AGORA: an integrated approach for collaboration in MANETs

Marcel Arrufat, Gerard París, Pedro García López
Universitat Rovira i Virgili
Av. dels Països Catalans, 26
43007 Tarragona, Spain
{marcel.arrufat,gerard.paris,pedro.garcia}@urv.cat

ABSTRACT
Nowadays, a growing interest exists for collaborative working environments. Proliferation of hand-held devices with networking capabilities, together with the free-of-charge characteristic have turned MANETs in a good alternative for group spontaneous collaboration. In order to reduce application development difficulty and adjust to MANET specific requirements, middleware approaches seem a good alternative for developing new collaboration applications. However, it is not easy to find a system which allows rapid application deployment. Building applications from scratch makes developing applications for ad hoc scenarios a difficult task. To cope with these requirements, we introduce AGORA, an integrated approach for collaborative applications in the MANET environment. AGORA provides an architecture with three well defined elements: a plug-in framework which simplifies the development of MANET collaborative applications; a collaboration middleware which offers communication and group services to application plug-ins, and finally, a routing layer in charge of providing efficient group communication.

Categories and Subject Descriptors
C.2.4 [**Computer-Communication networks**]: Distributed Systems—*Distributed Applications;* D.2.11 [**Software Engineering**]: Software Architectures —*Domain specificArchitectures*

General Terms
Algorithms, Performance, Design.

Keywords
MANET, middleware, integration, group collaboration

MOBILWARE 2008, February 13-15, Innsbruck, Austria

DOI 10.4108/ICST.MOBILWARE2008.2883

1. INTRODUCTION
Developing applications specially targeted for MANETs is not a trivial task. Devices' limited resources together with dynamic and multihop network present a serious challenge which applications must face. In these terms, it seems reasonable that middleware for ad hoc networks will highly help in reducing the complexity of MANET application development. This middleware approaches provide high level services which can be used by applications in order to construct more complex and flexible applications.

It is known that, due to MANET characteristics, there is not a unique middleware solution that copes with all needed requirements. Several and different challenges arise when facing these requirements [6]. In first place, efficient use of resources, such as memory, bandwidth and computational power, is needed. System scalability becomes crucial when a great amount of members join the network and try to intercommunicate. Other issues like quality of service, devices' heterogeneity and security may also be considered when creating a middleware for ad hoc networks. Therefore, it seems that requirements must be considered depending on the selected scenario for ad hoc networks.

During the last few years the reduction of the cost of portable devices has implied a growing utilization of mobile phones, handheld game consoles and pocket computers. In consequence, a new range of opportunities arise for collaborative working environments. However, bringing the features of collaborative systems to the mobile ad-hoc (MANET) scenario is not trivial. Although flexibility and low cost establishment make these networks attractive for spontaneous collaboration, several management and communication problems emerge when traditional collaboration systems are moved towards the MANET environment. Topology awareness, node dynamicity and scarce resources must be considered in order to build good performing communication primitives.

Communication functionalities stand as one of the most important constraints since one-to-one and group communication are the key which collaborative applications rely on. In this environment, chat rooms, file sharing and e-mail messaging are frequently used, so synchronous and asynchronous message delivery is necessary. Since using TCP as transport protocol seems to be ineffective in MANETs [8], lighter approaches using UDP must be considered. Furthermore, these applications may need a reliable communication channel in order to ensure full packet delivery. Besides, although it is usually not mentioned, an ordered channel is also necessary for delivering packets for most

applications. Although these assumptions may be too costly for large groups, AGORA's scope is restricted to small, medium-sized groups, so providing these needs is feasible. On the other hand, group membership and management are also necessary in developing group-aware collaborative application. Notification of online and joining/leaving members is useful for most collaboration systems.

Regarding issues other than collaboration requirements, middleware approaches should be easy to deploy, extend and use. Usability and ease of application development turn middleware in a powerful tool to develop end-user applications. Most current approaches just offer a set of functionalities that can be used to build these applications: publish a message under a certain topic, share a file or retrieve information from a given pattern. These functionalities are in fact useful but may not be sufficient in order to develop more complex MANET applications in a fast and straightforward manner.

Taking all these considerations into account, we present AGORA, an integrate solution for MANETs that brings together:

- A collaboration middleware: provided with a full set o communication mechanisms and membership information.
- A plug-in framework: benefits from middleware in order to build final applications in a rapid and simple manner.
- An application level multicast (ALM) protocol: designed to enhance communication performance, which avoids using a specific MANET multicast.

With these three components, AGORA offers a new ready-to-deploy solution for developing collaborative applications for MANETs in an easy and straightforward way. Since it is not restricted to offering just a single service like file-sharing or probabilistic multicast, developers can use a full set of functionalities in plug-ins in order to adjust to the needs of each application.

As we will see hereafter, the collaboration middleware provides synchronous and reliable communication: communication channels, naming and publish/subscribe services. Besides, the topology-aware multicast protocol takes care of minimizing global communication, whereas the plug-in framework is in charge of reducing application development complexity.

2. RELATED WORK

As pointed out by [6], it is difficult to establish an obvious taxonomy of available middleware solutions due to the tightly coupling of middleware with specific applications. However, a classification is proposed based on the programming model. In this section we will review several middleware approaches for MANETs that share common features and tackle similar challenges than our proposal. Following the mentioned classification, the reviewed solutions could be classified as Peer to Peer Based Middleware (JMobiPeer, Peer2Me) and Event Based and Message Oriented Middleware (STEAM, EMMA, AGAPE).

Event Based Middleware is used to support distributed applications that must react to changes in the environment, so this is very suitable for MANETs because of their lack of infrastructure. Likewise, Message Oriented Middleware provides asynchronous communication paradigms like publish/subscribe which are particularly adequate for pervasive environments. Among this classification we found STEAM [1], an event-based middleware that eliminates dedicated event servers and uses the implicit publish/subscribe model instead. Consumers subscribe to certain event types and publishers are able to publish particular events. Moreover, STEAM allows different filters to be applied to the published events. Content filters are used to define complex subscriptions at the subscriber's side. Likewise, proximity filters can be defined in the publisher's side in order to restrict the propagation of such events.

In EMMA [3], a well known standard from traditional distributed systems is adapted to cope with MANET requirements. EMMA is an implementation of the Java Message Service (JMS) that incorporates an epidemic routing mechanism to facilitate message delivery. This middleware provides point-to-point communication as well as publish-subscribe mechanisms. However, it must be taken into account that the epidemic routing protocol does not guarantee the reliability in message delivery.

AGAPE [2] is presented as Another Message Oriented Middleware specially designed for MANETs. This collaborative middleware provides group membership and message-oriented communication in pervasive environments. It also offers context information of co-located group members, such as their attributes and characteristics. AGAPE organizes members in locality based clusters and considers two different roles depending on the device features: the cluster head and the managed entities. Low-resource devices like mobile phones or PDAs act as managed entities and rely on a more powerful device like a laptop that would be the cluster head. Whereas this static role differentiation could be useful for team operations like emergency scenarios, we believe that is not specially suited for scenarios where collaboration between members is highly decentralized.

The next two reviewed solutions belong to the category of Peer to Peer Based Middleware. This kind of middleware utilizes a P2P communication model that involves resource and information sharing in order to perform a common task. P2P architectures share many similarities with MANET environments [7]: decentralization, dynamicity, and self-adjusting behavior. Hence, an association of both systems is believed to benefit the global operation of a collaborative application. However, most P2P systems were designed for wired and fixed infrastructures so adaptations are needed to use a P2P architecture in a infrastructure-less environment. Among the attempts to adapt P2P systems to mobile ad hoc networks we describe hereafter two middleware approaches that claim to offer an application framework: JMobiPeer and Peer2ME. The first one, JMobiPeer [4], is a JXTA compatible framework designed for J2ME CLDC environments. JXTA is the most mature P2P framework and provides interoperability and platform independence, allowing connection between heterogeneous devices. Hence, JMobiPeer benefits from these characteristics and introduces new features like a routing layer, emulation of multicast functionality to adapt JXTA to mobile environments, and the concept of code mobility. Nevertheless, JXTA based applications may introduce high communication overhead because the architecture does not take

into account locality of nodes and relies on the exchange of XML messages.

Finally, Peer2Me [5] is an application framework for mobile peer-to-peer applications which facilitates the development of this kind of applications and offers node discovery and messaging services. Several collaborative applications are provided together with the framework. It must be noted that Peer2Me is designed to be deployed on mobile phones under minimal J2ME configuration using Bluetooth devices. These last two frameworks only consider hand-held devices and therefore are not suitable for laptops or notebooks, where more complex applications could be deployed.

After evaluating these approaches, we can conclude that, to our knowledge, currently there is not an integrated solution which provides such a rich set of functionalities, together with the possibility of developing applications for the MANET environment in a fast and simple manner. Moreover, the inclusion of the application level multicast inside the middleware seems to be a novel approach that will abstract developers from the necessity of providing a conventional network-layer multicast protocol.

3. AGORA architecture

AGORA architecture can be defined in three main components. In first place, the plug-in framework allows fast development of applications as plug-ins. These plug-ins can be packaged, exchanged between peers and even be installed on the run. This is achieved thanks to the plug-in manager, which also verifies integrity and controls plug-in lifecycle.

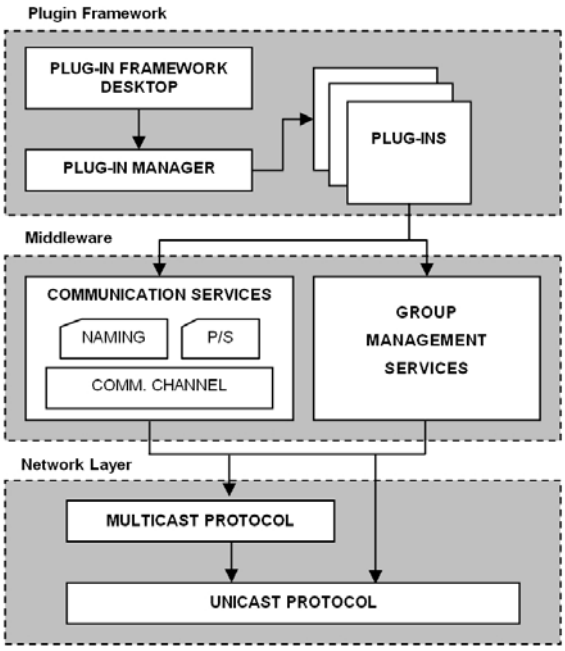

Figure 1 AGORA Architecture

Secondly, the collaboration middleware provides different communication mechanisms as well as group management primitives. Membership information, named communication channels and different communication paradigms are available for plug-ins. Finally, the application level multicast protocol is in charge of routing all multicast messages through the network. Considering topology and benefiting from broadcast nature of the medium, this protocol aims to reduce global traffic and minimize end-to-end delay.

3.1 Plug-in Framework

The plug-in framework is the highest layer of the architecture. End users can use the application framework by using existing plug-ins and extend it by implementing new ones. As depicted in Figure 1, we can distinguish three different components: the plug-in desktop, the plug-in manager and the plug-ins.

Plug-ins are the final applications that benefit from middleware layer. Thus, they have access to a set of functionalities provided by the communication and group management services. Plug-ins must comply with a given interface in order to be loadable by the plug-in manager. In order to maintain plug-ins light weighted, they consist in a compressed file which contains all plug-in classes and a short XML description file. Other resources as images can also be attached to the plug-in by adding them to the compressed file.

The plug-in manager is in charge of loading, registering and starting/stopping plug-ins. Plug-ins can be loaded from the local device or, what is more important in a collaboration scenario, can be retrieved from another peer. In order to execute this process, the plug-in manager utilizes naming services information for locating downloadable plug-ins. This is achieved by registering plug-in information into the naming services when a plug-in is correctly loaded into the framework.

The plug-in framework desktop offers a graphic user interface which the user can interact with. A list of local plug-ins is available as well as a list of remote plug-ins owned by other peers. Several plug-ins can be started at the same time and can be hot-deployed when downloaded from a remote peer.

3.2 Middleware

Middleware is responsible for offering a set of well defined services for plug-in development. These services are divided into group management and communication services. Both services benefit from routing layer in order to perform group communication in an efficient way.

3.2.1 Group management services

Group management services handle all interactions related to group creation/deletion and group membership. This module allows creating logical groups where members can collaborate. It is necessary that all members willing to collaborate belong to the same group. Initially, all members belong to a default group and afterwards they can create or join new groups. Group information is stored in the lightweight replicated naming service, so that it is available for all members in the network. Regarding group membership, this module also provides information about all connected members in the group. When members leave or join a group, events are triggered and forwarded to the plug-ins that should previously register to these events.

3.2.2 Communication services

Communication services represent the set of services available for sending messages to other peers. The main module is the communication channel, which allows peers to send messages to one or to all the peers in the group. Named communication

channels are available, so one plug-in does not have to deal with filtering messages from other plug-ins. Messages targeted to all group members are sent through application level multicast, instead of using a multiple unicast approach. Furthermore, the channel provides mechanisms for synchronous and asynchronous receive. On top of this channel, we have built two abstractions which may help in plug-in development: a publish/subscribe and a naming service.

The publish/subscribe service is implemented as a subset of the JMS (Java Message Service) interface. Since it follows JMS specification, its behavior is practically identical to the defined in the standard. The service offers a topic-based publish/subscribe model with persistent and non-persistent subscriptions. It must be considered that since there is no central JMS server, subscription information is partially kept by each peer in the group. In this way, when a node rejoins a group due to a temporary disconnection, it asks one of the members of the group in order to obtain previous messages published in the topic.

The naming service also follows a JAVA standard. In this case, it implements a subset of the JNDI (Java Naming Directory Interface) interface. This naming service is intended to be used to store lightweight data like resource discovery, group and other information. The mechanism to store this information is simple but effective: when changes are performed in the naming service, these changes are sent via multicast to all group members. Since we know this might be a resource-expensive mechanism of delivering data, naming services are supposed to be used to store minimal but critical data. On the other hand, information is rapidly and totally available for current group members.

3.3 Network layer

The network layer provides a message delivery mechanism to be used by the communication channel. Since AGORA is designed for collaboration scenarios, routing mechanisms must also consider its specific requirements and mobility model.

Collaboration scenarios like a meeting or a scientific conference are characterized by having a moderate number of nodes located in a small area. Normally, they are located in a room or in a large hall, where some of them may remain static for long periods of time. Some nodes may change its location from time to time in order to interact with other existing groups. However, most communication is performed in well-defined areas, where nodes are located at most two or three hops from the most distant hop.

Furthermore, it seems clear that multicast communication is the best option when applying to group communication. Reuse of paths to deliver data reduces bandwidth use and node overhead. Therefore, we implemented OMCAST, an application level multicast specially designed for collaboration scenarios. Its main features are:

- Broadcasting of data packets to 1-hop neighbors to reduce communication overhead.
- Decentralized membership information available for higher level layers.

By using OMCAST we reduced global traffic thanks to the local broadcast delivery technique. The main idea of this process is to send just once as a broadcast message if there are enough neighbors located at one physical hop willing to receive the message. However, as almost all application multicast approaches, OMCAST does not guarantee reliable message delivery neither message ordering.

3.3.1 Ordering and reliability

The network layer also provides means to send unicast and multicast messages through the network. Since the use of TCP connections is discouraged, we have focused on ways of bringing ordering and reliability to the UDP protocol. These two features will be needed for collaboration at the middleware layer.

In order to provide reliability and ordering in a transparent and easy way, we took the toolkit JGroups as the foundation of our network layer implementation. JGroups, based on Java technology, offers reliable group communication by providing a flexible protocol stack, where each protocol provides a specific functionality, like fragmentation handling, lost message retransmission, ordering, membership and encryption.

Reliability of multicast communication remains as its key characteristic, which is a deployment issue and does not have to be implemented by the developer. Furthermore, new protocols can be easily added to the protocol stack for extending system capabilities. JGroups follows the chain of responsibility pattern, where each protocol only handles its own messages, and forwards them to the next protocol.

In AGORA we have used the reliability and ordering protocols provided by JGroups. Since we are not using network layer multicast, we had to replace it by OMCAST, our application layer multicast, as it is depicted in Figure 2. UNICAST protocol is in charge of FIFO ordering and reliability for unicast packets whereas NACKACK behaves in a similar way for multicast packets. UDP allows sending unicast and multicast messages in the original JGroups structure. As previously mentioned, in AGORA multicast packets are sent via OMCAST.

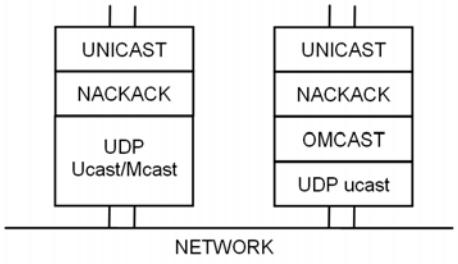

Figure 2 Protocol stack change

In this way, we ensure that all application messages will be delivered to the receivers. Furthermore, configurations parameters such as number of retransmission, retransmission timeout and other can be easily configured and adapted to obtain a more appropriate behavior.

4. Application implementation

For testing AGORA collaboration features, we have implemented two collaboration plug-ins: a chat plug-in and a photo-sharing plug-in. The chat plug-in uses group communication to send a message to all group members. Private messages, that is, one-to-one messaging, are also possible. The chat makes available membership information, so online users can be identified in a side panel. The photo-sharing plug-in uses the naming service to

store location of the shared photographs. Each member can download and visualize a photo by looking up the needed resource information in the naming service. On the one hand, the Chat plug-in shows that AGORA membership is accurate, reacting quickly to members joining or leaving, and unicast and multicast communication is reliable. On the other hand, photo-sharing plug-in allows sending big picture files to another member of the group through the MANET. Although multimedia transmission is not the main feature of AGORA, we believe that it must support medium-size file sharing. For instance, sending a 900 KB file between two hosts separated by a two-hop link took less than a second.

AGORA has been tested with both plug-ins under Linux (using OLSR and DYMO routing protocols) and Windows systems.

4.1 Performance evaluation

In order to test AGORA performance, we have recreated a typical collaboration scenario. As it is depicted in Figure 3, three rooms initially have been set up with different mobile nodes in each room. Nodes from room A and C can only communicate with nodes in room B. The two nodes located in room B can communicate directly with the rest of nodes. In this terms, when node A1 needs to send a message to C1, the message is forwarded by node B1 and finally delivered to C1. Thus, we create an initial two hop scenario, which may become three-hop when nodes start moving. Two nodes from Room A move to Room B and Room C during the test.

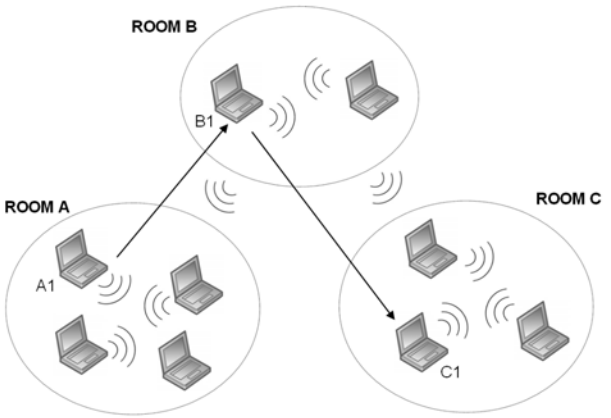

Figure 3 Real test scenario

For this particular test, all the laptops run Windows XP service pack 2 and Andreas Tønnesen's OLSR implementation.

The test consists in a plug-in which measures end-to-end delay of multicast packets. One of the nodes sends a REQ message to all members of the group. The rest of nodes, when receiving this REQ message, reply with an ACK unicast message to the source. ACK messages contain the timestamp value of the received REQ message. When the source collects all the ACK messages from the other members of the group, the mean round trip time value can be computed. 50 REQ messages are sent using multicast.

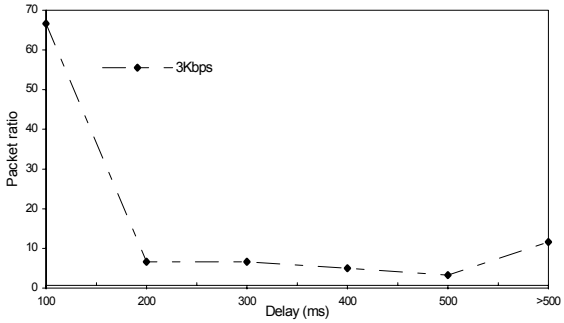

Figure 4 Multicast packet delay

In Figure 4 we can see the mean round trip time for all packets. In this figure, a measure of 100 ms. means that delay is between 0 and 100 ms; a measure of 200 indicates values between 100 and 200, etc. We can see that nearly a 70 per cent of packets are delivered in less than 100 ms. It must be considered that all we are using reliability and ordering mechanisms to ensure full packet delivery ratio, so packets are retransmitted and delivered in order if they are not received in first place. Therefore, we can conclude that AGORA performs more than acceptably in this real scenario.

4.2 Lightweight frameworkMobile devices portability

AGORA is capable also of being launched in framework for PDAs, as it is shown in Figure 5. The shared photo-album plug-in is displayed and the framework options are shown in the bottom.

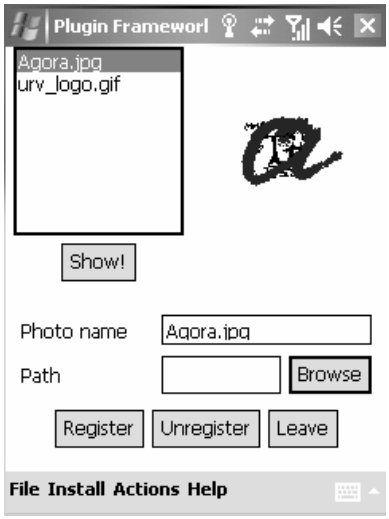

Figure 5 PDA Plug-in framework desktop

In this lightweight version, plug-ins can be installed and retrieved from other machines as well. Both chat and file-sharing plug-ins are available for the lightweight framework and have been tested in a one-hop environment. However, due to difficulties in finding OLSR implementations for PDAs, we have not been able to test performance in a multihop scenario.

5. Conclusion and ongoing work

In this paper we have presented AGORA, a complete approach for developing collaboration application in MANETs. Three layers in the architecture cope with the necessities: The plug-in framework eases application development whereas middleware offers a rich set of collaborative functionalities. The application level multicast is in charge of reducing group communication overhead. By deploying these three components together we offer a complete solution for rapid application development in MANETs.

Future work is focused on bringing PDAs to the multihop environment and improving multicast protocol efficiency by reducing control overhead and improving packet delivery ratio. We also plan to perform medium-scale real tests with 20-30 portable devices using AGORA plug-in framework.

6. ACKNOWLEDGMENTS

This work is funded by the Information Society Technologies programme of the European Commission, Collaborative Working Environments, under the FP6-2006-IST-034241 POPEYE project.

7. REFERENCES

[1] R. Meier and V. Cahill, "Exploiting Proximity in Event-Based Middleware for Collaborative Mobile Applications," First Workshop on Middleware for Network Eccentric and Mobile Applications (MiNEMA), Dublin, Ireland, 2004.

[2] D.Bottazzi, A.Corradi, R.Montanari, "AGAPE: A Location-Aware Group Membership Middleware for Pervasive Computing Environments", in Proc. of the 8th International Symposium on Computers and Communications (ISCC2003), IEEE Press, Turkey, July 2003.

[3] M. Musolesi, C. Mascolo, S. Hailes, "EMMA: Epidemic Messaging Middleware for Ad hoc networks". Journal of Personal and Ubiquitous Computing, Springer, Vol 10, No 1, February, 2006, p. 28-36.

[4] M. Bisignano, G. Di Modica, O. Tomarchio, "JMobiPeer: a middleware for mobile peer-to-peer computing in MANETs". First International Workshop on Mobility in Peer-to-Peer Systems (MPPS) (ICDCSW'05) pp. 785-791.

[5] Alf Inge Wang, Tommy Bjornsgard and Kim Saxlund, "Peer2Me - Rapid Application Framework for Mobile Peer-to-Peer Applications", International Symposium on Collaborative Technologies and Systems (CTS 2007), Orlando, Florida, USA, May 2007.

[6] S. Hadim, J. Al-Jaroodi, N. Mohamed, "Trends in Middleware for Mobile Ad Hoc Networks," invited paper in the Journal of Communications, Vol 1, No. 4, pp. 11-21, July 2006.

[7] Lu Yan, Kaisa Sere, Xinrong Zhou, Jun Pang, "Towards an Integrated Architecture for Peer-to-Peer and Ad Hoc Overlay Network Applications", In Proc. 10th Workshop on Future Trends in Distributed Computing Systems - FTDCS'04., pp. 312-318.

[8] X. Chen, H. Zhai, J. Wang, and Y. Fang, "TCP Performance over Mobile Ad Hoc Networks," Canadian Journal of Electrical and Computer Engineering (CJECE) (Special Issue on Advances in Wireless Communications and Networking), Vol. 29, No. 1/2, p129-134, January/April 2004.