Peer-to-Peer Cooperative Networking for Cellular Mobile Devices

Niranjan Suri[1], Giacomo Benincasa[1], Mauro Tortonesi[2], Enrico Casini[1], and Andrea Rossi[1,2]

[1] Florida Institute for Human and Machine Cognition, Pensacola, FL USA
{nsuri,gbenincasa,ecasini,arossi}@ihmc.us
[2] University of Ferrara, Ferrara, Italy
mtortonesi@ing.unife.it

Abstract. Cellular mobile devices, and in particular smartphones, have become ubiquitous. While bandwidth has steadily increased from 2G devices with Edge to 3G and now 3G LTE (4G), so has the demand for bandwidth intensive applications and streaming of multimedia content. Supporting high densities of such users in urban environments has become a challenge. In this paper, we describe an approach to peer-to-peer cooperative networking that exploits the WiFi interface in peer-to-peer mode in order to reduce the demand on the cellular network while at the same time increasing the reliability of data delivery. We describe multiple scenarios that benefit from such middleware and present some experimental results.

Keywords: Cooperative Networking, Peer-to-peer Networks, Multimedia Streaming, Information Dissemination, Opportunistic Communications.

1 Introduction

Mobile phones have evolved from simple devices that supported voice calling and text messaging into powerful and sophisticated multimedia devices with audio and video streaming. While the design capacities of the cellular networks have increased steadily from 2G devices with Edge to 3G and now 3G LTE (labeled 4G by some vendors), so has the user appetite for bandwidth intensive applications. Given that the cellular bandwidth is shared across multiple users, the available bandwidth quickly diminishes with an increase in user density. For example, the 3G LTE specification calls for a minimum of 100 Mbps downlink, but also a minimum support for 200 users in one cell. With large numbers of users, that reduces the bandwidth per user to 500 Kbps. Given the unreliability of wireless communications, the actual bandwidth realized is significantly lower, especially with users in indoor environments and with users on the move. Therefore, there is a requirement to alleviate the congestion on cellular networks and to increase the reliability of data delivery.

N. Venkatasubramanian et al. (Eds.): Mobilware 2011, LNICST 93, pp. 85–97, 2012.

One approach to solving this problem lies in exploiting with WiFi interface in order to offset the load on the cellular interface. Most mobile phones provide support for WiFi, which typically provides a shorter-range, higher-bandwidth communications link. Mobile phones are able to automatically switch their data access to the WiFi interface when an access point is available, in order to offload the cellular network. With Unlicensed Mobile Access (UMA), mobile phones are able to automatically switch their voice calls to the WiFi interface as well. This feature is often used with access points at home in order to compensate for poor cellular coverage in remote areas. While such an approach reduces the load on the cellular network, we propose an alternate approach that does not require WiFi access points.

This paper describes an approach to peer-to-peer cooperative networking among mobile phones in order to reduce cellular network load as well as increase reliability of data delivery. In particular, we propose to allow mobile phones to communicate via their WiFi interfaces, but in ad-hoc mode, with their peers. This peer-to-peer communication can be used to compensate for data loss on the cellular network by allowing one mobile phone that has received the data successfully to provide the data to another mobile phone that failed to receive the original data. We describe an abstract method to realize this capability, as well as a concrete implementation based on our DisService peer-to-peer information dissemination service. We begin by describing some scenarios.

2 Scenarios

2.1 Live Multicast Streaming

We begin with the simplest scenario – streaming live audio or video via a multicast transmission. This scenario involves multiple mobile phone users receiving a multimedia stream from a provider via their cellular network. Examples include watching live events such as a newscast, a show, or a sporting event. In this scenario, the multimedia stream being provided to each user is the same, regardless of when they subscribe to the stream. Therefore, the stream can be multicast over the cellular network and requires only the bandwidth for a single copy of the stream. We contrast this to a later scenario, where users can begin a stream from an arbitrary point on demand, which requires that the stream be potentially unicast to each user.

Cooperative networking in this scenario involves mobile devices using their peer-to-peer network to request and exchange packets that they did not receive successfully over the cellular network. In the absence of cooperative networking, a device that has missing packets would either have a break in the rendered media, or would have to request missing packets over the cellular network, requiring further bandwidth.

2.2 Live Multicast Relaying

This scenario is a variation of the simple streaming scenario above. In this situation, some of the mobile devices have either a poor or non-existent cellular link and hence cannot receive the multicast stream directly. For example, consider multiple users in

an office environment, where some users may be in interior rooms (with poor connectivity) whereas others may be in exterior rooms with good connectivity. Cooperative networking in this scenario involves one or more mobile devices that have a good quality connection relaying the stream that they are receiving to the disconnected (or poorly connected) mobile devices via the peer-to-peer WiFi link.

2.3 On Demand Streaming

This scenario differs from the previous two scenarios in that each mobile phone has an independent multimedia stream that is transmitted on demand. This scenario arises when users select viewing an archived multimedia stream from a website[1]. Given that the multimedia stream is not synchronized across users, it would not be possible to multicast the stream to multiple users. However, we make the observation that multiple users may watch a subset of popular videos at any given moment in time[2]. Consider a crowd of users gathered at an urban area, city center, or at a sporting event. Cooperative networking in this scenario involves one or more mobile devices caching their multimedia content for a period of time after it has been rendered in order to provide it to other peers on demand, over the peer-to-peer WiFi link. The duration of the cache may depend on a variety of factors, including recency of use and storage space available.

2.4 Generic Web and Data Caching

This final scenario is a generalization of the on-demand streaming and turns mobile phones into dynamic web and data caches. In this scenario, when a mobile phone decides to download content via the cellular network, it first queries other mobile devices via the WiFi link in order to determine if any of the peer devices already have the desired data in their cache. When a device retrieves an object (either via the cellular or WiFi link), the device caches the object in case it can provide the object to other peers at a later point in time. As in the above scenario, the duration of the cache may depend on a variety of factors.

The following section discusses related work that provides capabilities relevant to the above scenarios.

3 Related Work

Many research studies have focused on data replication in ad hoc networks. Padmanabhan et al. [1] and Derhab and Badache [2] provide a comprehensive survey of the topic. However, most of the proposals focus on the realization of medium to long term availability of data that gets rarely updated [3], and try to address related

[1] YouTube® is a popular worldwide example, along with Netflix® and Hulu® in the United States.

[2] This phenomenon is sometimes referred to as Cyber Rubbernecking.

issues such as network partition-aware [4] or energy consumption-aware [5] replication solutions. Other solutions leverage on federated tuple-spaces and configurable replication profiles [6]. Instead, we are interested in short term, time limited, and localized dissemination of data.

Among the existing proposals that seem to address a similar scenario to the ones presented in Section I, the most interesting one seems to be REDMAN [7]. However, REDMAN focuses on replica placement and does not address the problem of replica updates.

Researchers have also addressed the problem of reliable group communications in ad hoc networks. Several reliable multicast protocols designed to operate on top of an IP multicast infrastructure, such as NORM [8] and PGM [9], have also been proposed on mobile ad hoc networks. In addition, researchers recently started studying opportunistic communications, paying a particular attention to social networking concepts [10] [11] [12] [13] [14].

4 DisService

4.1 Overview

DisService is a peer-to-peer dissemination service that realizes cooperative networking for mobile devices. DisService is middleware that sits between the application and transport layer and offers an information dissemination service that provides a message-oriented, publish-subscribe paradigm.

DisService was designed to perform in mobile ad-hoc networks, and relies on the assumption that in this context, the cost of unicasting and broadcasting are equivalent and therefore the latter is always preferred to the former. The use of broadcast has the important feature of allowing all the neighboring peers to receive and opportunistically cache data that are not directed to them at no additional cost. We call this capability "opportunistic listening." Opportunistic listening allows a higher degree of availability and survivability of the data and ultimately allows greater overall dissemination performances.

Note that while we use the term broadcast in this paper to describe the exchange of data between DisService instances, DisService can also be configured to use multicast instead. The only requirement would be a mechanism to select a specific multicast address to be used by nodes. For example, all the devices that are receiving a specific live stream may subscribe to a single multicast group, which could be specified as part of the broadcast from the cell tower.

DisService implements several dissemination protocols, such as probabilistic (epidemic) protocols, reliable flooding, and heuristic protocols; the choice of which algorithm better suits the particular scenario is left up to the user. Furthermore, DisService lets the applications choose, for each of their subscriptions, whether they want the data to be delivered in a reliable and/or sequenced fashion.

Unlike most protocols, when reliable transmission is chosen, the receiver will be in charge of requesting the missing messages. This approach has the advantage of leaving the decision to request the missing fragments up to the receiver; it is possible

that different applications that subscribed to the same group have different reliability requirements. Moreover, this approach allows the receiver to autonomously retrieve the missing messages from different subsets of peers.

The guaranteed decoupling between sender and receiver as a result of the publish/subscribe architecture, the extensive use of broadcasting, and the use of "reliable reception" instead of "reliable transmission" all make DisService an effective architecture to deal with unreliable networks by means of cooperative caching. A more detailed architectural view of DisService is presented later, following an example of using DisService for the scenarios presented above.

4.2 Live Multicast Streaming Scenario

Consider the live multicast streaming scenario presented in section 2.1. Figure 1 below shows a snapshot of two users receiving a live feed from a cell tower and using cooperative networking in order to improve their reception. Each mobile device has a FIFO buffer that is used to hold packets that have been received via the cellular interface but have not yet been consumed and rendered by the device. Such a buffer is normally used to offset any jitter that might occur in the reception of data. In this scenario, this same buffer is also used to fill-in missing packets by requesting that data from neighboring peers, over the WiFi interface. For example, User 1 did not happen to receive packet n+4 and n+6 and User 2 did not receive packet n+3 and n+6. Each of them requests the packets they are missing from other users, with the result that User 1 receives packet n+4 and User 2 received packet n+3. Neither of them received packet n+6, so that would result in some degradation in the received stream despite the cooperative networking capability.

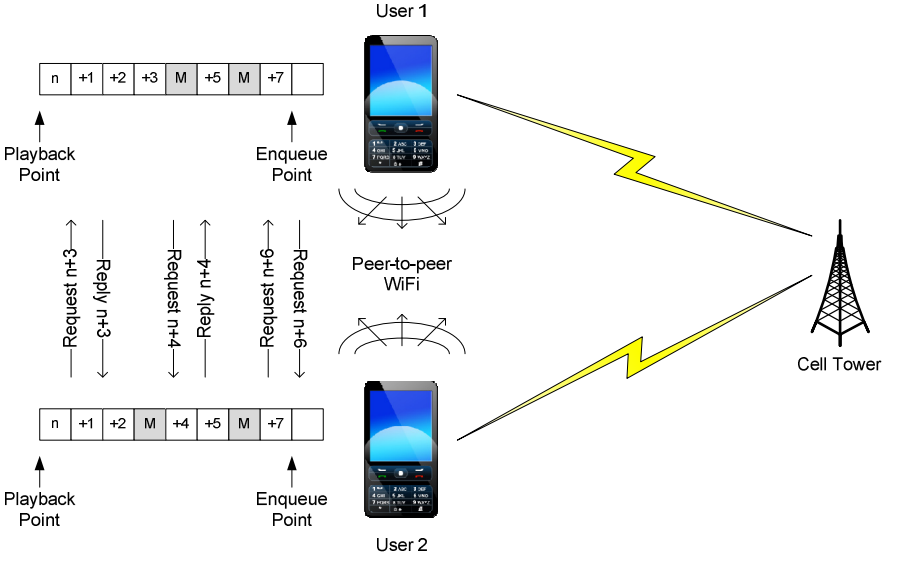

Fig. 1. Live Multicast Streaming Scenario with DisService

While the example above demonstrates a two user scenario, this is easily extensible to multiple users, in which case a user's device may receive the missing packets from any of the other devices. In the DisService architecture, the request for a missing packet is broadcast to all the other devices within reach, and any of them that have the desired data may respond.

Also note that the stream being transmitted over the cellular network will likely use Forward Error Correction, which embeds some redundant information in each of the packets in order to support recovery of missing packets. The DisService approach to cooperative networking is fully compatible with such streams of data. In the example above, the decoder for the multimedia stream may still be able to recover the data missing in packet n+6, which was not received by either user. Using DisService decreases the number of missing packets, which increases the quality of the rendered stream as well. Cooperative networking will also reduce the quantity of redundant error correction information that is selected to be included in the stream, which reduces the bandwidth required.

We are currently working on also realizing variations of DisService to support the other three scenarios described in sections 2.2, 2.3, and 2.4 respectively. In the following section, we describe the overall architecture of DisService.

4.3 DisService Architecture

DisService provides efficient and peer-to-peer dissemination of data without any reliance on centralized components. There are no assumptions made about the presence of stable network connectivity. Instead, DisService dynamically adapts itself to network changes and disseminates information as best as possible. Figure 2 shows the DisService architecture. The DisService components are described in the following subsections.

DisService has been realized on a variety of platforms, including the Android platform for mobile devices. The core capabilities of DisService are built as a C++ library using the Android Native Development Kit (NDK), with a Java Native Method Invocation (JNI) layer to support Java applications. Therefore, native Java applications are able to use the full capabilities of DisService. On non-android devices, DisService also supports C++ and C# applications by means of a proxy layer.

4.3.1 Message Propagation Service

This service provides two capabilities for efficient use of bandwidth: message consolidation and piggybacking. The first capability allows DisService to send multiple individual messages that are automatically consolidated into network packets in order to minimize the number of packets injected into the network. The consolidation capability allows DisService to include a delay tolerance in each message transmission request. This specifies how long the message should be kept in order to consolidate it with other messages. In addition, message consolidation allows DisService to send multiple, small messages without having to worry about the number of packets being generated. This is particularly important for some packet rate limited radios.

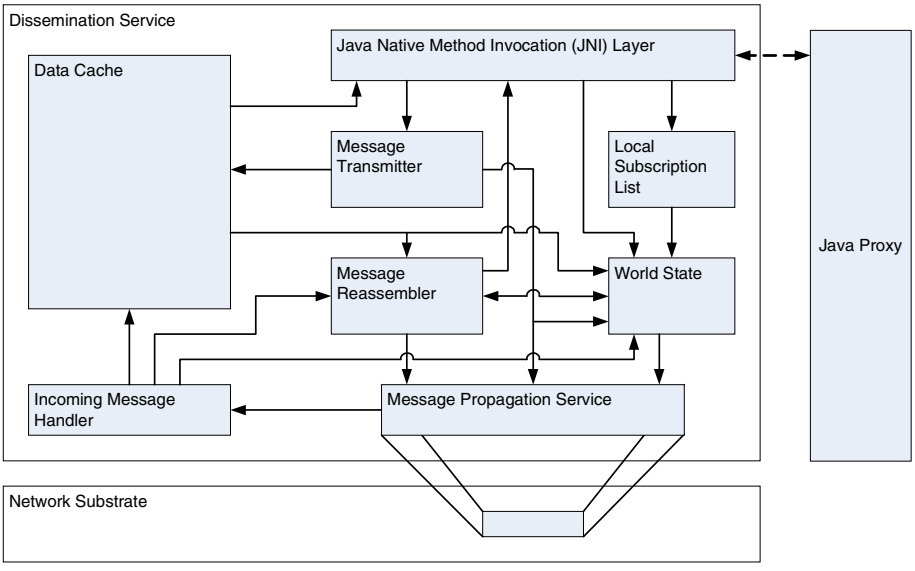

Fig. 2. Architecture of DisService

4.3.2 Data Cache

The design choice for DisService is to aggressively cache data on every node, limited only by the local storage capacity. Therefore, any data that has been previously pushed or received by a node is held in the Data Cache. This allows the node to readily provide the data both to local applications as well as any peer nodes that need the data. The current implementation of the Data Cache uses the SQLite library [15], a public domain embeddable SQL database library. The Data Cache can be configured to suit application requirements. For example, for the live multicast streaming scenario described in section 2.1, the Data Cache only needs to hold as many packets as the overall streaming buffer size. However, each of the subsequent scenarios described require more and more caching of data.

Expiration of data is controlled via an Expiration Controller that can take into account different policies for expiration, based on the group, sender, last request time, and potentially other parameters for selecting data to expire.

4.3.3 Message Transmitter

The Message Transmitter handles fragmentation of large messages and controls the bandwidth utilization of outgoing traffic. Messages that are larger than the Maximum Transmission Unit (MTU) are automatically fragmented. Each message contains a header that identifies the portion of the data contained in the message. Messages that are transmitted may be rate-limited in order to not overload the network. When multiple messages are awaiting transmission, the message priority is used to determine the transmission order.

4.3.4 Incoming Message Handler

Messages received by the Message Propagation Service are handled by the Incoming Message Handler, which examines the header to identify the nature of the message. An incoming message may contain payload data or control data. For payload data, the handler checks if the whole data is present in the current message or the data has been fragmented. Fragmented data is handled by the Message Reassembler. Before delivering the data to the application, the sequencing rules for the subscription are checked. If sequencing has been requested, an out of order message is not delivered. After that, the message is delivered to the correct applications.

Control messages are handled differently. Two types of control messages are possible. A World State message includes information about a neighbor node and is handed off to the local World State component. The second type is a Data Request message. When a data request arrives, the Incoming Message Handler checks both the Message Reassembler as well as the Data Cache. This is because partially received messages are stored in the Message Reassembler until they are complete before being stored in the Data Cache. In situations where only partial data is available, the local node will transmit the subset of data that is available. This contributes to satisfy the request.

4.3.5 Message Reassembler

This component takes incoming data fragments and reassembles them in the correct order. If reliable delivery has been requested by the subscribing application, the Message Reassembler identifies missing fragments, requests them from other nodes, and performs the reassembly procedure when all the missing fragments have been received. If an application has subscribed to the group with a request for sequenced delivery, two possibilities exist. If reliability is requested too, messages received out of order are buffered until the missing messages are received and then delivered in order to the application. If no reliability is requested, old messages that are received later are simply dropped. That is, delivery of a message with sequence number n ensures that no message prior to sequence number n will ever be delivered. The Message Reassembler periodically checks for missing messages or messages that have missing fragments and sends out a request to the peers for retransmission of the missing fragments.

4.3.6 World State

The World State maintains the best known information about the state of other nodes in the network, including the messages that they contain. Given the distributed nature of the system, this information might not be accurate and up to date. Each node maintains its own view of the world in the local World State component. As part of the World State, each node maintains information about local neighbors and their subscriptions, as well as all known remote nodes. The information held for a remote node includes the distance to the remote node (in terms of the number of network hops), the path to the remote node, and the lowest link capacity, which limits the overall bandwidth available to that node. A sequence number is attached to each World State, which is incremented at every change to the World State. Periodically

each node will broadcast its presence and the sequence number of its World State. This broadcast is received by all the peer nodes, which may request the complete World State if the node is new or if the sequence number has been updated. This reduces unnecessary transmissions of the World State.

4.4 Unique Features of DisService

The DisService design and implementation incorporate four unique features that are described in this section.

4.4.1 Self-describing and Self-contained Packets

DisService includes sufficient metadata in each packet transmitted on the network in order to allow any node receiving an individual packet to be able to interpret the packet correctly. In particular, each packet contains the identity of the sender, the group context for the message, the unique message sequence number for the message (unique from the perspective of a sender in a group), as well as the offset and length of the packet payload in the context of the message. This metadata allows any peer node in the network to receive and cache packets, and be able to respond to missing fragment requests from other peers.

4.4.2 Aggressive Broadcasting and Caching of Data

As described earlier, DisService always transmits data by broadcasting (or multicasting) the data, and aggressively caching the data on any node that happens to receive it. Combining this capability with the above notion of self-describing packets allows DisService to realize the notion of cooperative networking as described in this paper. Data that is cached is stored in the SQLite database, which is used to realize the data cache inside of DisService.

4.4.3 Neighbor Dependent Probabilistic Response Model

An important aspect of DisService is that a request for data may be received by multiple peers, which act independently from each other. In such cases, the receiver may get multiple responses for the same data request, wasting network capacity. In order to reduce this duplicated traffic, the probability of a node responding to a request is computed based on the number of neighbors of the requesting node. Each node maintains the number of its neighbors. When a node transmits a request for data, the request is received by all the neighbors. All of them potentially reply and transmit duplicate copies of the data. To avoid this, the requesting node includes its neighbor number, in the data request. Each node will then transmit with a probability that is the inverse of the number of neighbors. It is also possible that some nodes don't have the data sought. For instance, if just one neighbor has the data, the requesting node may not receive the data sought, due to the nature of the probabilistic response model. To alleviate this situation, when a node sees a repeated request for data, the probability measure is ignored and the requested data is always transmitted.

4.4.4 Reliable Reception Instead of Reliable Transmission

When reliable data transmission is desired, traditional network protocols such as TCP rely on the sender to ensure that the data being transmitted is received by the recipient. The DisService model differs significantly from the point-to-point model assumed by TCP. The first difference is that communication in DisService is point-to-multipoint: several peer nodes may be recipients of some data transmitted by one node. The second difference is that each recipient may independently request or not request reliable reception of data. The third difference is that the set of nodes that are reachable may change continuously. For instance, if a node is pushing information to another, and the second node moves away and loses network connectivity with the pusher, the TCP model would result in having the pusher node continuously attempting to retransmit data to the receiver. In the DisService model, when the receiver moves away, the pusher node does not care. Eventually, if the receiving node comes in contact again with the original pusher or some other node which has the messages, then the receiver will request and receive the missing messages at that time. Another difference is that the recipient node does not need to go back to the original transmitting node to get the missing information. This is because it may be obtained from any other peer node that has the desired information. This is an effective strategy for reliable delivery, especially when coupled with the design choice to aggressively cache as much data as possible at each node.

5 Experimental Results

The following experimental results show the benefits of cooperative networking with mobile cellular devices. We use a scenario consisting of 10 mobile devices that are receiving data from one transmitter via a cell tower. We use different settings for the reliability of the network link, and report on the number of packets that are received with or without the use of cooperative networking via DisService. Figure 3 below shows the scenario for the experiment.

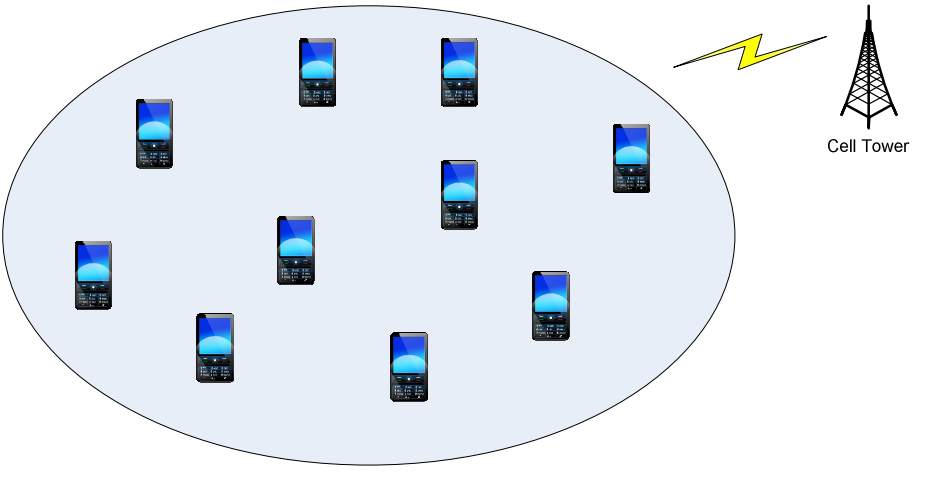

Fig. 3. Cooperative Networking Experiment Scenario

The experiment has been setup on the NOMADS Testbed, which emulates mobile ad-hoc networks and wireless links. The testbed allows control of the capacity, reliability, and latency of each individual network link in the network being emulated.

We set the capacity of the link between the cell tower and the mobile nodes to 256 Kbps, while the mobile nodes are connected in a full topology with links of 1 Mbps of capacity. The reliability of the cellular link varies from 70% to 80% in the first case, and from 30% to 50% in the second case. The results, in Table 1, show how the capacity to retrieve messages from cooperating peers benefits the performances in terms of number of delivered messages. Furthermore, as expected, the results show that the improvement is higher when the reliability of the cellular link is lower.

It is important to note that when a link reliability is set to a certain value (e.g., 70%), the implication is that the independent probability of a packet being delivered correctly to each node is 70%. Hence, when a specific packet is sent to the set of six nodes, the probability of at least one of the nodes receiving the packet and sharing it with the other nodes over the peer-to-peer link is high.

Table 1. Performance Improvement of Packets Received with DisService

Cellular Link	P2P Network	Packets	Packets Rcvd		Success Rate	
Reliability	Reliability	Sent	Multicast	DisService	Multicast	DisService
70% to 80%	80%	1293	976.0	1293.0	75.48%	100.00%
30% to 50%	80%	1305	546.6	1297.1	41.89%	99.39%

As described earlier, one of the benefits of cooperative networking is to save bandwidth by transmitting less Forward Error Correction (FEC) data. Using NORM [8], we sent a data stream of 800 KB in total to each of the 10 mobile nodes. We set the cellular link quality to 70%, and measured the number of nodes that successfully received the complete stream intact, with varying amounts of FEC data. Note that in the case of DisService above, all 10 nodes successfully received the complete stream.

The results, shown in Table 2, indicate that in order for all 10 nodes to receive the complete stream, NORM must be configured to use 48 FEC blocks per 64 blocks of real data (i.e., an FEC overhead of 75% and a total measured overhead of 94%). This extra bandwidth could be used for transmitting other useful data, or be used to improve the quality of the stream, such as increased resolution. As the results indicate, using cooperative networking significantly reduces the bandwidth requirements on the cellular network while at the same time improving the reliability of the data delivered.

Table 2. Forward Error Correction Performance in NORM

FEC Blocs per 64 Data Blocks	Stream Size (Bytes)	Total Bytes Transmitted	Nodes Successfully Receiving Stream	FEC Overhead	Total Overhead
16	811008	1102640	0	25.00%	35.96%
32	811008	1339760	4	50.00%	65.20%
36	811008	1399040	8	56.25%	72.51%
40	811008	1458254	9	62.50%	79.81%
48	811008	1576484	10	75.00%	94.39%

6 Challenges for Cooperative Networking

Security is one of the primary challenges raised by cooperative networking as suggested in this paper. In particular, a user may be concerned with a peer user maliciously modifying the content of a packet that is being forwarded through another peer. To some extent, such a problem occurs with other peer-to-peer protocols such as BitTorrent [16] as well. BitTorrent addresses the tampering problem by using checksums. Each block of data has an independent checksum that is included in the original meta information that is downloaded by each peer. DisService can also implement a checksum scheme. Since the data is being continuously streamed from a central location, an effective approach is to using a rolling checksum, which computes a checksum over the last x packets. For example, if x is set to 6, packet n would contain cumulative checksums for packets n-5, n-4, n-3, n-2, n-1, and n. Such a checksum mechanism would allow a peer to receive and validate missing packets from other peers. As long as the rolling checksum did not exceed the buffer window size, each node can validate the packets prior to attempting to consume them (e.g., by trying to decode the multimedia stream).

A second challenge raised by cooperative networking is resource utilization of the peers. For example, mobile devices would have to activate their WiFi interfaces in addition to their cellular interfaces, which could consume additional power. This is a tradeoff between improved reliability, lower bandwidth utilization over the cellular link, and increased battery utilization of the mobile devices.

A third challenge is being able to setup the mobile devices to use WiFi networks in Ad-hoc mode. For example, a current limitation in the Android operating system prevents many handsets from creating ad-hoc peer-to-peer networks between themselves. A workaround is to have another node (e.g., a PC) create the ad-hoc network, and have the mobile nodes join the ad-hoc network. Another possibility is to use base stations to create the WiFi networks.

7 Future Work

Much work remains to be done to further explore and exploit the notion of cooperative networking for mobile devices. We are currently enhancing the DisService-based solution to also address the other three scenarios mentioned in this paper. We are also integration DisService into an Android-based application to show video, either streaming, or archived (e.g., from YouTube®). Finally, we are conducting additional experiments to measure the advantages of cooperating networks under different network conditions.

Acknowledgements. This research was sponsored in part by the U.S. Army Research Laboratory under Cooperative Agreement W911NF-04-2-0013.

References

1. Padmanabhan, P., Gruenwald, L., Vallur, A., Atiquzzaman, M.: A survey of data replication techniques for mobile ad hoc network databases. VLDB Journal 17(5), 1143–1164 (2008)
2. Derhab, A., Badache, N.: Data Replication Protocols for Mobile Ad-Hoc Networks: A Survey and Taxonomy. IEEE Communications Surveys & Tutorial 11(2) (Second Quarter, 2009)
3. Hara, T.: Data Replication for Improving Data Accessibility in Ad Hoc Networks. IEEE Transaction on Mobile Computing 5(11) (November 2006)
4. Wang, K., Li, B.: Efficient and guaranteed service coverage in partitionable mobile ad hoc networks. In: 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002), vol. 2, pp. 1089–1098 (2002)
5. Thanedar, V., Almeroth, K.C., Belding-Royer, E.M.: A Lightweight Content Replication Scheme for Mobile Ad Hoc Environments. In: Mitrou, N.M., Kontovasilis, K., Rouskas, G.N., Iliadis, I., Merakos, L. (eds.) NETWORKING 2004. LNCS, vol. 3042, pp. 125–136. Springer, Heidelberg (2004)
6. Murphy, A.L., Picco, G.P.: Using LIME to Support Replication for Availability in Mobile Ad Hoc Networks. In: Ciancarini, P., Wiklicky, H. (eds.) COORDINATION 2006. LNCS, vol. 4038, pp. 194–211. Springer, Heidelberg (2006)
7. Bellavista, P., Corradi, A., Magistretti, E.: REDMAN: An optimistic replication middleware for read-only resources in dense MANETs. Pervasive and Mobile Computing 1(3), 279–310 (2005)
8. Adamson, B., Bormann, C., Handley, M., Macker, J.: NACK-Oriented Reliable Multicast (NORM) Transport Protocol. IETF Request For Comments 5740 (November 2009)
9. Speakman, T., Crowcroft, J., Gemmell, J., Farinacci, D., Lin, S., Leshchiner, D., Luby, M., Montgomery, T., Rizzo, L., Tweedly, A., Bhaskar, N., Edmonstone, R., Sumanasekera, R., Vicisano, L.: PGM Reliable Transport Protocol Specification. IETF Request For Comments 3208 (December 2001)
10. Zyba, G., Voelker, G., Ioannidis, S., Diot, C.: Dissemination in Opportunistic Mobile Ad-hoc Networks: the Power of the Crowd
11. Hui, P., Crowcroft, J., Yoneki, E.: BUBBLE Rap: Social-based Forwarding in Delay Tolerant Networks. In: MobiHoc (2008)
12. Mtibaa, A., May, M., Diot, C., Ammar, M.: PeopleRank: Social Opportunistic Forwarding. In: INFOCOM Mini Conference (2010)
13. Hossmann, T., Spyropoulos, T., Legendre, F.: Know Thy Neighbor: Towards Optimal Mapping of Contacts to Social Graphs for DTN Routing. In: INFOCOM (2010)
14. Daly, E.M., Haahr, M.: Social Network Analysis for Routing in Disconnected Delay-Tolerant MANETs. In: MobiHoc (2007)
15. SQLite Relational Database Library. Online Reference, `http://sqlite.org/`
16. BitTorrent Protocol Specification. Online Reference, `http://www.bittorrent.org/beps/bep_003.html`