Urgent message dissemination with differentiation in intermittently connected networks

Mikael Asplund, Simin Nadjm-Tehrani
Department of Computer and Information Science
Linköping University

This technical report contains a preliminary version of the book chapter titled "Rapid selection and dissemination of urgent messages over delay-tolerant networks (DTNs)", published online at:
`http://dx.doi.org/10.1533/9780857098467.2.187`

When referring to this work please use the following citation:

M. Asplund and S. Nadjm-Tehrani, *Rapid selection and dissemination of urgent messages over delay-tolerant networks (DTNs)* chapter in Advances in delay-tolerant networks (DTNs), Architecture and Enhanced Performance. Woodhead Publishing Series in Electronic and Optical Materials, Elsevier, 2015. doi:10.1533/9780857098467.2.187

Abstract

Today many new applications are emerging that take advantage of wireless communication in handheld and embedded devices. Some of these emerging applications, such as information sharing in vehicular systems, have strong requirements on timely message dissemination, even if the network is not always 100% connected. In this chapter we discuss message differentiation mechanisms that can be used in intermittently connected networks to improve delivery and latency properties when messages have a limited time to live in the network. We present a simulation-based study on a large-scale vehicular scenario comparing different prioritisiation mechanisms for a partition tolerant manycast protocol. We show that negative effects of overloads can be significantly reduced by using information within the message about how far it has spread and how much time is remaining.

1 Introduction

The number of connected devices in the world is increasing at an enormous rate. A whole new range of products and services are starting to appear that take advantage of high speed wireless communication being available anywhere at anytime. Cellular technology has led the way with very good coverage and reasonable bandwidth capabilities. However, these centralised systems are also vulnerable to overloads, unforeseen events (e.g., extreme weather), and there are still many rural areas with poor coverage. This motivates the need for alternative forms of communication where devices communicate directly with

each other, through wireless mesh networks [1], or other hybrid solutions. We are slowly starting to see the emergence of such systems with the WifiDirect standard for household devices and the 802.11p standard for vehicular communication.

Dissemination of information in such wireless ad hoc networks requires coping with intermittent connectivity and may at times need to cope with high network traffic load. Moreover, for many potential applications such as vehicular communication with warnings about hazards on the road (e.g., ice, fallen trees and flooded areas) it is imperative that messages are disseminated within a short and predictable time frame. While there is a rich body of work on how to best make use of the limited resources in delay-tolerant networks (especially for unicast routing), there are not as many works that focus on urgent one-to-many message dissemination in this context. By urgent we mean that messages should be delivered within a number of seconds[1] rather than within minutes or even hours which is not an uncommon assumption in research on delay-tolerant networks.

In this chapter we explore the provision of timely dissemination of information in intermittently connected networks. We provide a brief overview of the existing research in the area as well as a more in-depth study on how to achieve a higher level of predictability of the message dissemination latency. Specifically, we explore how different prioritisation mechanisms can be used to increase the delivery of urgent messages and thereby also increase the global system performance. We focus on the case where the system is subject to a traffic overload which would normally reduce the system performance significantly.

We build on Random Walk Gossip (RWG) which is a manycast protocol for partition-tolerant networks tailored for disaster area networks. In its original design RWG treats all messages equally until the time when their time to live (TTL) counter expires. In this work we extend the design of the protocol by differentiating messages based on their deadline and progress so far. Due to the unique way RWG works, we are able to categorise the level of lateness of a message ranging from very early to very late, and use this categorisation when setting dynamic priorities to provide the best possible use of the limited resources. In contrast to several other works, we do not need to collect system statistics in order to make differentiation decisions. Instead, the necessary information is immediately accessible from the message header.

We study four different prioritisation policies, including randomly ordering messages which we consider as our baseline. The other three policies are Earliest Deadline First (EDF), which is well known from task scheduling theory, Least Informed First (LIF), which prioritises messages with little progress, and Least Slack First, which is also a standard task scheduling algorithm but which we have adapted to fit RWG.

In order to evaluate these prioritisation policies we have conducted a simulation-based experimental evaluation using highly realistic traces from a vehicular scenario. Starting from traces of a large-scale realistic simulation of car movements in the city of Cologne, we have extracted a subset of traces which correspond to one fifth of all cars within 3km from the city centre and used this in our simulations. The results show that message differentiation can significantly increase

[1] there are some applications that require messages to be delivered within milliseconds, but such systems can hardly afford to be intermittently connected at any time

the amount of messages that the system can handle without being severely overloaded. Moreover we found LSF to provide the overall best results, but that EDF would still outperform the other policies when the system is very overloaded.

The remainder of this chapter is organised as follows. Section 2 gives an overview of the state of the art in one-to-many communication in resource constrained environments with intermittent connectivity. Section 3 presents the Random Walk Gossip protocol which we have extended with message prioritisation mechanisms as explained in Section 4. In Section 5 we present a simulation-based experimental evaluation of the differentiation policies. Section 6 concludes the chapter.

2 One-to-many communication in resource constrained environments

Timely message dissemination in challenged networks with intermittent connectivity and varying load is difficult at best and there is a rich body of research concerning different aspects of this problem. In this section we present a brief overview of the state of the art in this area. We begin with a more general look at one-to-many message dissemination in intermittently connected networks, proceed with a summary of the most relevant differentiation mechanisms and finally present a selection of DTN protocols where message differentiation (MD) is a key element.

2.1 DTN multicast/broadcast

As wireless networks were entering our everyday lives in the nineties, researchers began to wonder if and how it would be possible to deliver messages in networks with extremely poor connectivity. This gave rise to what is commonly referred to as disruption- or delay-tolerant networks (DTN). In these networks messages that cannot be forwarded due to lack of reachable neighbours are stored, to be forwarded at future encounters, giving rise to the notion of *custodian*. A significant portion of the research in DTNs has focused on unicast routing where each message has one sender and one destination. Other forms of communication such as one-to-many or many-to-one have often been considered as just extensions of unicast routing. However, in many emerging application areas such as vehicular networks and disaster area networks, *information sharing* is a crucial mechanism for which one-to-many communication is very suitable.

There are several variants of one-to-many communication, including *broadcast* where a message should reach all nodes in the network, *multicast* [23, 14] where a message should reach a designated set of nodes, *geocast* [4] where a message should reach all nodes within a given area, and *manycast* [5] where a message should reach a minimal *number* of nodes.

One of the early proposals for partition-tolerant multicasts is the Hyper-flooding protocol by Obraczka and Tsudik [15]. The idea is fairly straightforward. Each message keeps a hop count and a TTL and each node keeps track of its neighbours. When a node discovers a new neighbour, packets that have not expired in TTL or hop count are propagated. Later, Viswanath and Obraczka [21] improved this algorithm by adapting the flooding policy to the

estimated network conditions. The worse the connectivity, the more aggressive flooding policy.

Continuing along the same lines, Khelil et al. [11] use a two-stage approach called hypergossiping to achieve efficient broadcasts in partitioned networks. A message is first broadcasted within the current partition (using gossip). Every node keeps track of its neighbour nodes using regular hello messages. When a new neighbour is detected, the hello message is appended with a record that lists the recently received messages. By sharing these lists, the nodes can conclude that the new neighbour comes from a different partition which means that the nodes should exchange messages with each other. To find out which messages to share, a node sends the messages it already has so that the neighbour nodes can send the missing ones.

Vollset and Ezhilchelvan [22] present a manycast algorithm called Scribble. It is designed to be partition-tolerant and uses a node signature to keep track of informed nodes (several different signature types are suggested). In Scribble, a subset of nodes (termed responsible) periodically send messages until the message is considered to be delivered by a sufficient number of nodes (or another node takes over that responsibility).

Cooper et al. [6] propose a gossip-based broadcast algorithm called encounter gossip. The basic idea is to let a node repeatedly broadcast a message a fixed number of times, but only when encountering a new neighbour. This means that both the overhead and the delivery ratio can be kept constant for varying encounter frequencies, without the need for a time-based expiry. That is, if the nodes take long time to meet, then the counter will be slowly incremented so the packets will remain longer in the network. If, on the other hand nodes meet frequently, the counter is updated more often, and the packet can be discarded earlier. This model seems to be mostly suitable for scenarios with homogeneous mobility where nodes are likely to meet new nodes at more or less regular intervals.

Much of the more recent interest in delay-tolerant networking has focussed around social forwarding algorithms where the previous interaction patterns of nodes are used to make routing decisions. Such an approach is also taken by Gao et al [10] who use network centrality metrics to select message relay nodes.

2.2 Differentiation mechanisms

The notion that critical resources such as bandwidth and energy are constrained in wireless networks has always been at the core of research efforts in wireless and ad hoc networks. Naturally, for sparse networks with intermittent connectivity, the problem of limited resources becomes even more important. When faced with a situation where demands exceeds the available resources (i.e., an overload situation), it can make sense to prioritise which messages to disseminate and which ones to ignore. Such a prioritisation approach can be based on application-defined priorities (sometimes called message utility), or based on system-defined priorities with the aim of improving overall system performance metrics.

If the message priorities are defined by the application layer or if the network traffic can easily be divided into different quality-of-service classes, then the network protocols and buffer policies can be tailored to uphold the quality of service of the most important messages at the expense of the less important

messages. This can be done in several different ways, including weighted fairness and class-based queuing [8].

Message differentiation can also be useful even if there is no easy way to determine the relative importance of messages. By considering other message properties such as their size, destination, and time to live it is possible to optimise how network resources are used to provide better delivery rates and lower latencies. Two main strategies can be discerned for how to construct a message prioritisation policy, *message centred* and *system centred*. In the message centred approach, message properties are used to determine an ordering among messages, for example by giving higher priority to messages with a closer deadline. The resulting policies are then evaluated through analytical and experimental studies. In the system centred approach, an analytical model of the system is created which allows analytically deriving parameter values that optimises some particular system performance metric (e.g., delivery ratio). However, it might be difficult to obtain a good analytical model of a complex network and the resulting optimal policy might require global knowledge of the current network properties.

2.3 Message differentiation in intermittently connected networks

There are several ways in which message differentiation can be used in delay-tolerant networks. In *buffer management* MD is used to determine which messages to keep and which ones to drop when the buffer is full. Gossip-style protocols can used MD to determine whether or not to and in which order to forward a message [16, 7]. MD can also be used to determine the order in which to send messages when a node discovers a new neighbour, this is often called *message scheduling*.

Due to the overwhelming focus on unicast routing in the DTN community, most works on MD consider only unicast routing even though the policies are often general enough to be applicable for one-to-many communication as well. In particular, there are several works [17, 13, 19] that apply message differentiation to epidemic routing which is basically a one-to-many message dissemination protocol used to reach a single destination.

For example, the PRioritised EPidemic (PREP) routing algorithm by Ramanathan et al. [17] creates a topological map of the network including a link estimation metric. This information is used to prioritise messages that are closer (shorter paths) to their destination. Moreover, the expiry and creation time of messages are considered when scheduling messages for transmission.

The RAPID protocol by Balasubramanian et al. [3] is a utility-driven unicast protocol which tries to estimate the remaining time until a packet is delivered based on the historical meeting patterns in the network. This information is used by the protocol to greedily prioritise messages with for example the smallest expected delivery time.

Krifa et al. [12] derive a globally optimal policy for buffer management and message drop in networks with exponentially (possibly with different pair-wise means) distributed inter-meeting times. They also discuss a framework to collect network statistics as required to make decision at a local node level. A similar approach is taken by Elwhishi et al. [9] who also use a network estimation framework to collect information necessary to make scheduling decisions.

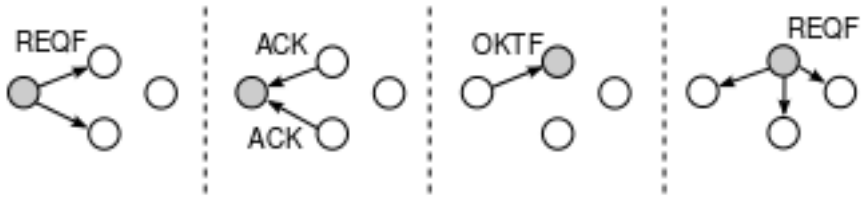

Figure 1: Random Walk Gossip Handshake

However, their analytical framework is based on a fluid epidemic model rather than discrete.

Differences among nodes in their available energy and bandwidth can have a big impact on the success of message delivery. Sandulescu et al. [18] focus on estimation of these node-specific resources in the vicinity of a node. This allows custodian selection and message ordering to take into account the available bandwidth and energy of its neighbours and thereby make better use of these resources, especially when messages differ in size.

Common for many of the above works, is that they try to estimate the current state of the network in order to improve the quality of MD mechanisms. In this chapter we take a different approach and investigate how MD can be achieved using rich message metadata that itself captures the relevant information. The benefit of this approach is that it does not require any apriori knowledge of the network characteristics, does not cause a lot of signalling overhead, and is not affected by changes in the network.

3 Random walk gossip

Random Walk Gossip [2] is a message dissemination protocol designed to cope with the challenges faced in disaster area networks including scarcity of bandwidth and energy, as well as unknown and unpredictable network topologies with partitions. RWG is a manycast protocol, which means that a message is intended to reach a given number k of nodes. When k nodes have been reached, the message is k-delivered and does not need to be propagated anymore, thus not wasting bandwidth and energy. RWG uses a store-carry-forward mechanism to cope with partitions.

We present an overview of the protocol in three steps. First we discuss the basic random walk mechanism in which messages are actively propagated in the network. Next we discuss the message structure and how the protocol uses this to perform message dissemination efficiently. Finally, we discuss the message activation mechanism which is of importance when adding message differentiation to the protocol.

3.1 Random walk and handshake mechanism

When a message is sent in a connected part of the network, it performs a random walk over the nodes, until all the nodes in the partition are informed of this message. This is controlled by a three-way packet exchange shown in Fig. 1. First a Request to Forward (REQF), that includes the message payload,

is sent by the current custodian of the message (grey node in the picture). The neighbouring nodes that hear the REQF reply with an acknowledgement packet (ACK), unless at least L nodes have already sent an acknowledgement (L being a parameter for which 3 has been found to give best results). The custodian randomly chooses one of the nodes that acknowledged and sends an OK to Forward (OKTF) to this node indicating that it will be the next custodian. The other nodes retain the message without actively disseminating it. In addition to delivering the message to the application layer, they keep the message as *inactive* until it expires. Partitions can be overcome by the movement of nodes. Thus, new uninformed nodes will be informed by some node that keeps the message as *inactive* and restarts to disseminate. This process will continue as long as no more uninformed nodes remain in the network or the message is k-delivered.

Finally, when a node realises that a message is k-delivered it sends a Be Silent (BS) packet to its vicinity. This packet will cause all receiving nodes to also realise that the message is k-delivered and thus remove it from their buffers. No new BS packets are sent upon the reception of a BS packet.

3.2 Message metadata

All the packet types share the same header structure. Since there are some complex features of the protocol which we do not discuss here, we concentrate on the most important header fields:

- Packet ID, 64bit identifier
- Group size (i.e., the k parameter), 16 bit integer
- Time to live (TTL), remaining time in milliseconds until the packet expires, 32bit integer
- Starting TTL, the lifetime of the message when first sent (stays constant), 32bit integer
- Informed, bit vector that represents informed nodes, 256bits

Note that the starting TTL field is not part of the original RWG protocol, but have been added in this study since this information is necessary for some of the prioritisation policies which we investigate. This adds 4 bytes to the header which slightly increases the overhead to the protocol. The full header size is 92 bytes and in our simulations we have used a payload of 500 bytes so the increase in packet size is less than one percent.

The *informed* bit vector (implemented as a Bloom filter) is used to keep track of which nodes have seen a given message (see Figure 2). When a node receives the message it produces a hash of its own address and puts a 1 in the bit vector in the field corresponding to the hash. This allows the protocol to know when a message is k-delivered, and to tell the potential future recipients of the message how far the message has reached towards its dissemination goal (summing the number of 1's indicates the current known local knowledge of this).

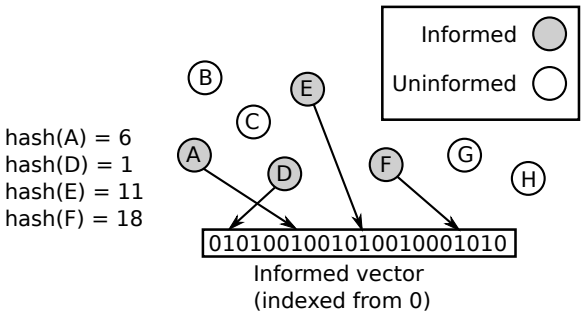

Figure 2: Informed bit vector

3.3 Message activation mechanism

The *informed* vector is also used for deciding whether or not to activate an inactive message. When a node A hears a new neighbour B, A will go through the messages stored in its buffer (i.e., the set of inactive messages) to see if B has not yet been informed of any of these messages, in which case those messages will be reactivated and broadcast to node B (and other uninformed nodes in the vicinity). If the hash of node B is marked with a '0' in the informed vector, then it is likely that B has not heard of this particular message. Naturally, the information in the informed vector represents a limited and potentially stale view of the network, meaning that some messages will be unnecessarily activated. Moreover, since the informed vector uses a hash function to index nodes, hash collisions can happen which will cause the message not to wake up even if that would have cause it to spread to a new node.

4 RWG and Message differentiation

The original version of RWG does not make any distinction or prioritisation between different packets. The protocol is designed so that each message should always produce a minimal amount of overhead independently of whether there are other packets in the system. Even in case of high load, the protocol will not adapt, but will simply send packets to the MAC layer where they are put in a queue. In order to make RWG better suited to deal with overloads we added the possibility to differentiate messages based on their properties. This message differentiation is used in two cases 1) when activating messages and 2) in the acknowledgement step of the handshake mechanism.

Message activation which mainly happens when a node hears another (potentially new) neighbour and activates messages was explained in Section 3.3. With message prioritisation enabled, message are activated in priority order. Thus high-priority messages are always activated first, allowing them to spread faster in the network compared to other messages.

In the acknowledgement step as explained in Section 3.1, nodes will not send an acknowledgement if enough nodes have already replied. With message prioritisation enabled, this acknowledgement decision will also consider whether the message has higher priority than the other currently active messages in which case an acknowledgement will be sent in any case. Again this speeds up

delivery of high priority messages at the expense of lower priority messages.

We now proceed to present the four different prioritisation policies which we have studied as possible options for how to decide which messages to give higher priority.

4.1 Random order

As a baseline we consider a random ordering of the messages, which is basically the same as having no differentiation mechanism at all. This policy has the benefit of being simple and straightforward as well as providing fairness among messages. Other policies such as first-in-first-out could also have been used as a baseline. However, due to the way RWG frequently removes and inserts elements in the list of inactive messages there is no obvious notion of "first in queue" which is why we opted for the random order as the most logical baseline.

4.2 Least Informed First

When the system is overloaded so that some of the messages might miss their deadline, but most are delivered in time, then it makes sense to help the messages that have not yet reached many nodes. This is the rationale for the Least Informed First (LIF) policy which gives higher priority to messages with fewer informed nodes.

4.3 Earliest Deadline First

Earliest Deadline First (EDF) is of the classical scheduling algorithms. EDF is optimal among all scheduling algorithms for uniprocessor systems in the sense that for any task set, if there exists an algorithm that can schedule the tasks without any deadline overruns, then EDF will also find such a schedule (under some assumptions such as task independence). EDF gives the highest priority to the task with the earliest deadline. This means that task priorities can change dynamically over time and that it is possible to dynamically schedule arriving tasks.

4.4 Least Slack First

The Least Slack First scheduling can be seen as a combination of the EDF and LIF policies in the sense that it considers both the deadline of a message and how much progress the message has done. It is based on the notion of slack which in uniprocessor task scheduling is defined as $s(t,i) = (D(i) - t) - r(i,t)$, where t is the current time, $D(i)$ is the deadline of task i, and $r(i,t)$ is the remaining computation time of task i at time t. We use the same notion for messages by defining the remaining time as the number of remaining nodes to inform times an information spreading rate. Mathematically we write this as $r(i,t) = (k(i) - I(i,t))R(i)$, where $k(i)$ is the group size of message i, $I(i,t)$ is the number of nodes that has been informed by i at time t and $R(i)$ is the expected rate at which new nodes are informed. LSF gives higher priority to messages with a smaller slack value.

5 Evaluation with vehicular mobility models

In this section we present our simulation-based evaluation of RWG when combined with the above four different prioritisation policies. The main objective of the evaluation is to investigate how these policies behave in a large realistic large-scale scenario where the system load exceeds the network capacity.

We consider a vehicular scenario where cars use direct communication and local forwarding to exchange information about the current traffic scenario. Such information could be that a queue is building up in a certain location, that an emergency vehicle is approaching, or that there are obstacles in the road. Common for these examples is that they are relevant only for a short period of time, and that they need to be delivered within a short time frame to be of any use for the receiver. Moreover, the information is mostly local and messages need not reach all nodes in the network.

5.1 Experimental setup

We used the Ns3 network simulator version 3.16 which provides a packet-level simulation environment with a well-documented structure. In accordance with the evaluation scenario, we consider messages with a relatively short time to live (TTL). One third of the messages have a TTL of 5 seconds, one third have a TTL of 10 seconds and the rest can live for 20 seconds. We assume that even if some events can be of relevance for a longer duration than 20s, such information will be continuously resent as new messages. All messages have a group size of 30 nodes (unless otherwise stated), meaning that they should be delivered to at least 30 nodes in order to be k-delivered. Messages are generated at a constant rate but from randomly selected senders at a rate ranging from 5 to 25 messages per second.We have used the 802.11p wifi standard for car2car communication with a transmission power of 28.8 dbm, which is the maximum allowed power according to the standard and corresponds to a transmission range of approximately 300 meters in our simulation.

We logged a total of 300 messages for each point on the illustrated charts. In order to remove any boundary effects caused by the simulation we ran the simulation for 20s (same as the longest TTL) before starting to log messages, and we let the simulation continue 20s after creating the last of the logged messages, which resulted in a simulation time ranging from 52s to 100s. The simulation parameters are summarised in Table 1.

Mobility In order to get an as realistic vehicular mobility model as possible, we used a large scale mobility trace generated by the microscopic (i.e., each car is simulated with a driver model) traffic simulator Sumo. The trace is made available by the TapasCologne project [20] and is based on the German city of Cologne. The tracefile we used ranges from 6am to 8am and contains a total of 121140 vehicles.

We assume that only 20% of all vehicles are equipped with communication capabilities which reduces the number of nodes to 23492. In order to reduce the scale of the simulation we then decided to concentrate on the vehicles whose initial position was within 3km from the centre and which had some movement in a 20 minute period from 6:50 to 7:10. This reduced the number of cars to 1092

Time to live	5,10,20s
Group Size (k)	30
System load	5-25 messages/s
Wifi standard	802.11p SCH
Transmission power	28.8 dbm
Number of logged messages	300
Simulation time	52-100s
Mobility model	Vehicular
Number of nodes	1092 (at most 366 active)
Area	$28.3km^2$

Table 1: Simulation parameters

which is still a large number of nodes to handle with a packet-level simulator like Ns3.

We further take advantage of the fact that most journeys are relatively short meaning that most of the time the car is turned off (and thus not taking part in the network). It turns out that out of the 1092 vehicles in total, only 366 drive around at any point in time, the rest are parked. Therefore, in the simulator, we use 366 different node entities that behave as the 1092 original vehicles. The simulator allows nodes to instantaneously move from one position to another (when a node changes identity) as well as completely removes any information about which messages a removed node has encountered or stored. The end result is that we get the same system behaviour as the original 1092 nodes which corresponds to 20% of all the vehicles within 3km from the city centre.

Metrics We measure the performance of the differentiation mechanisms using k-delivery ratio and the k-delivery latency. The k-delivery ratio is defined as $\frac{1}{N}\sum_{i=1}^{N} d_i$ where N denotes the number of logged messages, and $d_i = 1$ if message i was delivered to at least k nodes and $d_i = 0$ otherwise. The latency of a k-delivered message is defined as the time from when the message was created to when it became k-delivered.

5.2 Comparison of message differentiation mechanisms

Message differentiation makes most sense when some messages cannot be given the required service due to restricted resources which is what happens in an overload situation. Thus, in order to assess the effectiveness of the different differentiation mechanisms, it makes sense to consider a scenario where the load negatively affects the system performance. In the case of our experimental setup this happens when the total number of messages introduced in the system (we denote this *system load*) exceeds 10 messages per second.

Figure 3 shows the k-delivery ratio of the different mechanisms as the load varies from 5 messages/second (system not overloaded) to 25 messages per second (severe overload). As expected, the performance when the system is not overloaded (5-10 messages/second) is basically the same independent of which differentiation mechanism which is used. When the load exceeds 10 messages/second the performance with random message ordering (which basically means no message differentiation) drops drastically, whereas prioritising mes-

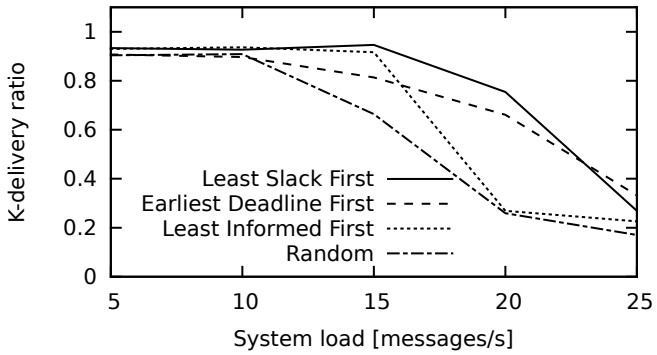

Figure 3: K-delivery ratio versus system load (k = 50)

sages with short deadline or slack keeps an acceptable performance for considerably higher loads. Specifically, if the LSF mechanism is used, the system is capable of handling at least 50% higher load compared to random message ordering in our scenario.

Earliest deadline first seems to behave similarly to LSF, but as we will soon see, there are some differences that cannot be seen in this graph alone. The Least Informed First mechanism which prioritises messages that have not yet reached as many nodes, performs well at moderate overload but quickly degrades in performance as the load increases further. This makes sense since LIF prioritises messages that have not yet reached a lot of nodes. If the overload is moderate, this will help the messages that would have otherwise missed their deadline. As the overload gets worse, and there is no chance of delivering all messages LIF is counterproductive since it gives higher priority to messages that anyway have no chance of being delivered.

Since RWG is a manycast protocol, it is also interesting to consider how the group size affects the protocol performance under different differentiation mechanisms. Figure 4 which shows the k-delivery ratio as a function of group size tells a similar story as in Figure 3. The LSF and LIF mechanisms allow the k parameter setting to be 25% higher (from 40 to 50) compared to the baseline without degrading the system performance.

As the load increases and the more transmissions that will happen in the system, the longer each message will need to wait in each step of the dissemination process. Figure 5 shows how this affects the average latency. As expected, the latency increases significantly, as the system becomes overloaded, and the random order baseline shows the worst performance. More interestingly, as the overload becomes severe with 25 messages/second, the earliest deadline first approach seem to perform much better compared to the other mechanisms.

5.3 Detailed analysis of resulting latency distributions

In the previous subsection we discussed how the delivery ratio and average latency was affected by increasing load and group size under four different scheduling policies. In order to gain a deeper understanding of these results we also analyse the latency distributions which are more useful illustrate how quickly

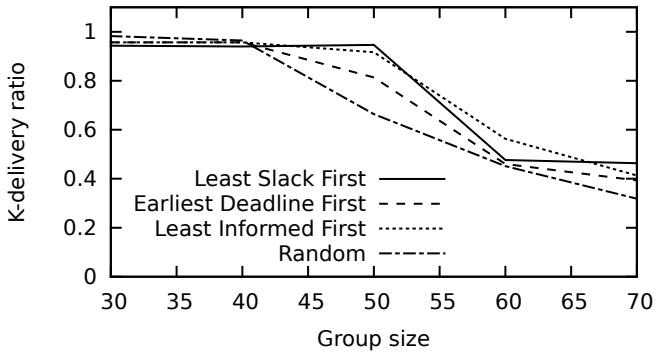

Figure 4: K-delivery ratio versus group size (15 messages/s)

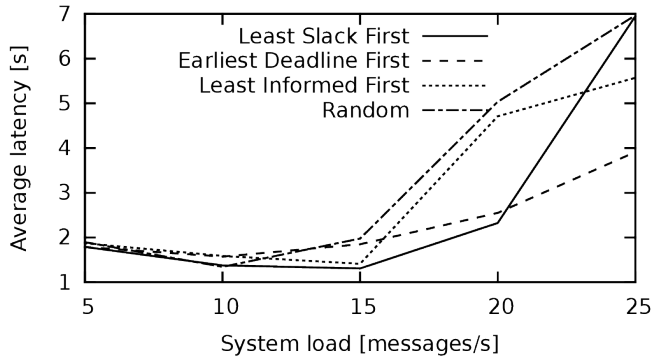

Figure 5: Average latency versus system load

messages gets disseminated in the network.

Figure 6 shows the latency distribution as a Cumulative Distribution Function (CDF) for a system load of 15 messages/s. Here we consider undelivered messages as having an infinite latency, which is the reason why the distributions never reach 1. Interestingly, while the final delivery ratio differs among the different policies, they all have a steep rise before 5 seconds after which not many messages become delivered.

Figure 7 shows the corresponding results for a system load of 25 messages/s meaning that the system is severely overloaded. There are some interesting differences. First of all, EDF seems to perform much better than the other policies. We could see this also in the average latency which was lower for EDF at 25 messages/s. Now we can see how this is reflected in the shape of the latency distributions. For EDF the shape is similar to those in Figure 6 in the sense that most of the k-delivered messages have a latency of less than 5 seconds. For the other policies, the shape is closer to a straight line with message latencies being spread evenly in the interval 0-20 seconds. Note that after 20s, all curves will stay at a constant level since no messages are delivered after more than 20s which is the longest time to live of any message in the system.

Finally, figures 8 and 9 show the latency distribution for the most urgent messages (those with a time to live of 5 seconds) for load profiles of 15 and

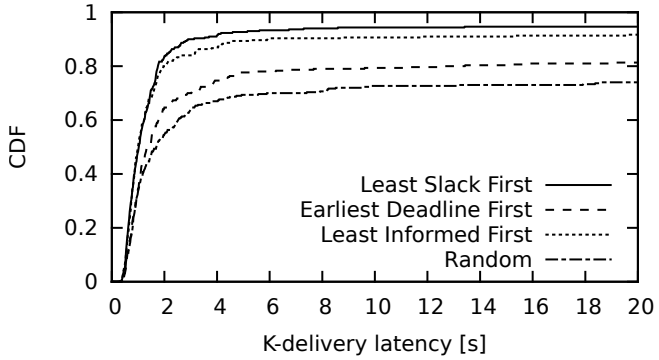

Figure 6: Cumulative Distribution Function (CDF) at 15 message/s

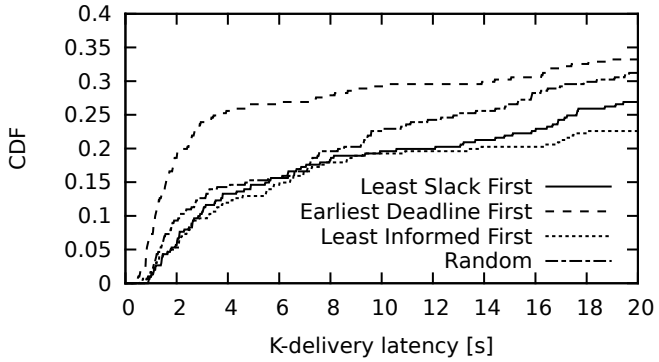

Figure 7: Cumulative Distribution Function (CDF) at 25 messages/s

25 messages/s respectively. At 15 messages per second the latency distribution of the urgent messages is similar to when considering all messages. The main difference is that EDF performs significantly better for the urgent messages (compared to the EDF for the total messages in Figure 6). At 25 messages per second (Figure 9), the difference is more significant. Here EDF clearly outperforms the other policies.

6 Discussion

In this chapter we discussed message dissemination protocols for intermittently connected networks and how message differentiation has been used to increase message delivery and reduce latency in such networks. A lot of progress has been made in understanding intermittently connected networks in the last few years. Most general basic phenomena are fairly well understood by the community, and deeper insights are often inhibited by the lack of proper domain specific data which can be analysed. Message differentiation has been discussed more or less actively since the early paper on RAPID by Balasubramanian et al. [3], but the main focus has been on unicast routing for networks with very long delays.

We have tried to complement the existing body of work by studying how

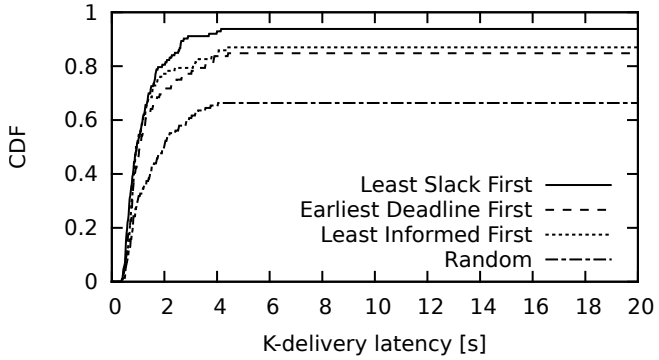

Figure 8: Cumulative Distribution Function (CDF) for urgent messages at 15 messages/s

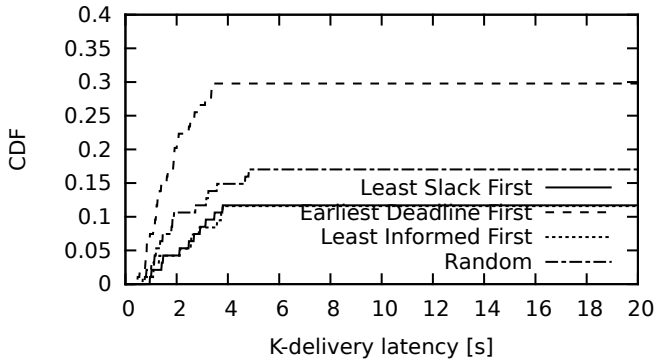

Figure 9: Cumulative Distribution Function (CDF) for urgent messages at 25 messages/s

message differentiation mechanisms can be effectively used in an overloaded network with very short message deadlines. The study was based on four different prioritisation policies, random order, Earliest Deadline First, Least Informed First, and Least Slack First. Using realistic vehicular traces we simulated the performance of an overloaded network running the RWG manycast protocol in conjunction with these four policies. Due to the way RWG keeps track of informed nodes, it is possible to implement these prioritisation policies without collecting network statistics over time in order to estimate some locally unknown parameters.

The Least Slack First, which is basically an adaption of a well-known uniprocessor scheduling algorithm, seems to provide the best overall results. For more severe overloads, the simpler EDF policy outperforms the other policies, especially in keeping down the latency of the delivered messages.

The bottom line is that relatively simple time-based prioritisation mechanisms can be a powerful tool to increase the performance in wireless networks with intermittent connectivity and constrained resources. In particular, for vehicular networks with urgent message dissemination requirements, the system load can be increased by 50% by using the Least Slack First policy.

References

[1] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4), 2005. doi: 10.1016/j.comnet.2004.12.001.

[2] M. Asplund and S. Nadjm-Tehrani. A partition-tolerant manycast algorithm for disaster area networks. In *28th International Symposium on Reliable Distributed Systems (SRDS)*. IEEE, 2009. doi: 10.1109/SRDS.2009.16.

[3] A. Balasubramanian, B. Levine, and A. Venkataramani. DTN routing as a resource allocation problem. *SIGCOMM Comput. Commun. Rev.*, 37(4), 2007. doi: 10.1145/1282427.1282422.

[4] R. Baldoni, K. Ioannidou, and A. Milani. Mobility versus the cost of geocasting in mobile ad-hoc networks. In A. Pelc, editor, *Proc. 21st International Symposium on Distributed Computing (DISC)*, volume 4731 of *Lecture Notes in Computer Science*, pages 48–62. Springer-Verlag, 2007.

[5] C. Carter, S. Yi, P. Ratanchandani, and R. Kravets. Manycast: exploring the space between anycast and multicast in ad hoc networks. In *Proc. 9th annual international conference on Mobile computing and networking (MobiCom)*. ACM, 2003. doi: 10.1145/938985.939013.

[6] D. Cooper, P. Ezhilchelvan, and I. Mitrani. Encounter-based message propagation in mobile ad-hoc networks. *Ad Hoc Networks*, 7(7), 2009. doi: 10.1016/j.adhoc.2008.12.003.

[7] A. Cornejo, C. Newport, S. Gollakota, J. Rao, and T. Giuli. Prioritized gossip in vehicular networks. *Ad Hoc Networks*, 11(1), 2013. doi: 10.1016/j.adhoc.2012.06.016.

[8] M. El-Gendy, A. Bose, and K. Shin. Evolution of the internet QoS and support for soft real-time applications. *Proceedings of the IEEE*, 91(7), 2003. doi: 10.1109/JPROC.2003.814615.

[9] A. Elwhishi, P.-H. Ho, K. Naik, and B. Shihada. A Novel Message Scheduling Framework for Delay Tolerant Networks Routing. *IEEE Transactions on Parallel and Distributed Systems*, 2012. doi: 10.1109/TPDS.2012.197.

[10] W. Gao, Q. Li, B. Zhao, and G. Cao. Social-Aware Multicast in Disruption-Tolerant Networks. *IEEE/ACM Transactions on Networking*, 20(5), 2012. doi: 10.1109/TNET.2012.2183643.

[11] A. Khelil, P. J. Marrón, C. Becker, and K. Rothermelns. Hypergossiping: A generalized broadcast strategy for mobile ad hoc networks. *Ad Hoc Netw.*, 5(5), 2007. doi: 10.1016/j.adhoc.2006.03.001.

[12] A. Krifa, C. Barakat, and T. Spyropoulos. Message Drop and Scheduling in DTNs: Theory and Practice. *IEEE Transactions on Mobile Computing*, 11(9), 2012. doi: 10.1109/TMC.2011.163.

[13] Y. Li, Y. Jiang, D. Jin, L. Su, L. Zeng, and D. O. Wu. Energy-Efficient Optimal Opportunistic Forwarding for Delay-Tolerant Networks. *IEEE Transactions on Vehicular Technology*, 59(9), 2010. doi: 10.1109/TVT.2010.2070521.

[14] M. Mongiovi, A. K. Singh, X. Yan, B. Zong, and K. Psounis. Efficient multicasting for delay tolerant networks using graph indexing. In *2012 Proceedings IEEE INFOCOM*. IEEE, 2012. doi: 10.1109/INFCOM.2012.6195503.

[15] K. Obraczka and G. Tsudik. Multicast routing issues in ad hoc networks. In *Proc. IEEE International Conference on Universal Personal Communications (ICUPC)*, 1998. doi: 10.1109/ICUPC.1998.733066.

[16] P. Ramanathan and A. Singh. Delay-Differentiated Gossiping in Delay Tolerant Networks. In *2008 IEEE International Conference on Communications*. IEEE, 2008. doi: 10.1109/ICC.2008.619.

[17] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan. Prioritized epidemic routing for opportunistic networks. In *Proceedings of the 1st international MobiSys workshop on Mobile opportunistic networking - MobiOpp '07*. ACM Press, 2007. doi: 10.1145/1247694.1247707.

[18] G. Sandulescu, P. Schaffer, and S. Nadjm-Tehrani. Exploiting resource heterogeneity in delay-tolerant networks. *Wireless Communications and Mobile Computing*, 13(3), 2013. doi: 10.1002/wcm.2195.

[19] K. Shin and S. Kim. Enhanced buffer management policy that utilises message properties for delay-tolerant networks. *IET Communications*, 5(6), 2011. doi: 10.1049/iet-com.2010.0422.

[20] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. Barcelo-Ordinas. Generation and analysis of a large-scale urban vehicular mobility dataset. *Mobile Computing, IEEE Transactions on*, PP(99), 2013. doi: 10.1109/TMC.2013.27.

[21] K. Viswanath and K. Obrazcka. An adaptive approach to group communications in multi hop ad hoc networks. In *Proc. seventh IEEE Symposium on Computers and Communications (ISCC)*. IEEE Computer Society, 2002. doi: 10.1109/ISCC.2002.1021730.

[22] E. Vollset and P. Ezhilchelvan. Design and performance study of crash-tolerant protocols for broadcasting and reaching consensus in MANETs. In *Proc. 24th IEEE Symposium on Reliable Distributed Systems (SRDS)*, 2005. doi: 10.1109/RELDIS.2005.15.

[23] W. Zhao, M. Ammar, and E. Zegura. Multicasting in delay tolerant networks: semantic models and routing algorithms. In *Proc. 2005 ACM SIGCOMM workshop on Delay-tolerant networking (WDTN)*. ACM, 2005. doi: 10.1145/1080139.1080145.