Data Dissemination in the Wild

A Testbed for High-Mobility MANETs

Vingelmann, Peter; Pedersen, Morten Videbæk; Heide, Janus; Zhang, Qi; Fitzek, Frank

Published in:
2012 IEEE International Conference on Communications (ICC)

DOI (link to publication from Publisher):
10.1109/ICC.2012.6364123

Publication date:
2012

Document Version
Early version, also known as pre-print

Data Dissemination in the Wild: A Testbed for High-Mobility MANETs

Péter Vingelmann[*†], Morten Videbæk Pedersen[†], Janus Heide[†], Qi Zhang[‡], Frank H. P. Fitzek[†]
[*] Department of Automation and Applied Informatics, Budapest University of Technology and Economics, Hungary
[†] Department of Electronic Systems, Faculty of Engineering and Science, Aalborg University, Denmark
[‡] Department of Engineering, Aarhus University, Denmark

***Abstract*—This paper investigates the problem of efficient data dissemination in Mobile Ad hoc NETworks (MANETs) with high mobility. A testbed is presented; which provides a high degree of mobility in experiments. The testbed consists of 10 autonomous robots with mobile phones mounted on them. The mobile phones form an IEEE 802.11g ad hoc network to communicate with each other. A dynamic network topology is assumed, where the mobile devices form a cooperative cluster in order to exchange data packets among each other. In our multimedia exchange scenario, the initial state is that one device carries all information, and the goal is to convey that information to all devices. A strategy is proposed that uses UDP broadcast transmissions and random linear network coding to facilitate the efficient exchange of information in the network. An application is introduced that implements this strategy on Nokia phones. The measurement results collected from the testbed are presented, and the performance of the proposed strategy is compared to a reference strategy that is based on TCP unicast connections.**

Keywords – data dissemination; MANET; mobility; testbed; robots; network coding

I. INTRODUCTION

The problem of efficient data dissemination in mobile ad-hoc networks is a particularly interesting research topic. Typically we intend to share some kind of multimedia content, such as pictures, audio or video files, originating from a single *source* with a larger set of receivers or *sink* nodes. The size of the data set to be shared can vary greatly depending on the scenario. This content is always divided into numerous packets because of the limitations of the underlying communication systems. The goal is to ensure reliable delivery for the content as a whole, and not only for the individual packets.

Since the same data set should be delivered to all receivers, the broadcast nature of the wireless channel allows for an efficient delivery of innovative information. The existing reliable broadcasting protocols for mobile ad hoc networks [1], [2], [3] generally focus on the delivery of individual packets, which is an inherent limitation of these protocols that results in significant management overhead. It is more efficient to consider the entire data set as a unit, and apply some sort of coding scheme on top of it. Using a rateless erasure code, a single coded packet can convey useful information about the whole data set.

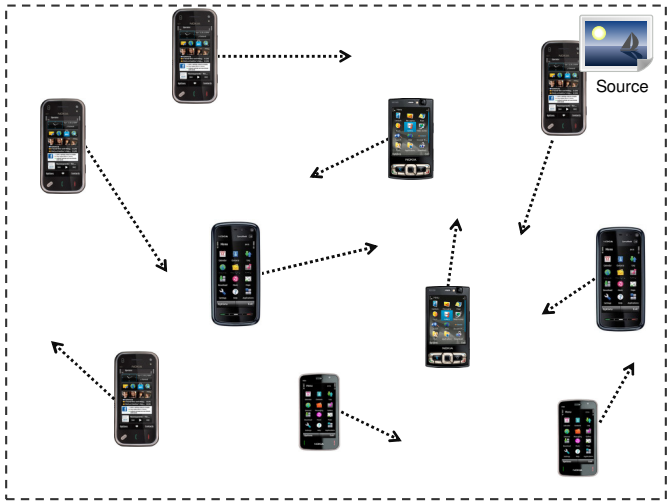

Figure 1. Multimedia exchange scenario with 10 moving devices

Thus it suffices for a receiver to retrieve *any subset* of coded packets of size slightly larger than the set of source packets.

Network coding [4], [5] is well-suited for this task because of its clear advantages for mobile multi-hop communication systems [6], [7], [8]. Our previous work also focused on the feasibility and tuning of network coding for mobile platforms [9], [10]. The authors in [11], [12] proposed to use network coding to enhance cooperation among network nodes in various communication scenarios. We have also developed a protocol based on network coding that forms a cooperative cluster of mobile devices to repair packet losses on-the-fly [13].

The authors in [14] analyzed the problem of data dissemination and proposed a centralized MAC–level scheme that also uses network coding. This work was extended in our previous paper [15], where static multi-hop networks were investigated. We developed a custom-built simulator and compared the performance of several data dissemination schemes.

As the next logical step, we continue this work by adding mobility to the network nodes (see Figure 1). The process of data dissemination is different in a dynamic network, where an agile protocol is required to enable the cooperating devices to discover transmission opportunities when other devices come into their communication range.

Simulations are the fastest and most convenient way of evaluating various transmission schemes and protocols, but simulation results can also be misleading if the underlying transmission characteristics are not completely realistic. Therefore we have assembled a testbed to conduct real-life experiments with mobile phones and robots in different environments.

This paper is organized as follows. Section II presents the investigated scenario, and then Section III introduces our testbed. The proposed strategy is described in Section IV, in addition Section V discusses its implementation on mobile phones. Section VI outlines an alternative strategy. Section VIII presents measurement results collected from our testbed, and the final conclusion is drawn in Section VIII.

II. SCENARIO

In our multimedia exchange scenario (depicted in Figure 1), we assume that several mobile devices form an IEEE 802.11g ad hoc network to communicate with each other. A data set with a size of several megabytes is available on one of the devices. Thus the initial state is that the *source* node carries all information, and the goal is to convey this data set to all *sink* nodes. Our objective is to minimize the time and energy needed to perform this task.

We assume that all nodes are constantly moving, and they are partially connected in a possibly multi-hop network. A dynamic network topology is assumed, where the nodes frequently make contact with each other, exchange some data, and then possibly move out of range. The mobile devices should remain within the designated measurement area, which is preferably of rectangular shape.

Mobility should be provided by moving robots on which the mobile devices can be mounted. The robots are meant to imitate human motion to some extent (and obviously on a smaller scale). The main requirements regarding the robot platform are sufficient speed, ease of steering, durability and programmability.

III. TESTBED

A. Robot platform

Since the robots are meant to represent humans, we do not assume any control or communication from the phones to the robots. The robots are considered autonomous units that follow their own programming.

It is important that the robots provide a sufficient level of mobility. Their speed must be comparable to the transmission range of the mobile devices. If they are too slow, then we end up with a static and partially connected network. In this case, all information could be exchanged very quickly, as the phones remain in range of each other for a relatively long time. Ideally the robots should be able to drive as fast as humans walk.

Different robot platforms were tested to determine if they can provide the required mobility to the mobile phones. We considered assembling a testbed similar to [16] using vacuum cleaner robots, and we experimented with the iRobot Create, which was built especially for research purposes. It is equipped with a programmable microcontroller and bumper sensors, but it is quite slow, especially outdoors.

Figure 2. LEGO robots carrying 10 Nokia mobile phones

As an alternative, we tested the LEGO Mindstorms NXT robot platform. We built several different prototypes, and we ascertained that these robots are indeed suitable for our needs. Our final prototype is similar to a remote controlled car, as can be seen in Figure 2. These cars can drive faster than the iRobot Create (max. 10 km/h), and they can also drive outdoors if equipped with proper wheels. A programmable unit, the NXT brick is responsible for controlling the motors and sensors onboard the robot.

We have built 10 copies of this prototype, and each robot is equipped with an ultrasonic and a color sensor from the standard LEGO NXT kit. The main purpose of these sensors is to keep the robots within the designated measurement area, and to prevent any collisions between robots and solid objects. The ultrasonic sensor can measure the distance of any obstacle in front of the robot, and the color sensor signals when the robot drives over a green line that we painted to mark the edge of the measurement area.

We programmed the robots using the Not eXactly C (NXC) language, which is similar to standard C, but designed specifically for the NXT brick. It supports the concept of multiple "tasks" that are executed in parallel.

The robots are programmed to follow a random path. From the starting position, they drive straight ahead for a random time interval (between 20 and 45 seconds), then they turn with a random angle, then drive straight again, and so on. When the ultrasonic sensor detects an obstacle, the robot is programmed to drive backwards, turn 30-90 degrees, then continue straight ahead. When they drive over a green line, they switch to full reverse for 2 seconds, and then turn around. This containment method generally works well outdoors, although it may not be perfect depending on the lighting conditions.

B. Mobile phones

The robots have holders where the mobile phones can be docked during the experiments. We used 7 Nokia N97 Minis, 1 Nokia N97, and 2 Nokia 5800 XpressMusic devices to form an IEEE 802.11g ad hoc wireless network for our measurements. These mobile phones have similar hardware and software specifications that are summarized in Table I. The transmission power was adjusted to 10 milliwatts on these Nokia phones in order to lower their transmission range. The automatic transmission settings were also disabled.

TABLE I. SPECIFICATIONS OF THE NOKIA N97 AND 5800 XPRESSMUSIC

Operating System	Symbian OS S60 5th edition
CPU	ARM11 @ 434 Mhz
Memory	128 MB SDRAM
Display	640 x 360 pixels, 3.2 inch
Battery	5800 XpressMusic: BL-5J (3.7V, 1320 mAh) N97: BP-4L (3.7V, 1500 mAh) N97 Mini: BL-4D (3.7 V, 1200 mAh)

IV. OPERATING PRINCIPLES

Our objective is to minimize the time and energy needed to perform the data dissemination task. This can be achieved if we can make the best use of the transmission opportunities in this dynamically changing network topology. We propose a strategy that operates based on the following principles.

A. Detecting other devices in range

The first step is to detect nearby devices with which information can be exchanged. All network nodes periodically broadcast short *update* messages on UDP to notify the other devices about their presence. The advantage of this approach is that the *update* messages also carry valuable information about the current status of the sender node. This information can be used as feedback for the data dissemination process.

B. Data transfer

We consider several megabytes of data that should be conveyed to all receivers from a single *source* device. The data set is divided into packets of 1400 bytes. With Random Linear Network Coding (RLNC), we can take several of these original packets, and generate linear combinations of them over a finite field using random coefficients. The linear combinations will contain information about all of these packets, but they will have the same size as a single original packet, since addition and multiplication are performed over a finite field. Ideally, we would generate a linear combination of all packets in the data set, but if there are several thousand packets, then the computational complexity of the encoding and decoding operations would be prohibitively high on a mobile device [10]. Therefore we have to partition the data set into several chunks also called generations, each consisting of g packets. A generation is a series of packets that are encoded and decoded together.

Once a nearby device is detected, the actual data transfer may begin. Each *update* message contains a bit vector (also called "knowledge vector") where each bit indicates if a given generation has been decoded on the sender device. Thereby the receivers know which generations are missing on the other devices in range. Based on this information, a receiver can start sending uncoded data packets (systematic coding) from a chosen generation. Later the losses can be repaired with encoded packets which are linear combinations of the original data packets in the current generation. To minimize encoding and decoding complexity, we can use GF(2) as a finite field as described in [9]. On the receiver, these encoded packets are passed to the decoder, which will be able to reconstruct the original data packets after receiving at least g linearly independent (coded or uncoded) packets from a given generation.

If a node is inactive when receiving an *update* message, then it tries to select a random generation that is not present on the other device, but already received on this node. The chosen generation should be transmitted as fast as possible. Both forward and backward error correction methods can be used with network coding to repair packet losses. A simple solution is to transmit extra coded packets right after a generation to combat anticipated losses.

Each node maintains an up-to-date list of the knowledge vectors of the other devices in range. Thus after sending a full generation, a node can continue by choosing a generation that is missing on all devices in range. If there is no such generation, then it attempts to select one that benefits the most nodes in the vicinity.

C. Completion and restart

The source node should be able to determine when all receivers have decoded the current data set. The *update* messages also carry the completion status flags of all receivers. These Boolean values indicate which nodes are finished according to the sender of the *update* message. Thereby information about the other nodes can be quickly propagated through the whole network, and the source can monitor the status of all nodes. Note that this method is similar to hearsay in real world. When all nodes are finished, the source initiates the distribution of a new data set, possibly with different size and parameters.

V. IMPLEMENTATION

We have developed an application to implement the proposed strategy on Nokia phones based on the principles outlined above. We used the cross-platform Qt framework that uses standard C++, but makes extensive use of a special code generator (called the Meta Object Compiler, or moc) together with several macros to enrich the language. Qt is the recommended framework to develop high performance, native applications on Nokia phones.

The application can monitor neighboring devices, and measure their momentary *connection quality* by counting the received (and lost) *update* messages during the last second. After adjusting several parameters, the source device is started manually, but the rest of the dissemination process is fully automated. The application also displays the progress of data dissemination by showing which parts of the data set are received on the given device. Data transfer is carried out by transmitting UDP broadcast packets with a payload size of 1400 bytes that is the optimal size on WLAN to avoid fragmentation.

The application records all important events and status updates to ASCII log files. The *update* messages are numbered by their sender, and they are also saved to the log file when sent or received. Each device sends 5 *update* messages per second. Later the existing connections and the network topology can be reconstructed for any time instant by examining the log files.

For visualization purposes, the application records positional data obtained from GPS receivers on the phones. The absolute accuracy of GPS coordinates is not relevant, since we only intend to record a relative path from a given starting point. The recorded path might have deviations from the actual path, but it can be used to visualize the motion of the robots in a simulator with respect to a common reference point.

Another important requirement is the correct temporal ordering of all recorded events; otherwise we would experience temporal inconsistencies when we try to analyze the recorded log files. To tackle this problem, we have implemented multi-client clock synchronization among the Nokia phones. This protocol enables the phones to precisely synchronize their timers with their neighbors. One device is selected as the master time server, and the others are to be synchronized to this one. The measured accuracy of this method is usually better than 1 millisecond, which is more than sufficient for our needs.

During the development process, we encountered strange issues with the operation of UDP sockets in Qt. After sending and receiving for several minutes, some devices just stopped receiving data on the UDP sockets. This problem is rather inexplicable, since it only happens at data rates higher than 100 KB/sec (for receiving). We believe that the UDP socket implementation in Qt for Symbian (or the OpenC library that is used in Qt) has a bug which causes this issue. Therefore we decided to use a native Symbian socket implementation that works normally even at high data rates. It also provides an option to make the UDP socket blocking, which guarantees that all packets will be sent by the MAC layer. Normally, some of the UDP packets would be dropped in case of network congestion, or if the application attempts to send data too fast. With the blocking flag enabled, the "writeDatagram" function will block until the MAC layer can actually transmit the packet. This solution can be used to achieve the highest possible sending rate for UDP connections. We tested the performance for UDP unicast transmissions with two devices in close proximity, and we observed that the application-level data rates can reach 1400 KB/sec. When broadcasting on UDP, we measured 1050 KB/sec at best.

VI. Reference Strategy

The strategy described above is based on transmitting data as UDP broadcast messages, but unicast connections (UDP or TCP) might perform better in certain situations. Unicast connections have the advantage of using link-level acknowledgments and rate adaptation, which are unavailable when broadcasting. Therefore we have implemented another strategy that uses TCP unicast connections to perform the data dissemination task among the mobile devices. This strategy is implemented within the same application, and it uses the same *update* messages to detect other devices and to signal their current status. The only difference is the method of data transfer: direct TCP connections are established from one device to another.

When the source receives an *update* message, it opens a TCP connection to the sender of that message, and it starts transmitting those parts of the data set which are not present on the sender. The transfer continues until the entire data set is delivered or until the connection is lost (when the devices move out of range). TCP provides reliable data transfer with the automatic retransmission of lost packets, therefore no additional coding is necessary. To avoid congestion, the *source* can have at most 3 active transfers at a time. The *sink* nodes can also propagate the data set to other devices. They select which parts to send based on the information in the received *update* messages. A *sink* node can only have 1 outgoing TCP connection at a time to keep the total number of simultaneous data transfers to a minimum.

We observed that if the number of active connections is not controlled, then the network becomes congested very quickly. The uncontrolled *sink* nodes might even suppress the outgoing transmissions from the *source*, which results in extremely high completion times. The controlled approach ensures that data is transmitted as fast as possible from one device to another by minimizing the probability of congestion in the network. The implementation uses a standard TCP socket in Qt, which emits a signal every time data has been written to the network device. The signal handler writes a new batch of data to the socket if the output buffer is empty. This solution provides optimal data rates with low CPU usage. A data rate of 1300 KB/sec can be achieved with a TCP unicast connection with 2 devices located close to each other.

VII. Results

Numerous experiments were carried out with the described testbed, and several hundred data dissemination test runs were completed. The measurements were performed in both outdoor and indoor environments for comparison.

To illustrate how the connections change over time among the 10 network nodes, we calculated the number of direct neighbors and the connection qualities as measured on each device. For each second, we considered the incoming *update* messages from other devices. This tells us how many nodes were in communication range at that second. We can also determine the *connection quality* which is a fraction between 0 and 1 that indicates how reliable the connection was to another device. For example, if 3 *update* messages were received from a given node, then the *connection quality* is 3/5 to that device.

We have chosen a secluded, asphalt-paved parking lot with dimensions of 52 by 35 meters for outdoor measurements. Figure 3 shows the number of direct neighbors (each neighbor is weighted with its *connection quality*) to for each of the 10 devices during a 15-minute-long measurement at this location. This experiment was carried out after hours, consequently there were no vehicles in the parking lot. The starting position of the 10 robots was very close to each other, and they were following a random path for 15 minutes. As a result, the number of neighbors approached 9 in the beginning, but later – as the robots moved apart – this value decreases. On some occasions,

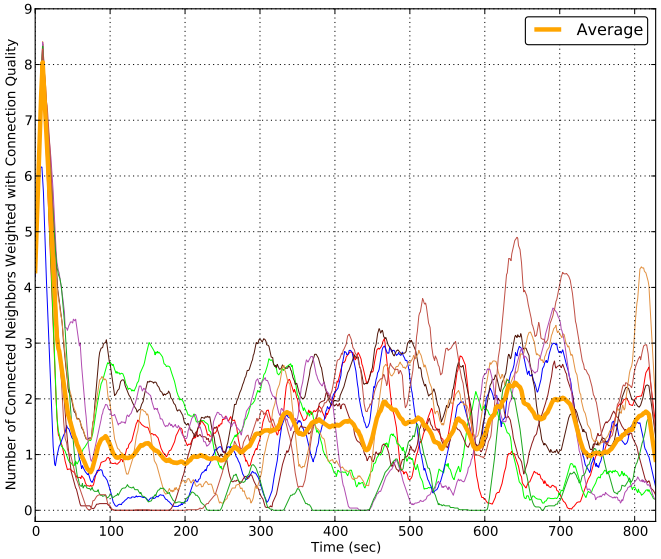

Figure 3. Number of direct neighbors during an **outdoor** measurement with 10 robots (as measured per device)

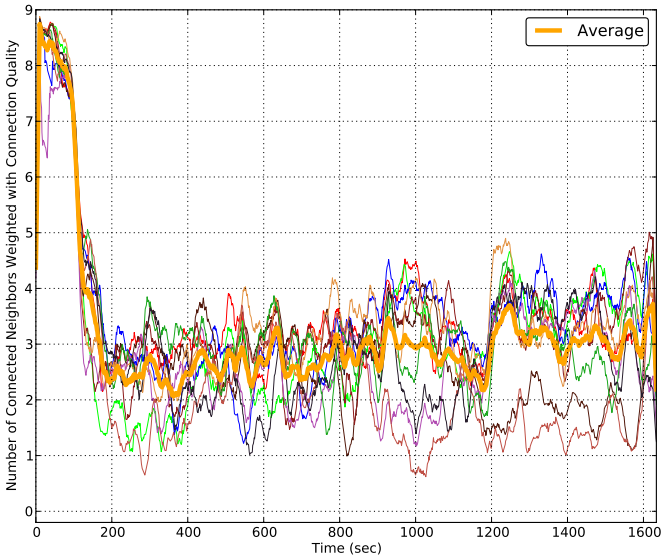

Figure 4. Number of direct neighbors during an **indoor** measurement with 10 robots (as measured per device)

it even drops to 0 for certain devices, meaning they were out of range of all other phones. The values are fluctuating heavily, which means that connections are frequently lost and re-established. The average number of neighbors was 1.55, and the average duration of a connection was 6.8 seconds.

For indoor measurements, we used a sizable canteen area at the university campus. This area is half as large as the parking lot, and numerous obstacles (e.g. tables, chairs, walls, stairs) are present. In Figure 4, we plotted the same indicator for a 25-minute-long measurement at this location. The robots also started from the same position and followed a random path. The graphs are also fluctuating, although they are relatively more stable. The average number of direct neighbors was 3.18, and the average duration of a connection was 49.6 seconds. This indicates a higher level of connectivity indoors, which can be explained by the signal reflections from the walls. Note that the values never drop to 0 in this case.

Several hundred data sets were also transferred to all participating devices during these measurements. Our application recorded various events and statistics to log files as described in Section V. In addition, the application was extended to use the Control API of the Nokia Energy Profiler to programmatically monitor (and record) the energy consumption of the mobile phones. Previous experiments confirmed that these energy readings are accurate.

An important metric of data dissemination is the completion time which is measured from the starting moment at the *source* until the data set is fully received by a given sink node. For each test run, we have sorted the results for the 9 receivers in the order of finishing. In Figure 5 and 6, the average completion times are plotted for the proposed and the reference strategy for outdoor and indoor measurements, respectively (the error bars represent one standard deviation). The first observation is that the proposed strategy performs similarly in both

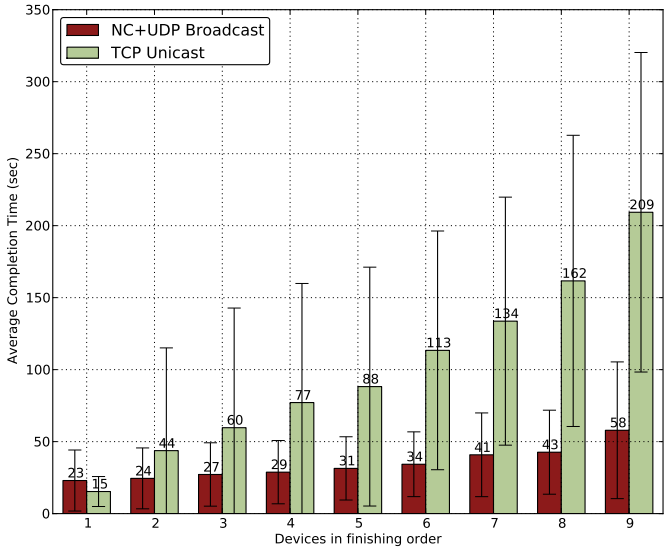

Figure 5. Comparison of completion times for the 9 receivers **outdoors**

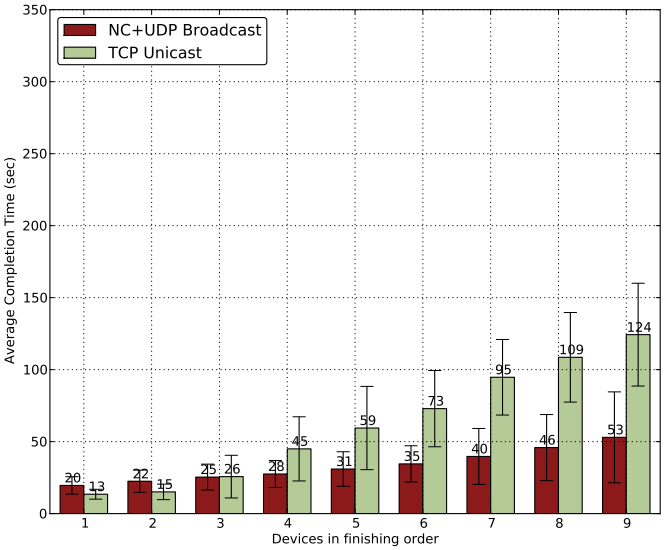

Figure 6. Comparison of completion times for the 9 receivers **indoors**

environments, whereas the reference strategy is significantly slower outdoors. If we consider the last (9^{th}) finishers to compare the two strategies, we observe that the completion times for TCP are 2.4 times longer indoors and 3.6 times longer outdoors. This can be explained by the high management overhead associated with TCP, since connections are frequently lost and must be re-established. We also see a higher variance outdoors for both strategies due to the more dynamic network topology.

We have summarized the collected statistics about the data dissemination process in Table II. Results from 10 devices and hundreds of test runs were averaged, and the presented values refer to a single device and single data set.

TABLE II. DATA DISSEMINATION STATISTICS

Metric	**NC+UDP Broadcast**		**TCP Unicast**	
	Outdoors	***Indoors***	***Outdoors***	***Indoors***
Size of data set [bytes]	4 160 000			
Average completion time (on the last device) [sec]	58.01	53.22	208.76	124.15
Average goodput (per device) [KB/sec]	112.4	118.09	37.34	60.92
Average energy consumption (per device) [J]	55.04	43.55	217.13	141.77
Normalized energy consumption [µJ/bit]	1.65	1.31	6.52	4.26

The best goodput (the rate of receiving useful data) was achieved indoors by the proposed strategy. Since more time means more energy, we see large differences in the average energy consumption values. With the TCP-based strategy, the devices consumed almost 4 times more energy outdoors and 3.25 times more indoors in comparison with the corresponding values for the proposed strategy. In both cases, the connectivity was also lower outdoors, possibly resulting in more retransmissions and higher energy consumption.

VIII. CONCLUSION

In this paper we have introduced a strategy to facilitate the dissemination of multimedia content in a dynamic network of mobile devices. A testbed consisting of 10 autonomous robots was used to provide a reasonably high degree of mobility in our experiments. We considered a scenario where the mobile devices are mounted on moving robots, and initially one device possesses the entire data set that should be conveyed to all other devices. We proposed a strategy that can detect other devices in range, perform the actual data transfer, and signal the completion of the dissemination process. This strategy was implemented on Nokia phones using the Qt cross-platform application framework. For comparison, we also introduced a reference strategy that uses standard TCP connections. We presented measurement results collected during numerous experiments in both indoor and outdoor environments. We also analyzed the connectivity among the mobile devices at these locations. Results show that the proposed strategy outperforms the reference strategy 3 or 4 times in terms of completion times and energy consumption. Moreover, the proposed strategy performs similarly in both environments.

ACKNOWLEDGMENTS

This work was partially financed by the CONE project (Grant No. 09-066549/FTP) granted by the Danish Ministry of Science, Technology and Innovation as well as by the collaboration with Renesas Mobile throughout the NOCE project.

REFERENCES

[1] S. Alagar, S. Venkatesan, and J. R. Cleveland, "Reliable broadcast in mobile wireless networks," in *IEEE Military Communications Conference, 1995. MILCOM'95, Conference Record*, 1995, vol. 1.

[2] E. Pagani and G. P. Rossi, "Providing reliable and fault tolerant broadcast delivery in mobile ad-hoc networks," *Mobile Networks and Applications*, vol. 4, no. 3, pp. 175–192, 1999.

[3] C. S. Hsu, Y. C. Tseng, and J. P. Sheu, "An efficient reliable broadcasting protocol for wireless mobile ad hoc networks," *Ad Hoc Networks*, vol. 5, no. 3, pp. 299–312, Apr. 2007.

[4] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204-1216, Jul. 2000.

[5] C. Fragouli, J.-Y. Le Boudec, and J. Widmer, "Network coding: an instant primer," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 63–68, 2006.

[6] R. Matsumoto, "Construction algorithm for network error-correcting codes attaining the Singleton bound," *IEICE Trans. Fundamentals*, vol. E90-A, no. 9, pp. 1729-1735, Sep. 2007.

[7] S. Katti, H. Rahul, W. Hu, D. Katabi, M. Médard, and J. Crowcroft, "XORs in the air: practical wireless network coding," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 243–254, 2006.

[8] J.-S. Park, M. Gerla, D. S. Lun, Y. Yi, and M. Médard, "Codecast: a network-coding-based ad hoc multicast protocol," *IEEE Wireless Communications*, vol. 13, no. 5, pp. 76-81, Oct. 2006.

[9] J. Heide, M. Pedersen, F. H. P. Fitzek, and T. Larsen, "Network Coding for Mobile Devices - Systematic Binary Random Rateless Codes," in *Workshop on Cooperative Mobile Networks 2009 - ICC09*, 2009.

[10] M. V. Pedersen, J. Heide, F. H. P. Fitzek, and T. Larsen, "PictureViewer - A Mobile Application using Network Coding," in *European Wireless 2009*, Aalborg, Denmark, 2009.

[11] L. Militano, F. H. P. Fitzek, A. Iera, and A. Molinaro, "A Genetic Algorithm for Source Election in Cooperative Clusters Implementing Network Coding," in *IEEE International Conference on Communications (ICC 2010) - CoCoNet Workshop*, 2010.

[12] Z. J. Haas and T. C. Chen, "Cluster-based Cooperative Communication with Network Coding in Wireless Networks," in *The 2010 Military Communications Conference - Unclassified Program*, San Jose, California, USA, 2010, pp. 596-603.

[13] P. Vingelmann, M. V. Pedersen, F. H. P. Fitzek, and J. Heide, "On-the-fly Packet Error Recovery in a Cooperative Cluster of Mobile Devices," in *IEEE GLOBal COMmunication conference, exhibition & industry forum (GLOBECOM) - Next Generation Networking Symposium*, Houston, Texas, USA, 2011.

[14] D. E. Lucani, F. H. P. Fitzek, M. Medard, and M. Stojanovic, "Network Coding For Data Dissemination: It Is Not What You Know, But What Your Neighbors Know," in *RAWNET/WNC3 2009*, 2009.

[15] P. Vingelmann, F. H. P. Fitzek, and D. E. Lucani, "Application-level Data Dissemination in Multi-hop Wireless Networks," in *IEEE International Conference on Communications (ICC 2010) - CoCoNet Workshop*, 2010.

[16] J. Reich, V. Misra, and D. Rubenstein, "Roomba MADNeT: a Mobile Ad-hoc Delay Tolerant Network Testbed," in *MC2R: Mobile Computing and Communications Review*, 2008.