Application-level Data Dissemination in Multi-hop Wireless Networks

Péter Vingelmann
Budapest University of Technology and Economics
Dept. of Automation and Applied Informatics
Budapest, Hungary

Frank H. P. Fitzek
Aalborg University
Dept. of Electronic Systems
Aalborg, Denmark

Daniel E. Lucani
MIT
Research Laboratory of Electronics
Cambridge, Massachusetts, USA

***Abstract*— We investigate different schemes for data dissemination in multi-hop ad-hoc networks using network coding. We study the performance of these schemes in terms of the completion time when a set of data packets must be disseminated from a single source to all nodes in a multi-hop network, i.e. a network where at least one node is several hops away from the source. Therefore some network nodes must relay information to other nodes that are farther away from the source. In this setting, a relay node does not send a particular data packet, but a linear combination of the packets that it has previously received. The selection of such relays has a significant impact on performance. We present a graphical simulator based on OpenGL that allows to study performance and illustrate the status of network nodes in real time during the dissemination of an image file. Features of real-life ad-hoc networks such as packet losses and collisions are taken into consideration in our simulator. Numerical results are presented for simple linear meshed networks and for arbitrary topologies. Results indicate that schemes promoting parallel non-interfering transmissions complete the dissemination process faster.**

I. Introduction

Data dissemination in wireless ad-hoc networks can be used to share data, e.g. pictures, sound, video, among a set of source nodes and a set of receivers or sink nodes. Data may be a large file or a continuous stream which is divided into multiple batches of packets as necessary. An interesting objective is to find a trade-off between completion time, energy consumption and protocol overhead. Since all receivers are requesting the same data, the broadcast nature of the wireless channel allows for an efficient delivery of innovative information. That is, with a single transmission possibly more than one node and potentially all nodes in range will increase their knowledge of the original data set.

Wireless ad-hoc networks typically have 1) packet losses, i.e. an uncertainty of reception of transmitted packets, and 2) information delivered through possibly different paths or routes. Consequently different nodes may require entirely different data packets to increase their current knowledge. A shrewd way to maximize the impact of each transmission is by applying network coding [1], [3]. Transmitting linear combinations of data packets, instead of just forwarding information, relaxes the objective of each node. A network node is no longer required to gather all data packets one-by-one, instead it only has to receive enough independent linear combinations to decode the entire data set. This relaxation allows us to simultaneously benefit multiple nodes in a single transmission, because the transmitter can send a linear combination instead of choosing a specific packet, which may benefit only one node in range.

The authors in [8] proposed an efficient solution for mobile devices considering a single-hop network (i.e. all nodes are within range of the source). A simple application based on this solution was presented in [9]. However, multi-hop networks, where some nodes are *not* directly reachable by the source, require different techniques to deliver data to sinks that are farther away from the source. This is only possible if some nodes, called relays, propagate the received data to other nodes. A relay node in the context of this work is both interested in receiving the original data set and helping in the dissemination process by generating and transmitting encoded packets.

Note that a single packet can be delivered to every node using simple flooding, but precautions have to be taken to avoid duplicate deliveries and infinite loops. However, using (naive) flooding is not a feasible approach for disseminating a large set of packets: it would cause a lot of contention, collision, and congestion in the system. Reference [6] called this the *broadcast storm* problem. There are numerous references, e.g. [2], [10], [7], [4], dealing with efficient reliable broadcasting protocols for wireless ad-hoc networks. They have one thing in common: they intend to ensure reliable delivery of individual packets to all nodes. This approach can be extremely costly, because all protocols require a significant overhead (e.g. acknowledgement and notification packets) to be generated for every single packet.

The authors in [5] investigated the same scenario and compared the performance of several network coding schemes. They proposed a greedy algorithm that tries to maximize the impact on the network at each time slot, i.e. maximize the average number of nodes that will benefit from a transmission. The scheme based on this greedy algorithm also tries to benefit transmissions that will allow potential parallel non-interfering transmissions in the future. The scheme is designed for the MAC level, and it requires a centralized controller with a perfect knowledge of the information at each of the nodes.

In this paper, we devise new schemes which are instantly deployable in any modern wireless network to disseminate data to all nodes in the network. We developed a new network

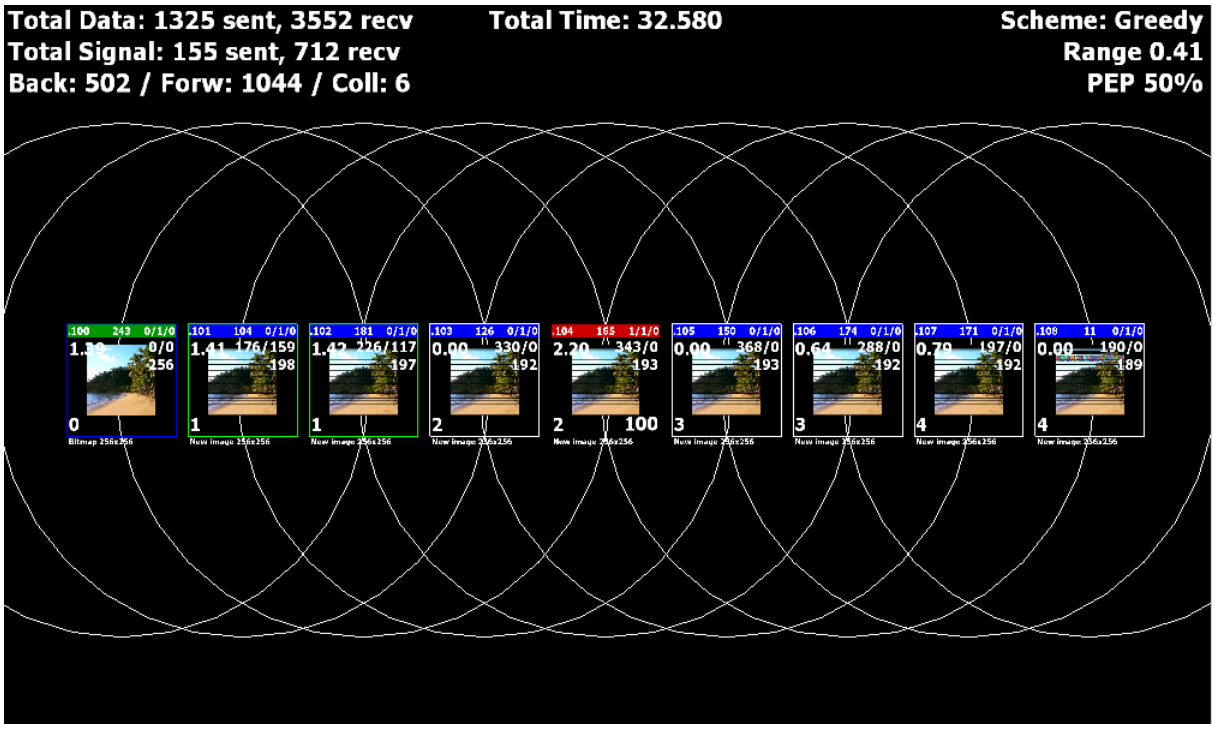

Fig. 1. Screenshot of the simulator showing 9 nodes. The leftmost node was chosen as the initial source. Nodes in green are complete. Nodes in blue are in progress. Nodes in red are transmitting at the moment.

simulator to test the performance of several schemes before conducting experiments with real mobile devices. Using a simulator allows us to determine the performance of the new schemes compared to a modified version of the aforementioned greedy scheme, which is used as a reference in terms of completion times. Similarly to [5], we first focus on linear networks, i.e. network nodes deployed in a straight line. Later we compare the performance of our schemes in arbitrary network topologies.

This paper is organized as follows. Section II introduces our network simulator. Section III describes the operation of the implemented schemes. Sections IV and V present simulation results for linear networks and arbitrary topologies. Section VI concludes this paper.

II. Simulator

The simulator's design is based on a few assumptions that are commonly known as the "Flat Earth" model:

1) The world is flat.
2) A radio's transmission area is circular.
3) All radios have equal range.
4) If I can hear you, you can hear me (symmetry).
5) Signal strength is a simple function of distance.
6) All nodes are fixed.
7) There is no external interference.
8) There are no obstacles.

The simulator has a visual frontend which was developed in OpenGL (see Figure 1). Nodes are depicted as boxes, and certain interactions are allowed to manipulate these nodes. They can be moved or deleted, and new nodes can be inserted interactively. Complete networks can be saved or loaded on demand. A certain node is selected as the initial source by double-clicking it. It loads up and renders a specified image whose lines will form the data packets that we intend to disseminate. For example, an RGBA image of dimensions 256×256 has 256 scanlines, which translates into 256 packets. The size of each scanline in memory is $256 \times 4 = 1024$ bytes, which will be the size of a single packet (not including headers).

There can be only one source at a time. When a certain node is selected to be the source, the memory of the other nodes is wiped (they start with a blank image), and the source will begin sending out data packets according to the chosen scheme and other simulation parameters. All of these settings are also adjustable in the GUI. Time is assumed to be continuous and the nodes are constantly making independent choices to send or to remain silent. Our intent is to simulate the behavior of UDP broadcast transmissions with appropriate delays, collisions and packet losses.

A predefined channel capacity determines the maximum sending rates of the nodes. Collisions will occur if they try to exceed this limitation. Naturally, there is a small delay between sending and receiving a packet. Packet losses are based on the Packet Error Probability (PEP) variable, which can be adjusted run-time, and on the distance to the recipient node. The PEP value is used at the maximum transmission range, but nodes closer to the sender experience lower loss probabilities starting from $0.5 \cdot PEP$ at 0 range, incresing linearly with

the distance. Losses are independent and memoryless, i.e. a sent packet is lost at each receiver independently of losses at other receivers, and independently of past transmissions. Collisions are also detected on the receiver side, overlapping transmissions are considered to be corrupted, therefore they are automatically dropped.

If an uncoded packet is received, its data will be inserted into the image instance at the receiver, i.e. a new line becomes visible. Nodes can also use network coding over GF(2) (as described in [8]) or over GF(2^8) (as described in [11]). Based on the current scheme, certain packets can be encoded if necessary. Coded packets will be decoded by the receiver, whose image instance will be updated to reflect changes in the decoding matrix. A separate signaling channel exists which is free from packet losses and collisions, but transmissions are also delayed. Nodes should use this channel as little as possible, and no data packets can be sent over it.

Note the simulator provides a simplified model of wireless networks. We consider some important characteristics, e.g. packet losses, collisions, but we ignore other effects present in real-life wireless networks.

III. Schemes

Since we consider multi-hop networks, the first problem to address is how to select relay nodes that will forward data packets to nodes that are farther away from the source. Let us divide network nodes into hop levels: the 1^{st} hop level consists of nodes directly reachable from the source, nodes at the 2^{nd} hop level can be reached with 2 hops (through a relaying node), and so on. Hop levels can be quickly determined with a hop counter variable upon starting a new dissemination session. In Figure 1 the resulting hop levels are visible at the bottom left corner of each node's box. Nodes can also keep a record of their direct neighbors and propagate this list to them. Based on this information, certain nodes can be selected as relays. If Node A can communicate with Node B at the next hop level, and no other nodes from Node A's hop level can reach Node B, then Node A must be a relay. In this case Node A covers Node B. If Node B is covered by multiple nodes from the previous hop level, it is enough to select one of these. Moreover, Node B might be indirectly covered by Node C at the same hop level, if Node C is connected to another relay at Node A's hop level. Consequently, Node A should not necessarily be a relay. It is imperative to select as few relays as possible to minimize network load and energy consumption. Without prior knowledge of the network topology, individual nodes should be able to select themselves as relays, i.e. we need a self-configuring multi-hop network.

Four different schemes have been implemented in the simulator, these are explained in the following. The first two schemes use the relay selection mechanism described above, but they activate the relays at different times.

- **Progressive Base Station**: The initial source broadcasts all data packets, and ensures that all relays at the first hop level receive all those packets. Packet losses are corrected using *systematic network coding* as described in [8], i.e. the source transmits random linear combinations of all packets until all its neighbor nodes are ready. After this, the selected relays (at the first hop level) will become active and continue to disseminate data packets to all nodes at the next hop level, and it goes on until we reach the last hop level. Note that relays follow the same approach as the initial source. A limited number of acknowledgements are used in this scheme to indicate when a node could fully decode all data packets. Completion time is expected to be a linear function of the number of hop levels.
- **Scheduled**: This scheme is similar to the previous one, but data packets are transmitted in *short bursts* (e.g. 32 packets) from the source to relays at the first hop level. When a relay fully receives a burst, it begins broadcasting that burst towards the next hop level. The sender stops automatically when it senses a transmission from a higher hop-level, i.e. we use implicit acknowledgements only. In the Scheduled scheme, data is spreading in *waves*. The initial source starts the next burst (i.e. next wave) after waiting at least 2 burst intervals. Packet losses within a burst are corrected right away by the sender with some encoded packets to ensure that data packets can spread efficiently towards higher hop levels as soon as possible. This protocol leverages the power of *parallel non-interfering transmissions* in combination with a *strict schedule*. Completion time is *not* expected to be a linear function of the number of hop levels.
- **Greedy**: This scheme is loosely based on the Greater Impact/Greedy algorithm proposed in [5]. This is the only scheme where a *centralized controller* manages the nodes' actions. Consequently, it is not applicable in a real-life network. The controller periodically selects the node with the most benefit points to transmit. A node gets benefit points for each neighbor to which it can send an innovative packet. Neighbors with less knowledge are worth more benefit points which is multiplied by the probability of a successful transmission to that neighbor. *Parallel non-interfering transmissions* are also possible here, since multiple nodes are selected to transmit if their transmissions do not cause collisions. In accordance with the formulae presented in [5], we expect a significant gain compared to the Progressive scheme.
- **Naive Broadcast**: When a node receives a new (or innovative) packet, it simply retransmits it after a short delay. Nodes at the highest hop level (i.e. leaf nodes) will *not* retransmit any packets. This scheme is similar to the default broadcast solution in the upcoming IEEE 802.11s standard. As we are dealing with multiple packets, a couple of adjustments must be made. The initial source is transmitting at full rate. If a node decodes all the data, it begins transmitting encoded packets at a variable rate. Explicit acknowledgements are not implemented, a node simply stops when all its neighbors are ready. A lot of collisions are expected here due to the *broadcast storm* problem.

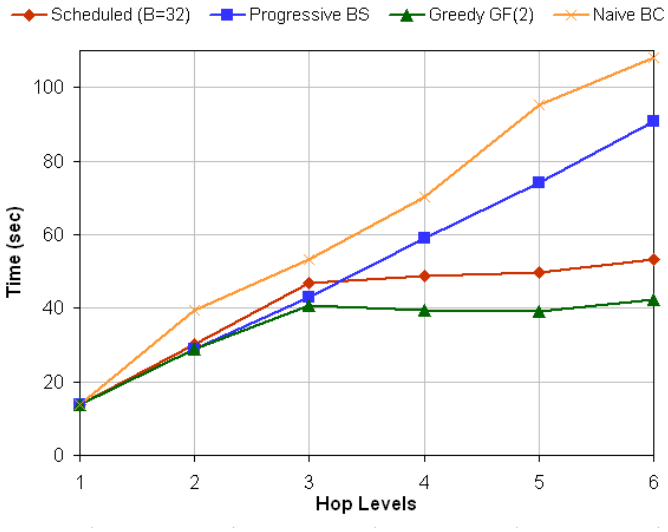

Fig. 2. Completion times over increasing line length ($M = 256, B = 32, N = 2, P = 30\%$, varying H)

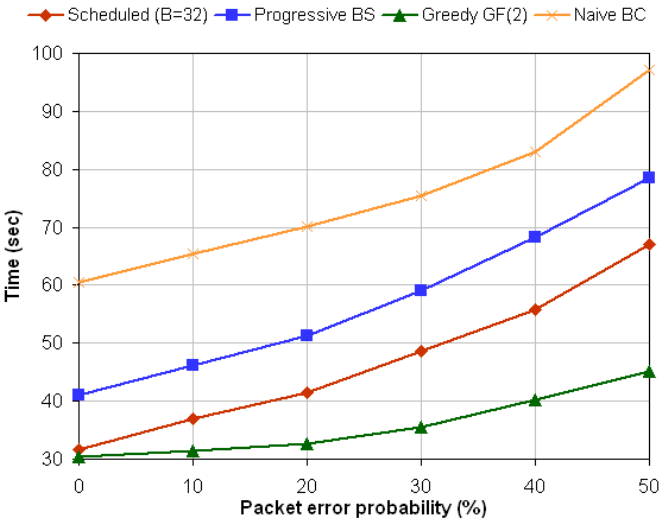

Fig. 3. Completion times over increasing Packet Error Probability. ($M = 256, B = 32, N = 2, H = 4$, varying P)

IV. Linear networks

Linear networks are toy examples where all nodes are along a straight line (e.g. on Figure 1). In addition, we also assume that every hop level consists of the same number of nodes. We begin our simulations with these networks to get a clear view about the performance of our schemes. Let us declare the following symbols for describing network topology and other parameters:

- H = number of hop levels
- N = number of nodes per hop level
- K = total number of nodes ($H \cdot N$)
- M = number of packets to be disseminated
- B = burst size for the Scheduled scheme
- P = packet error probability

The simulator can be used to measure completion times. A dissemination session is completed when all nodes have all the data packets, i.e. the full image is decoded. In every session an RGBA image of dimensions 256×256 will be disseminated. It means 256 packets, each containing 1024 bytes of raw image data. Maximum sending rate is set to 25.6 packets/sec, so that it takes exactly 10 seconds to transmit all packets to a direct neighbor if $P = 0\%$. By default, GF(2) is used for network coding. GF(2^8) results can also be included, but this is always explicitly stated.

First of all, we examine how completion times change with increasing line length (see Figure 2). In this simulation P is fixed to a typical value of 30%, we start with a source and two direct neighbors in a line, then we gradually increase the number of hop levels by adding two more nodes to the end of the line.

Note that results presented here are averages of several measurements, except for Naive Broadcast where the lowest measured value is shown, since this scheme may fail to complete. Completion times for the Progressive Base Station scheme increase linearly with the line length, as expected. However, the Greedy and Scheduled schemes start with similar performance to the Progressive Base Station scheme, but after 3 hop levels their completion times only slightly increase. This is because parallel non-interfering transmissions are present in these schemes, and two nodes must be separated by at least 3 hop levels if they intend to transmit simultaneously without causing a collision. Results have a relatively low standard deviation for the first 3 schemes, but the completion times for Naive Broadcast are highly erratic. Even though we chose topologies with only 2 nodes per hop level to minimize interference, thousands of collisions are registered at this scheme.

It is also interesting to test how schemes react to increasing PEP values (see Figure 3). Notice that completion times for the Scheduled and Progressive Base Station schemes follow the expected $T_0 \cdot 1/(1-P)$ value where T_0 is the completion time at $P = 0\%$. Decay for the Greedy scheme is much better, because the probability of successful transmissions to different nodes is taken into account in this scheme, therefore it will try to benefit closer nodes with lower PEP values. In general, the Greedy scheme is much more flexible in comparison with Scheduled and Progressive BS which are restricted to use fixed relays (after they are selected).

The simulator is also capable of producing time diagrams depicting the knowledge of each network node. The dissemination progress is clearly visible on these diagrams. We chose to make a time diagram for the network shown in Figure 1. The differences between certain schemes are easy to understand by looking at Figure 4. Each row represents a specific node. Note that nodes at the bottom are closer to the initial source. Progressive Base Station starts by transmitting the entire data set to nodes at the first hop level, which prevents any parallel transmissions in the system. On the contrary, the Greedy

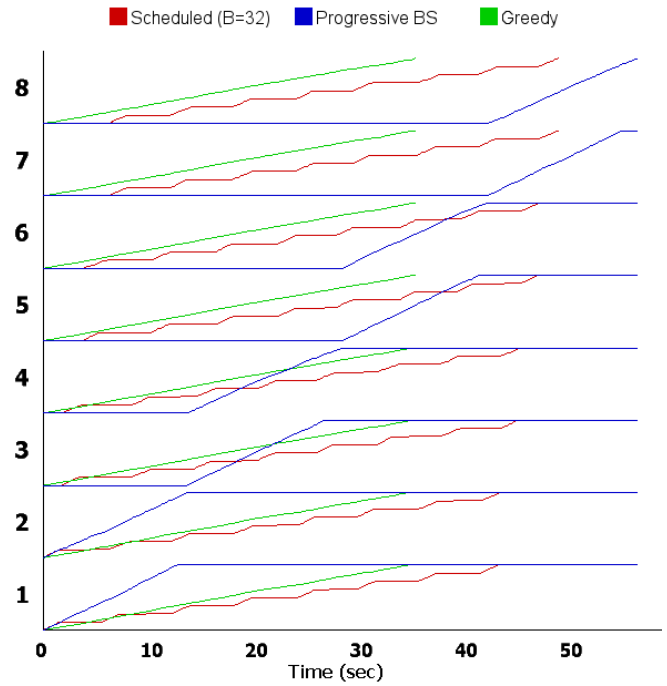

Fig. 4. Time diagram depicting the knowledge of individual nodes. ($M = 256, B = 32, N = 2, H = 4, P = 30\%$)

scheme tries to continuously increase the knowledge of every node (favoring nodes with less knowledge), thus we see lines with a steady slope. Innovative packets will quickly spread towards the last node in the line, thereby facilitating parallel transmissions. Diagrams for the Scheduled scheme indicate that this is a mixture of the other two. If we decrease the burst size to 8 or 4, the lines will get closer to those of the Greedy scheme. However, this only works when there are no packet losses ($P = 0\%$), since a smaller burst size would imply network coding with a really small generation size, which would require more transmissions per packet. On the other hand, we can approximate the behavior of the Progressive Base Station scheme by increasing the burst size significantly. According to our measurements, the burst size of 32 gives the best results, it seems to be a good trade-off between the characteristics of the other two schemes.

V. Arbitrary Networks

In this section we consider an arbitrary network topology consisting of 33 randomly placed nodes. Some nodes were added manually to establish a connected network. The topology is shown in Figure 5.

We consider two different starting nodes in our simulations, marked with blue and purple frames in Figure 5, to get a better understanding about the performance of our schemes. We chose a node in the middle and one in a long branch. Figures 6 and 7 present numerical results obtained in the two sets of measurements with increasing PEP values.

Notice that the charts are quite similar to Figure 3, but the Scheduled scheme outperforms Greedy in this setting. The

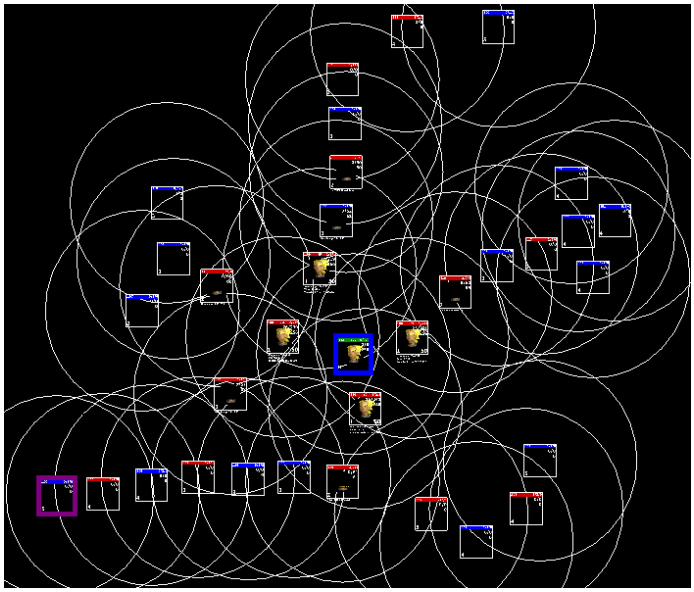

Fig. 5. The chosen network topology. Nodes marked with blue and purple frames are selected as initial sources.

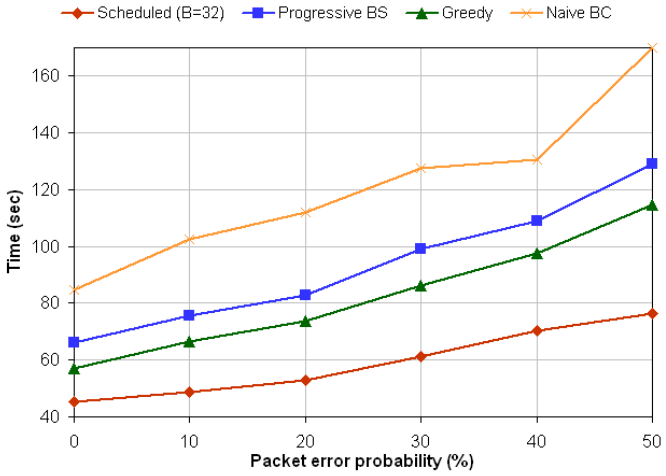

Fig. 6. Completion times when starting from the middle node ($M = 256, B = 32, K = 33$, varying P)

performance of Naive Broadcast is even worse in this arbitrary network. More than 10000 collisions are typical during a single session, and many times it is unable to complete, therefore the lowest measured value is shown, not the average.

Although the Greedy scheme performs well in many cases, we know that it is a suboptimal strategy in general. As it was discussed in [5], the optimal solution would be to find the best sequence of transmissions, which is a difficult scheduling problem. In the given topology, the initial source (marked with blue frame) will get the most points for benefiting 4 neighbors. Consequently, no other node can transmit until these 4 neighbors are ready, i.e. information is not spreading towards higher hop levels. Later on, another problem may arise: branches with fewer nodes can be neglected due to lower

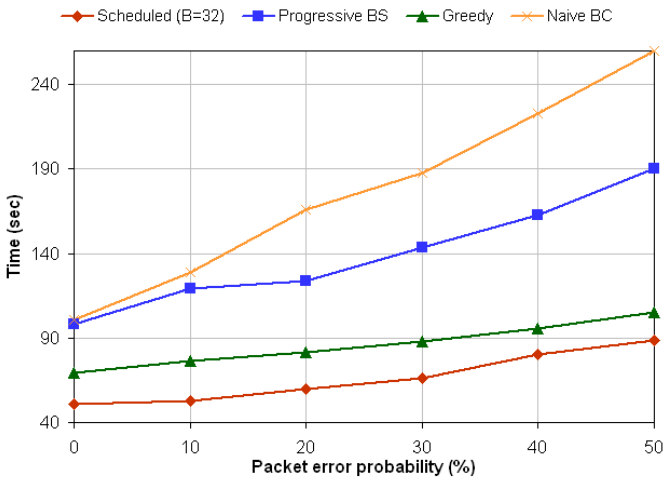

Fig. 7. Completion times when starting from the leftmost node ($M = 256, B = 32, K = 33$, varying P)

benefit points. In order to avoid these issues, we modified the Greedy scheme to always favor nodes at higher hop levels in benefit point calculations. Even this mechanism did not show a considerable improvement.

On the other hand, the predefined behavior in the Scheduled scheme always forces the waves to happen in all directions. Spatial diversity is better exploited here, the dissemination process cannot get stuck as it does with Greedy. In addition, branches with fewer nodes are considered to be equally to important. Advantages of the Scheduled scheme are derived from the strict, pre-programmed schedule and the proper selection of relay nodes.

An important difference between starting from the middle and the leftmost node is that we get 5 and 9 hop levels, respectively. It means that data packets must traverse a longer chain of nodes, which has a severe negative impact on the performance of the Naive Broadcast and Progressive Base Station schemes (as shown on Figure 7). On the contrary, completion times do not increase significantly for Scheduled and Greedy, because a longer chain also means more opportunities for parallel transmissions.

VI. Conclusion

We have introduced a graphical simulator that allows us to design and evaluate schemes for data dissemination. We have described four different schemes and presented numerical results to compare their performance under various circumstances. The Scheduled and Greedy schemes perform quite well in most cases due to the high number of parallel non-interfering transmissions. The Progressive Base Station scheme works in a sequential manner, thereby its performance will deteriorate as the network size increases. Our last scheme, Naive Broadcast suffers from the broadcast storm problem, consequently it is outperformed by the other schemes, especially in larger networks.

In this paper we only considered static networks, so the next logical step is to experiment with moving nodes. Nevertheless, designing schemes for mobility requires a different approach, since the concept of hop levels is no longer valid if neighbor nodes might move out of range. Other features of real-life networks, such as obstacles or nodes with variable range and bandwidth, should also be taken into consideration in the future.

Naturally, we intend to conduct experiments with state-of-the-art mobile devices. Our decision to focus on an application-level implementation has its advantages. We have designed our schemes (except for Greedy) to be readily deployable in any wireless network that supports UDP broadcast, no tampering with the MAC layer is required.

References

[1] R. Ahlswede, Ning Cai, S.-Y.R. Li, and R.W. Yeung. Network information flow. *Information Theory, IEEE Transactions on*, 46(4):1204–1216, Jul 2000.

[2] S. Alagar, S. Venkatesan, and JR Cleveland. Reliable broadcast in mobile wireless networks. In *IEEE Military Communications Conference, 1995. MILCOM'95, Conference Record*, volume 1, 1995.

[3] Christina Fragouli, Jean-Yves Le Boudec, and Jörg Widmer. Network coding: an instant primer. *SIGCOMM Comput. Commun. Rev.*, 36(1):63–68, 2006.

[4] C.S. Hsu, Y.C. Tseng, and J.P. Sheu. An efficient reliable broadcasting protocol for wireless mobile ad hoc networks. *Ad Hoc Networks*, 5(3):299–312, 2007.

[5] 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*, June 2009.

[6] Sze-Yao Ni, Yu-Chee Tseng, Yuh-Shyan Chen, and Jang-Ping Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, New York, NY, USA, 1999. ACM.

[7] E. Pagani and G.P. Rossi. Providing reliable and fault tolerant broadcast delivery in mobile ad-hoc networks. *Mobile Networks and Applications*, 4(3):175–192, 1999.

[8] M. Pedersen, J. Heide, 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*. IEEE, June 2009.

[9] 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, May 2009.

[10] J. Tourillhes. Robust broadcast: improving the reliability of broadcast transmissions on CSMA/CA. *HP LABORATORIES TECHNICAL REPORT HPL*, 1998.

[11] P. Vingelmann, P. Zanaty, F.H.P.Fitzek, and H. Charaf. Implementation of random linear network coding on opengl-enabled graphics cards. In *European Wireless 2009*, Aalborg, Denmark, May 2009.