscribe in the *NetworkContainer* class, then messages are routed to devices using this new technology.

As mentioned before, new routing algorithms may be implemented as well. Since these algorithms should decide when and to whom messages should be forwarded to, the complexity lies on the algorithms themselves. From the perspective of the framework, any routing algorithm basically sends and receives messages using the *Sender* and *Receiver* classes. All the complexity regarding the technology itself is abstracted away from the routing algorithm developer.

The framework is intended to be used by developers of android applications that require direct communication among nearby users. To use the communication functionalities the application developer should create an instance of the *NetworkManager* class, passing the *android.context.Context* class from Android, along with an implementation of the *NetworkEventListener* callback interface, to be notified when a network event occurs. This code should be put in the *onCreate()* method of the activity.

The next step is to invoke the *onResume()* method on the *NetworkManager* instance. It starts all communication technologies and the network layer. With the implementation of the *NetworkEventListener* interface, the application is notified of events occurring on the network, such as device connection or message received. It is up to the application developer to decide what should be done when these events occur. Finally, an application may send messages by simply calling *sendMessage(Message, destinationAddress)* or *sendBroadcastMessage(Message)* methods.

4. Proof of concept

To validate the concepts proposed in this paper, a proof of concept was developed putting together the communication framework and the framework to interact with existing OSN [Maia et al. 2012]. The idea is to share opinions between different communication technologies, post them in the social networks and retrieve the average opinion of the users.

Figure 5. Sharing a tag with nearby devices

Figure 5 shows three screenshots of the application. The first phone, on the right hand side, shows user A rating a McDonald's restaurant he had just eaten in. McDonald's is selected from a list of places accessed from Facebook. After selecting it, the user chooses the properties that are going to be rated and informs the value referring to each property.

The next step is to share that opinion. It may be posted on Facebook or shared with devices nearby. Once the user chooses to share with nearby devices and sends the opinion content, user B receives a notification on his phone (figure 5, middle phone) letting him know that a new opinion is available. Clicking on the notification icon, opinion is showed.

Figure 6 shows a code snippet extracted from the application. It is divided in three parts: framework initiation, message broadcast and message receive. First, every communicating device must initiate the framework. They do so by executing the *onResume()* method from the *NetworkManager* (lines 2-5). Once the framework is initiated, device A broacasts a message to all known devices, using any available technology (lines 7-10, on the left). Before the message is sent, its content is parsed to JSON. Notice that activities related to each technology (Wi-FI, Bluetooth), neighbor selection in each technology and routing is performed by the layers implemented in the framework. Finally, for an application to be notified of an incoming message, it must implement the *onReceiveMessage()* method from the *NetworkManagerClass* (lines 9-11, on the right).

```
1
2  //Both devices execute initiation activities
3  mMessageListener = new MessageListenerImpl( mContext );
4  mNetworkManager = new NetworkManager( mContext, mMessageListener );
5  mNetworkManager.onResume();
6
7      //Device A sending a broadcast messsage
8      message.setMessageType(ConceptMessage.SHARING);
9      JSONObject object = putConceptsIntoJSON();
10     message.setRatings( object.toString() );
11
12     mNetworkManager.sendBroadcastMessage( message );
13
```

```
//Device B receiving a network message
public void onReceiveMessage(String deviceName, String deviceAddress, Parcelable message) {
    ConceptMessage received = (ConceptMessage) message;
    createNotification(received);
}
```

Figure 6. Framework initiation, message broadcast and receive

Notice how the application only uses methods exposed by the framework to handle communication activities. Any functionality regarding communication technology and routing is handled exclusively by the framework. This modular approach hides away the comlexities related to device communication.

5. Related Work

Flock [Boix et al. 2011] is an abstraction to represent user groups in mobile social scenarios, offering to developers primities to define groups at runtime, fostering interaction among nearby users. The difference is that they do not give much attention to communication mechanisms. In a way, our communication framework could be used by them to create the interaction infrastructure and group communication.

AmbientTalk [Suzuki et al. 2011] is a object-oriented programming language aimmed at peer-to-peer mobile applications. Focusing on the application layer, it offers distributed programming primitives such as events, services and concurrency. Interaction between devices is accomplished based on massage-passing. AmbientTalk and our communication framework could be placed together to offer full-fledged coordination stack for peer-to-peer mobile applications.

Mobiclique [Pietiläinen et al. 2009] is a mobile social networking middleware that exploits ad hoc social networks to disseminate content, using on opportunistic connections between neighboring devices. They offer an environment to deploy and test the performance of opportunistic forwarding algorithms which leverage the social profiles and networks of users, as it monitors at the same time their mobility and social behavior. Similarly, our framework permit interactions between nearby users, however, focusing on reusability and extensibility of layers and modules present in the communication infrastructure.

6. Conclusions and Future Work

This paper presented a communication framework to facilitate the development of direct communication among devices, as well as to permit new communication technologies and routing algorithms to be added. The framework was designed aiming at two basic goals: 1) to be independent of any communication technology and routing algorithm; 2) to be as modular and extensible as possible. These goals were accomplished by separating the functionalities into layers and modules, communicating using publish/subscribe mechanisms.

The framework was divided in three layers. The lower layer comprises the different communication technologies. It has implementations to Bluetooth and Wi-Fi technologies. Reusing these modules, the network layer is responsible for delivering messages using multi-hop. To do so, it has implemented Flooding and AODV, offering to application developers two routing algorithms. Finally, on the upper layer, application developers may use the interfaces exposed by the lower layers to send and receive messages among nearby devices.

As a future work, a detailed performance analysis should be carried out. The ideia is to analyze how the framework behaves as parameters such as the number of messages, number of devices and mobility affects metrics like the percentage of messages delivered, delivery latency and round trip time.

References

[Altundag and Gokturk 2006] Altundag, S. and Gokturk, M. (2006). A practical approach to scatternet formation and routing on bluetooth. In *Computer Networks, 2006 International Symposium on*, pages 23 –29.

[Boix et al. 2011] Boix, E. G., Carreton, A. L., Scholliers, C., Van Cutsem, T., De Meuter, W., and D'Hondt, T. (2011). Flocks: enabling dynamic group interactions in mobile social networking applications. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, pages 425–432, New York, NY, USA. ACM.

[Braga et al. 2012] Braga, R. B., Tahir, A., Bertolotto, M., and Martin, H. (2012). Clustering user trajectories to find patterns for social interaction applications. In *Proceedings of the 11th international conference on Web and Wireless Geographical Information Systems*, W2GIS'12, pages 82–97, Berlin, Heidelberg. Springer-Verlag.

[Conti et al. 2010] Conti, M., Giordano, S., May, M., and Passarella, A. (2010). From opportunistic networks to opportunistic computing. *Communication Magazine*, 48(9):126–139.

[Drabkin et al. 2011] Drabkin, V., Friedman, R., Kliot, G., and Segal, M. (2011). On reliable dissemination in wireless ad hoc networks. *Dependable and Secure Computing, IEEE Transactions on*, 8(6):866 –882.

[Maia et al. 2009] Maia, M. E., Rocha, L. S., and Andrade, R. M. (2009). Requirements and challenges for building service-oriented pervasive middleware. In *Proceedings of the 2009 international conference on Pervasive services*, ICPS '09, pages 93–102, New York, NY, USA. ACM.

[Maia et al. 2012] Maia, M. E. F., Filho, J. B. F., de Q. Filho, C. A. B., Castro, R. N. S., Andrade, R. M. C., and Toorn, F. (2012). Framework for building intelligent mobile social applications. In *SAC '12: Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 525–530, New York, NY, USA. ACM.

[MAIA et al. 2013] Maia, M. E. F., Fonteles, A. S., Gadelha, R., Almeida Neto, B. J., Viana, W., and Andrade, R. M. C. (2013). Loccam - loosely coupled context acquisition middleware. In *Proceddings of the 28th Symposium on Applied Computing*, SAC'13, pages 82–97, New York, NY, USA. ACM.

[Mascolo 2010] Mascolo, C. (2010). The power of mobile computing in a social era. *Internet Computing, IEEE*, 14(6):76 –79.

[Perkins and Royer 1997] Perkins, C. E. and Royer, E. M. (1997). Ad-hoc on-demand distance vector routing. In *In Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100.

[Pietiläinen et al. 2009] Pietiläinen, A.-K., Oliver, E., LeBrun, J., Varghese, G., and Diot, C. (2009). Mobiclique: middleware for mobile social networking. In *Proceedings of the 2nd ACM workshop on Online social networks*, WOSN '09, pages 49–54, New York, NY, USA. ACM.

[Sarkar and Lol 2010] Sarkar, N. I. and Lol, W. G. (2010). A study of manet routing protocols: Joint node density, packet length and mobility. In *Computers and Communications (ISCC), 2010 IEEE Symposium on*, pages 515 –520.

[Sharafeddine et al. 2012] Sharafeddine, S., Al-Kassem, I., and Dawy, Z. (2012). A scatternet formation algorithm for bluetooth networks with a non-uniform distribution of devices. *J. Netw. Comput. Appl.*, 35(2):644–656.

[Suzuki et al. 2011] Suzuki, T., Pinte, K., Van Cutsem, T., De Meuter, W., and Yonezawa, A. (2011). Programming language support for routing in pervasive networks. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 226–232.

[Viswanath and Obraczka 2006] Viswanath, K. and Obraczka, K. (2006). Modeling the performance of flooding in wireless multi-hop ad hoc networks. *Comput. Commun.*, 29(8):949–956.

[Whitbeck and Conan 2010] Whitbeck, J. and Conan, V. (2010). Hymad: Hybrid dtn-manet routing for dense and highly dynamic wireless networks. *Comput. Commun.*, 33(13):1483–1492.