Adding Redundancy to Replication in Window-aware Delay-tolerant Routing

Gabriel Sandulescu, *University of Luxembourg, Luxembourg*, Email: gabriel.sandulescu@uni.lu
Simin Nadjm-Tehrani, *Linköping University, Sweden*, Email: simin@ida.liu.se

***Abstract*— This paper presents a resource-efficient protocol for opportunistic routing in delay-tolerant networks (DTN). First, our approach exploits the context of mobile nodes (speed, direction of movement and radio range) to estimate the size of a contact window. This knowledge is exploited to make better forwarding decisions and to minimize the probability of partially transmitted messages. Optimizing the use of bandwidth during overloads helps reduce energy consumption since partially transmitted messages are useless and waste transmission power. Second, we use a differentiation mechanism based on message utility. This allows allocating more resources for high utility messages. More precisely, messages are replicated in the order of highest utility first, and removed from the buffers in the reverse order. To illustrate the benefit of such a scheme, global accumulated utility is used as a system-wide performance metric. Third, we present a combined fragmentation/redundancy scheme which not only improves delivery ratio but also, if infrastructure is available, allows messages to be completed by pulling dropped fragments.**

Simulations illustrate the benefit of our model and show that our scheme provides lower overhead and higher delivery ratio, as well as higher accumulated utility compared to a number of well-known algorithms including Maxprop and SprayAndWait.

***Index Terms*— DTN, Routing, Opportunistic, Contact Window, utility, fragmentation**

I. Introduction

Delay-tolerant networks (DTN) define an abstraction layer on top of transport layer and below application layer, called bundle layer. As no assumption can be made about underlying networks, this overlay architecture is responsible for routing data from source to destination. DTN neither defines any fixed-length data units nor puts any upper or lower bounds on application data unit size. The bundle layer is responsible for the end-to-end delivery mechanism of messages, called virtual message forwarding ([1], [2]). DTN uses a minimal conversational model. Therefore, DTN applications should be designed in such a way as to optimize the number of end-to-end transactions, using larger, self-contained messages. Although a large bundle size is beneficial in a store-and-forward context, it may pose new challenges when it comes to resource-constrained networking.

This paper is based on "Opportunistic Routing with Window-aware Adaptive Replication" by G. Sandulescu and S. Nadjm-Tehrani which appeared in the Proceedings of the 4^{th} Asian Internet Engineering Conference (AINTEC08), Bangkok, Thailand, November 2008.

The second author was partially supported by a grant from the Swedish Research Council.

In particular, considering that networking activities account for 10-50% of the energy spent by a mobile device [3] and the gap between battery capacity and mobile-device energy requirements is increasing, designing energy-efficient network architectures is vitally important.

Motivated by these two factors, large bundle size in DTN and a constant need for energy-efficient schemes, we propose in this work: (1) an energy-efficient manner to treat large bundles in intermittent connectivity when fragmentation is not available, (2) a mechanism to deal with transient overloads in traffic volumes by prioritizing messages according to size and utility, (3) methods to further boost delivery ratio when additional technologies such as fragmentation and coding are available. Utility is a general term to measure benefit. We assume there is a system-wide agreement about what benefit accrues from delivering a packet relative to the resources it uses.

In this paper, we demonstrate distributed mechanisms at node level that optimize the use of transmission power during relays, as well as bandwidth and memory during overloads. DTN architecture actually offers three relative priority classes which differentiate traffic based on an application's expression of urgency at the message source. These have some impact on solving traffic contention as well as resource allocation. For example, in the current reference implementation [4], when storage at one node becomes short, expiration of bundles will start with the low priority class. While this is a suitable mechanism for differentiation at user level, it does not take account of message size and thereby does not provide an optimized use of resources at system level. In this paper we show how "per bit utility" can be combined with the traditional idea of priorities to achieve better use of resources at system level.

The contributions of this paper are as follows. We present a utility-aware DTN routing scheme, Opportunistic Routing with Window-Aware Replication (ORWAR), based on calculation of the maximum deliverable bundle size estimated from the mobility context. The proposed algorithm uses a fixed number of message replicas similar to those described by Spyroupoulos et al. [5]. In our case these copies are not automatically distributed at the first meeting but based on the evaluation of the contact window currently at a node's disposal. ORWAR selects the most valuable message to be sent, whose size does not exceed the doable limit and thus avoids partial transmissions. Selection of a message is based both on

the size and contact characteristics and the utility per bit of the message. We compare performance of ORWAR in terms of overhead, delivery ratio and latency, as well as system-wide accumulated utility with other protocols: MaxProp [6], SprayAndWait [5], Prophet [7], among others. We demonstrate improved delivery ratio by up to 15% more when using large messages and show that this degrades gracefully when higher loads are injected in the system. We then evaluate the performance when fragmentation/coding is variable and show that the ideal fragmentation size should be in the range of the previously calculated contact window. When fragmentation is possible we propose a scheme using a more expensive infrastructure to pull the dropped fragments.

This paper is an extension of a preliminary work on ORWAR [8] presented earlier. The work has been extended in three directions: (1) Extensive simulation of all routing schemes in different scenarios allowing to determine a confidence interval, (2) Extension of the framework by adding fragmentation/redundancy which shows substantial gains in delivery ratios on top of the base protocol, (3) Improvement of the delivery ratio by pulling the dropped fragments when an infrastructure network can be used for the "last few". Hence, the earlier DTN approach is shown to work in a hybrid context. As a target application for this work we can imagine an urban scenario where all nodes are mobile, and use short-range interfaces for opportunistic communications. In the latter part of the paper we demonstrate that in a crisis scenario where the (celular) infrastructure is highly overloaded, the opportunistic scheme can be combined with a selective use of the infrastructure resources to increase the capacity of the DTN network.

In the next section we provide a background to routing in delay-tolerant networks and related problems. In section III we present ORWAR. Section IV is devoted to comparative evaluation of the protocols whilst in section V we present the additional benefits of the fragmentation / coding scheme. The paper ends with some conclusions and ideas for future work.

II. Background and Related Work

Our work benefits from and builds upon a large body of work in opportunistic routing. This section, divided into four categories, introduces some key elements by referencing related works. We start by arguing that bundle size is an important factor in opportunistic DTN routing, especially within the mobility context. Then, we review replication as a method to increase delivery probability in an opportunistic environment. In the third subsection we cover techniques such as message differentiation used as an effective means to control resource allocation and key success criteria of a routing scheme. We conclude by describing fragmentation and redundancy mechanisms and their benefits in an opportunistic context.

A. Mobility and bundle size

Considering a network of nodes advancing at various speeds, it is obvious that the delivery ratio is not driven solely by the nominal available bandwidth. Ott and Kutscher [9], after carrying out extensive laboratory and field measurements, show that cars can reach about 1800 m of connectivity when connecting to a stationary WLAN point of access on a highway and moving at 120 km/h. They show that the size of data exchanged is between 30 and 70 MB in one pass. This implies that for mobile adhoc transfers there is a maximum size limit for the bundle to be exchanged depending on relative speed. To transmit a bundle exceeding this size, a node has no other alternative than to wait for a better contact opportunity (i.e. a node with lower relative speed) or to use proactive fragmentation. In a disaster scenario [10], there may be little prior knowledge about mobility patterns, but the type of communication and size of data/messages to be exchanged is likely to be known among rescue teams.

Another study by Ott and Kutscher [11] shows that application protocols are differently suited for DTN, those with less interactive features performing better. For example, while email exchange is fundamentally asynchronous, the present application protocols for sending (SMTP) and retrieving (POP3, IMAP4) mails are fairly verbose, involving numerous message exchanges, and often require user credentials to be provided. DTN suggests combining all application level data and metadata to form a single bundled message, in order to minimize the number of end-to-end transactions. For example, all IMAP4 metadata (login-name, password, host, port etc.) and actual data (attachment(s) if applicable) can be sent together, bundled into one single message. In the absence of a DTN bundle layer size limit, we can expect messages to get bigger in size. In our work we take message size and relative speed into account when selecting the window for forwarding/replication. In the rest of the paper bundles and messages are used interchangeably.

B. Message replication

Opportunistic routing makes no assumption about the contact schedule between nodes. In order to cope with this uncertainty, some routing algorithms forward multiple copies of each message to a few custodians in order to increase the chance that at least one copy will be delivered [12]. This also decreases delivery latency. However, it also consumes resources (bandwidth and implicitly energy, as well as storage at custodians) proportional to the number of copies. Therefore, a whole class of algorithms avoid replication, such as Prophet [7], which makes use of historic encounters to estimate probability and only forwards messages to some neighbor whose probability exceeds a certain threshold. The Epidemic protocol [13] is an early replication attempt but this strategy works well only when message volume and node density are very low. Other protocols, like SprayAndWait [5], overcome the overhead problem of epidemic schemes by maintaining only a controlled number of copies in the network. They also

show that the number of copies necessary is independent of network size. Our algorithm uses a similar replication mechanism but with a different message selection scheme. Further up on the complexity scale, one of the best performing protocols, MaxProp [6], also uses multiple copies. Besides, Maxprop calculates the cost for each route while also leveraging delivery notifications to purge old replicas. MaxProp is intended for use in storage and bandwidth-constrained environments; therefore, it purges messages that have a lower chance to be delivered.

C. Resource-centric routing

Traditional routing schemes usually focus on selecting the delivery path by optimizing a simple metric (number of hops or delay). For DTN networks, as Zhang suggested in her survey [14], the success criteria of a routing scheme are still a research topic. Most protocols still try to maximize delivery ratio or to minimize transmission delay between source and destination. However, there are other criteria to be taken into account. Reducing overhead, for example, is an important topic, especially in order to improve energy efficiency. Even if bandwidth is sufficient for the given load, every transmission consumes power and depleted battery levels can be of concern. Many DTN scenarios depend on devices with limited energy supplies and energy-aware frameworks are of high interest in all mobile contexts. There are several approaches which deal with energy constraints in a DTN environment. One proposal is to control radio wakeup intervals [15] or neighborhood sensing [16] by probing algorithms which trade off energy consumption against the probability of missing a contact. However, this will usually imply further degrading connectivity as network interface sleep time is increased. Another proposed solution is to architecture the network into multiple tiers as in Zebranet [17] or DataMules [18], thus maximizing energy savings at one single tier - namely the most sensitive one - the sensor tier. However, such a network specialization is hard to implement in social, urban scenarios where heterogeneity is the key and association/dissociation of nodes to the network is very dynamic. Our approach to this problem is to select the message replica in a contact-window aware manner, that limits energy waste.

A rather different approach is to introduce utility and differentiation between messages when dealing with resource allocation. This has been studied in fully connected mobile ad-hoc networks [19] where construction of the route also involves maximizing the accumulated utility for the whole network. In the context of DTNs, Balasubramanian et al. [20] use utility in order to optimize resources with respect to delay related metrics, in particular minimizing average delay, minimizing missed deadlines or minimizing maximum delay. In our case the use of utilities will result in efficient use of transmission power and optimization of bandwidth and memory. Spyropoulos et al. [21] use utility to choose the fittest custodian node to carry the message. However, in both papers, there is no network-wide optimization of the accrued utility.

In our work, we use the concept of utility for network-wide optimization, and relate it to message priority to enforce differentiation. The global optimization mechanism is, however, built-in in the routing algorithm in a distributed fashion.

Another important aspect that we focus on is node density. Earlier work has studied similar allocation schemes in highly overloaded and dense networks [19]. Zhang [14] and Erramilli et al. [22] show that in sparse networks differences between routing protocols are more accentuated as a bad forwarding decision could lead to infinite delay without rollback possibility, due to shortage of contacts. In our case, we have chosen a very sparse test bed.

D. Message fragmentation and DTN

The ability to fragment bundles, either prior to transmission (proactive fragmentation) or while in transit (reactive fragmentation) has been introduced early in the DTN design [23]. As Bundle Layer is agnostic about lower layers, the DTN architecture cannot rely on them for fragmentation. Instead, it is the responsibility of upper layers (bundle layer, application layer) to limit the impact of a challenged environment by using smaller messages [24]. There is no notion of Maximum Transmission Unit defined in DTN, but, as our simulation confirms in section V, there is a best-fitted fragment size, which is shown to be related to the smallest contact window within a network with a given mobility.

The vast majority of routing protocols (SprayAndWait [5], MaxProp [6], RAPID [20]) consider bundles as indivisible, making routing decisions simpler. Jain et al. formulate the problem of optimizing the probability of successful message delivery by splitting, replicating and erasure coding [25]. This problem is complex even when study is limited to the simpler case where delivery probabilities remain constant over time. In this work, we adopt a practical solution where fixed sized fragments, created by proactive fragmentation, are routed independently over different paths and reconstructed at destination.

Transferring large data in small fragments without explicit acknowledgments may lead to reliability degradation in best-effort networks because all fragments should arrive at destination in order for the initial message to be reconstructed [26]. Redundancy ratio can then be chosen to improve delivery ratio while keeping volume transferred over the network at an acceptable limit. A redundancy mechanism, such as the one indicated by Byers et al. [27], overcomes the need for receiving a fixed set of messages before reconstruction. It represents a way to regenerate messages at destination from a proper subset of all fragments with low overhead using Reed-Solomon or Tornado codes. In this paper, the choice of the exact erasure coding algorithm is not important, as we focus on showing that fragmentation and coding are especially worthwhile if only the ”last few” fragments can be pulled via an alternative infrastructure that is more reliable but highly expensive.

III. CONTACT BASED ROUTING

This section describes Opportunistic Routing with Window-Aware Replication (ORWAR), a distributed algorithm running at bundle layer.

A. Protocol design rationale

ORWAR uses local connectivity knowledge in order to route messages from source to destination. Connectivity knowledge is not available in advance but is gathered from the vicinity on a peer-to-peer basis during contact. Specifically, we know neither message arrival rate nor meeting schedule, so routing is completely opportunistic. However, a node knows its own vectorial speed.

There are two design criteria for our algorithm. First, one goal of the algorithm is to optimize system level resources, in particular energy and bandwidth. Message priority levels are a simple means of achieving differentiation when resources are scarce. However, when message sizes vary considerably we need a more fine-grained differentiation mechanism. Thus, we propose utility/bit as an abstract declaration of the benefit of one transmission in comparison to others. Assuming that every message comes with a given utility value, accumulated utility can be used as a system-level evaluation metric. Note that the unit for measuring utility is unimportant since it only reflects a global measure of benefit. Second, we strive for a high delivery ratio in partitioned networks. For this we use store-and-forward and replication mechanism in DTNs. In order to make replication energy-efficient, we propose the decision to be based on both utility/bit for involved messages and local connectivity characteristics.

To perform routing under intermittent connectivity, ORWAR proposes a multi copy routing scheme, using a controlled replication and a fixed number of copies distributed over the network. At each contact the node tries to forward half of the message copies, keeping the rest for itself. Up to now, this is similar to the SprayAndWait mechanism presented by Spyropoulos et al. in [5]. However, enhancements are done in 4 directions:

1) Messages with the best utility per bit ratio are first selected and they are sent only if their size meets contact properties, thus diminishing partially transmitted messages.
2) The replication factor is a function of message utility, thus increasing delivery probability and diminishing latency for bundles with highest utility.
3) Purging messages from the buffer starts with the least utility per bit message.
4) Bundles known to be delivered are removed.

B. Algorithm data structures

Every node i keeps the following data structures: 1) the message queue (mq_i), that includes information about utility (u_k) and size (s_k) for each message m_k, kept in utility/bit order, and 2) a record of known delivered messages (kdm_i). In presence of unlimited message sizes, we assume finite buffers for custodians.

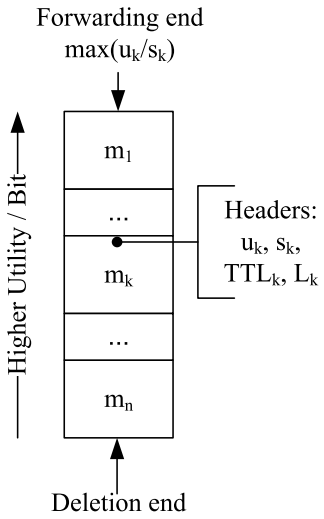

Figure 1. ORWAR queue

Figure 1 shows the structure of the message queue. New messages from the application layer as well as those from neighboring peers are inserted in the correct position with respect to the u_k/s_k ordering. Messages are deleted from the lower end of the queue. This might occur when a new message with higher utility per bit rate is to be inserted and the queue is full. In order to relate to the notion of priority in DTNs we have chosen 3 values of utility to reflect differentiation amongst messages. However, the approach is general and can be extended to multiple levels of utility. In this work the utility per message is time-invariant. An extension of the work may consider time-varying utilities. Every message header also includes L_k which denotes the intended number of message copies.

Finally, TTL_k is an application-based parameter that indicates time to live for a message. This can be implemented as an absolute value where all nodes can be considered to have access to synchronized clocks (e.g. GPS based), or an interval to be decremented at each node that the message arrives at using the local clock of that node.

kdm_i is used to keep track of delivered messages using a hash table where the keys are the IDs of the messages. These records are exchanged at each meeting and all messages known as delivered are subsequently deleted from the message queue. The size of kdm_i will be kept to a minimum using the message time-to-live (TTL) parameter as described in the algorithm shown below.

C. Contact window

Before sending or relaying a message the algorithm computes the size of largest transferable message s_{max} between 2 meeting neighbors. This is used to avoid sending messages that have definitely no chance of getting transmitted in the given contact window, thereby allocating resources to those that do. Both energy and bandwidth are optimized by the following 2 steps: 1)

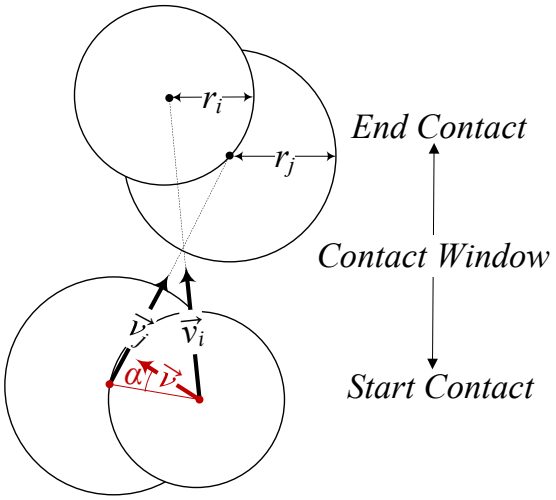

Figure 2. Estimation of the contact window

keeping messages in utility/bit order and 2) selection of messages that fit into s_{max}.

Given 2 nodes advancing at a vectorial speed of $\overrightarrow{v_i}$ and $\overrightarrow{v_j}$ respectively, having the radio range r_i and r_j, we can calculate the contact window time t_{cw} as being:

$$t_{cw} = \frac{2 * min(r_i, r_j) * \cos\alpha}{|\overrightarrow{v_i} - \overrightarrow{v_j}|} \quad (1)$$

where α is the angle between the relative speed $\overrightarrow{v} = \overrightarrow{v_i} - \overrightarrow{v_j}$ and the line connecting the two nodes at contact time, as depicted in Figure 2 in which dashed trajectories denote movement of nodes. This equation assumes of course that our nodes are confined to a 2D space, that radio ranges are perfect circles and that no signal obstruction, diffusion or scattering occurs. Under these assumptions nodes will be in contact as long as distance between them will not exceed the minimum radio range $min(r_i, r_j)$. In ORWAR each node estimates the contact window from node speed ($\overrightarrow{v_i}$, $\overrightarrow{v_j}$) and transmission range (r_i, r_j) using equation (1).

Of course, mobility implies that nodes can change speed or movement path during a given transmission. If the actual contact window is different from calculated t_{cw} then it is possible that the transmission of some selected message will fail or the selection is suboptimal. Although these cases cannot be avoided, calculating the fittest message to relay is by far a better solution than randomly taking any. Moreover, in some scenarios, e.g. in a city where nodes (cars, pedestrians) have mostly rectilinear trajectories (given by streets) we expect that velocity will be mostly constant for the short interval of the contact.

Having calculated the contact window, maximum exchangeable message s_{max} can be then easily calculated from device radio properties (b = data rate) as follows:

$$s_{max} = b * t_{cw} \quad (2)$$

By preventing the node from transmitting a message which has no chance to complete, ORWAR achieves two objectives: 1) limiting overhead in terms of bandwidth, and 2) conserving power as radio energy is not wasted for messages that cannot be sent anyway.

D. Algorithm description

The pseudo-code for the algorithm is described in Figure 3. The algorithm has two main parts: generation of a packet from an application, and exchange of metadata and forwarding at each new encounter. The former leads to insertion of the new message in the mq of the node in the order of utility/bit. The latter, data exchange and forwarding, has a number of phases. First, each node updates its kdm list using the knowledge of its neighbors in their respective kdm. Known delivered messages that appear in the node's mq are then deleted. Furthermore, messages that are older than their stipulated TTL are removed from the queue in order to stop their spreading in the network.

Second, the contact window for each pair of nodes at the encounter is computed. This contact window is used to directly deliver messages that are destined for a neighbor first. Then, messages held in the node queue are forwarded as replicas in the order of utility/bit. This continues until the contact window ends.

At each new hop messages are replicated as follows. Similar to binary SprayAndWait [5], L_k is divided by 2 at each replication. The initial value of L_k is chosen according to Table I, where L and Δ are algorithm parameters.

TABLE I.
INITIAL MESSAGE COPIES AS A UTILITY FUNCTION

Priority Class	Utility	L_k=# message copies
High	3	L + Δ
Medium	2	L
Low	1	L - Δ

IV. EVALUATION

A. Simulation setup

We evaluate the performance of ORWAR in comparison with five well-known delay-tolerant network routing protocols: MaxProp [6], SprayAndWait [5], Prophet [7], Epidemic [13] and DirectDelivery. We use ONE (Opportunistic Network Environment) [28], a powerful tool for generating mobility traces, running DTN simulations with different routing protocols, visualizing simulations interactively in real time, and presenting the results after their completion. ONE version 1.3 comes with the following protocol implementations: MaxProp, SprayAndWait, Prophet, Epidemic and DirectDelivery and we have run the evaluation using these shipped protocol versions. As both SprayAndWait and ORWAR use fixed number of replicas, we make sure both are run in the evaluation with the same replication factor (L=6). Since messages are evenly distributed between 3 utility classes in our experiment, the total (maximum) number of copies in the

For each node i:
- $\overrightarrow{v_i}$ // node vectorial speed
- r_i // node radio range
- kdm_i // known delivered messages
- mq_i // current message queue

For each message m_k:
- id_k // message id
- u_k // message utility
- s_k // message size
- L_k // message # copies

foreach *initiation of message* m_k **do**
  insert m_k in mq_i (based on u_k/s_k);

foreach *meeting between* i *and* j **do**
  `// signaling`
  send $\overrightarrow{v_i}, r_i, kdm_i$ to j;
  receive $\overrightarrow{v_j}, r_j, kdm_j$ from j;
  $kdm_i = kdm_i \cup kdm_j$;
  remove m_k from mq_i if $id_k \in kdm_i$;
  if TTL_k *expired* **then**
    remove m_k from mq_i;
    remove id_k from kdm_i;
  `// from equation (1) and (2)`
  $s_{max} = \frac{2 * min(r_i, r_j) * cos\alpha}{|\overrightarrow{v_i} - \overrightarrow{v_j}|} * b$;
  `// sending thread`
  foreach $m_k \in mq_i$, *if* $s_k < s_{max}$ **do**
    if *destination*(m_k)=j **then**
      `// direct delivery`
      send m_k to j;
      $s_{max} = s_{max} - s_k$;
    else if $L_k > 1$ **then**
      `// j is custodian`
      send m_k with $L_k/2$ to j;
      $L_k = L_k/2$;
      $s_{max} = s_{max} - s_k$;
  `// receiving thread`
  foreach m *received from* j **do**
    if $m = ACK_k \wedge j$=*destination*(m_k) **then**
      insert id_k into kdm_i;
      remove m_k from mq_i;
    else if $m = m_k$ **then**
      `// data messages`
      send ACK_k to j;
      $s_{max} = s_{max} - s_k$;
      if $s_k <$ *space available in* mq_i **then**
        insert m_k in mq_i (based on u_k/s_k);
      else if $\exists\, m_n \in mq_i$ *such that* $(u_k/s_k > u_n/s_n) \wedge (s_k < \sum_{i=n}^{last} s_i)$ **then**
        delete m_n to m_{last} in mq_i ;
        insert m_k in mq_i (based on u_k/s_k);
      else
        drop m_k;

Figure 3. ORWAR pseudo-code

system is not different from SprayAndWait, thus giving us comparable results. ORWAR just applies a bigger replication factor (L + Δ) for high utility messages and a smaller one (L - Δ) for low utility messages. In our work we experimentally found the ideal Δ as being about L/3, thus in our evaluation $\Delta = 2$. Prophet [7] is run with the following parameters: delivery predictability $P_{(a,b)} = 0.75$, scaling constant $\beta = 0.25$ and aging constant $\gamma = 0.98$.

In the evaluations below we consider a city setup with 126 nodes (80 pedestrians, 40 cars, 6 trams) sharing a 4500m x 3500m playground. All input parameters in this section were chosen with the following goals in mind: 1) as close as possible to real-life (e.g. speeds), and 2) in conformity with the recommended parameters, such as those suggested by comparable protocols (e.g. replication factor L). Every point plotted in the figures of this section is the result of 10 measurements where we vary the initial node position and initial direction of movement. The confidence interval is relatively small, i.e. 1-4% of the average value. We assume that each node has a network interface allowing a transmission range of 10m for pedestrians and 20m for cars and trams. For both cars and trams we consider a transmission speed of 250kBps (2Mbps). Buffers are considered to be 5MB except for trams which are 50 MB. The mobility pattern is close to reality, pedestrians, cars and trams follow a map-based movement. Cars drive only on roads and trams run only on their well-defined itinerary (we kept the Helsinki map and the original setup in order to maintain comparable results). Speeds for cars are set in the interval [10, 50] km/h and for pedestrians [1.8, 5.4] km/h and pauses are randomized. The network is still very sparse, with the accumulated transmission area for all nodes being 0.25% of the playground, and total meeting time accounts for about 3% of elapsed time. Each simulation runs for 12 hours and in our setup message TTL is considered to be infinite. We consider a mix of bundle sizes corresponding to:

- 1000 short messages averaged at 100B,
- 1000 documents averaged at 10kB,
- 1000 multimedia files averaged at 1MB.

Size distribution is shown in Figure 4. Every message comes with a constant utility which is evenly distributed over size classes, i.e. every size class (short messages, documents and multimedia) has an equal number of bundles of utility 1, 2 and 3. In what follows we refer to this size distribution as S.

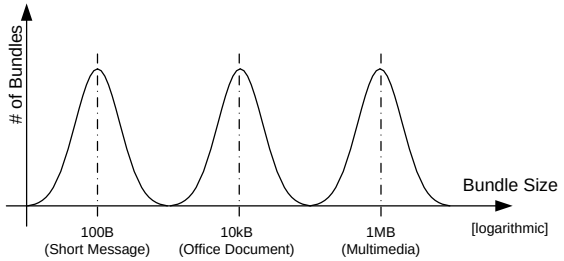

Figure 4. Size distribution in standard message set (S)

B. Message size implications

We run the first experiment in order to analyze the impact of message size. Starting from the standard message size distribution S, we gradually divide the size and increase the number of messages. That is, we start injecting 3000 messages at initial size, then 6000 messages where message size is halved, finishing with 30000 messages with the initial size divided by 10. Thus, at every simulation the same total amount of data is injected into the system.

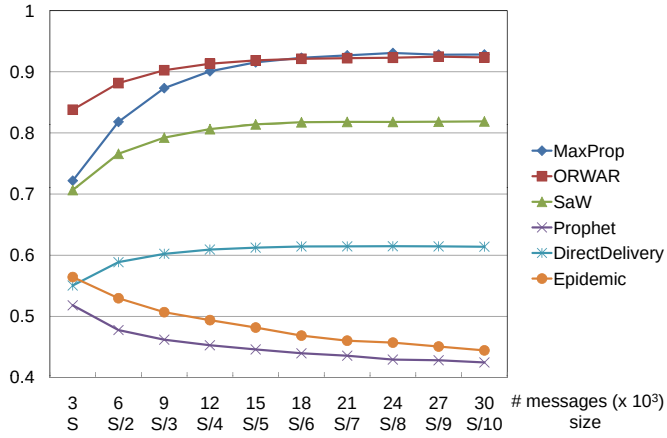

Figure 5. Delivery ratio versus message size

Figure 5 shows that ORWAR has the best delivery ratio and performs better when bigger messages are injected into the system. When analyzing overheads, defined here as the number of transmitted bundles divided by the number of messages delivered to destination, Figure 6 shows that ORWAR has the lowest overhead after DirectDelivery. Moreover, it compares favorably with SprayAndWait by a margin of roughly 10% which can be explained by the fact that ORWAR diminishes partial transmissions.

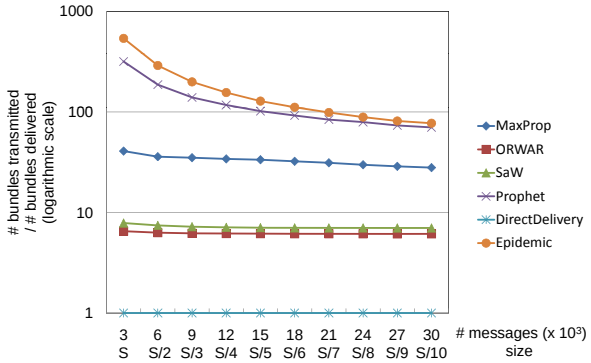

Figure 6. Overhead versus message size

From these 2 figures we can conclude that ORWAR has the best overall performance over the comparable algorithms. Moreover, when larger messages are to be transmitted, ORWAR advantages increase. Thus, it appears as an effective alternative for cases where larger messages are to be transmitted and fragmentation is not available/desirable. Based on the overview of how ORWAR compares with the other 5 schemes, in the subsequent sections of the paper we are going to present only the best performing protocols: MaxProp, ORWAR and SprayAndWait.

C. Energy implications

The main goal in designing ORWAR was diminishing partial transmissions in order to save energy. We have shown that ORWAR has 10% less overhead than SprayAndWait and much less overhead compared with other schemes. Although protocol overhead defined as the number of transmitted messages divided by the number of delivered messages is widely used in the literature, simply counting the number of messages would potentially disadvantage protocols that use (small sized) acknowledgements, such as MaxProp, over those that do not, such as SprayAndWait. Instead of focusing on a number of bundles which might have very different sizes, we focus on the total amount of data transmitted, aborted or dropped. Moreover, overheads are related to different mechanisms: connection abortions (i.e. neighbor out of reach whilst sending message, wireless contention), messages sent but dropped (i.e. buffer shortage at custodians) or inherent to replication factor (number of copies in the system). By estimating contact window and estimating s_{max}, ORWAR tries to diminish transmission abortions. Therefore, we consider this as an appropriate metric for measuring "waste cost".

In the next experiment we increase network load by gradually increasing the number of messages whilst keeping the average message size - as defined by S. We start with 3000 messages injected within 12 hours and show the effect of higher load - 6000, 9000, 12000 messages, up to 60000 messages.

Figure 7 depicts the accumulated size of aborted transmissions on the y axis.

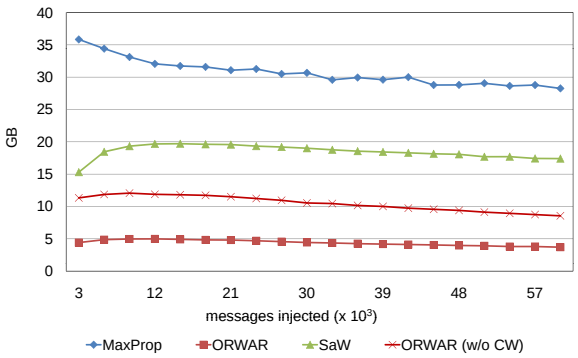

Figure 7. Partial transmission total size versus load

In addition to MaxProp and SprayAndWait, we plot in Figure 7 a new curve - ORWAR without the module responsible for estimating contact window. Thus, we can directly measure the added value of the contact window estimation, and separate that from other ORWAR mechanisms, such as queue management or utility-based replication. We can measure an improvement by a 4

to 6 factor against both MaxProp and SprayAndWait. The remaining aborted transmissions in ORWAR can be explained by nodes changing trajectories or speed during message transmission, or by wireless contention. Obviously these cases cannot be avoided, and we show that computing the contact window gives a 50% reduction of aborted transmissions over not calculating it at all.

Note that in our scenario the overhead of transmitting the speed and radio range at each contact is negligible (in a 10^{-5} magnitude order) compared with the benefits of partial transmission savings. Consider 4 bytes per message (for vectorial speed) times 3×10^4 contacts that we have in our simulation and compare them with the Gigabytes savings as presented in Figure 7.

As ORWAR computes the most valuable message to be sent in a given meeting context, we expect that it will not always send small messages at the cost of dropping the bigger ones. It would be unacceptable that energy savings would be at the cost of delivering less data. To verify this, we plot in Figure 8 the throughput achieved during 12 hours.

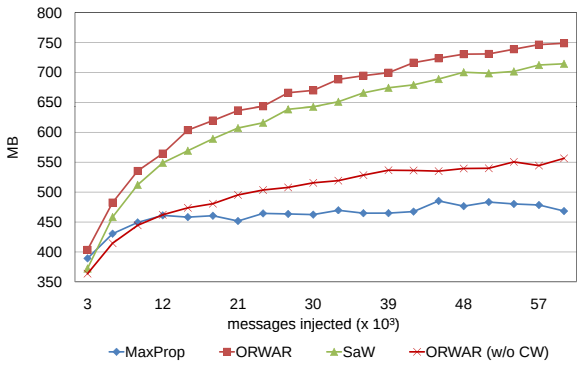

Figure 8. Throughput versus load

It shows that ORWAR sends 10-90% more data than SprayAndWait and MaxProp over the same interval of time (12h). It also shows that throughput is increased at the same rate when using contact window estimation. To conclude, in this section we have shown that the accumulated volume of aborted messages is very favorable against competing protocols and, most importantly, limiting the biggest message to be sent within s_{max} gives a 50% reduction over not limiting it at all. By estimating the contact window and selecting the "fittest" message to be sent, ORWAR will not only diminish partial transmissions, but it will also increase throughput.

D. Load implications

In Figure 9 we return to the study of the delivery ratio and increase the load on the network. We inject 3000, 6000, and up to 60000 messages over 12h, keeping the message distribution S.

ORWAR not only has the best overall delivery ratio but also its relative performance compared to other protocols increases at higher loads. The explanation is that ORWAR

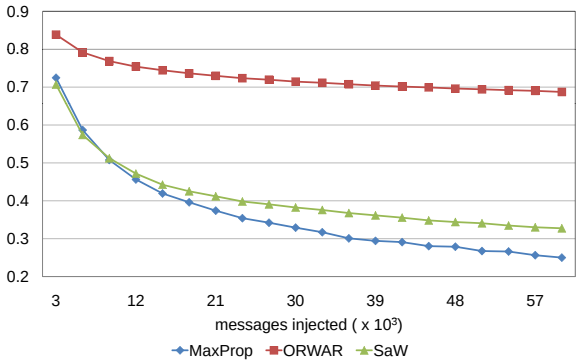

Figure 9. Delivery ratio versus load

maintains a low overhead which pays off when the network is congested. Other mechanisms, such as effective queue management, utility-based replication, usage of acknowledgements, and contact window estimation are also contributing.

Congestion occurs when critical resources, such as buffer space or transmission window, are finite during increased loads. Basically, adding more buffer space on nodes will only change the point where the system becomes overloaded, but will not affect protocol behavior in other ways. However, we believe that in mobile scenarios buffer space is the least problematic of the scarce resources.

As far as latency is concerned, which we study in Figure 10, ORWAR performs second best after SprayAndWait. This is reasonable as the messages will stay longer in the buffers in order to get the suitable contact window.

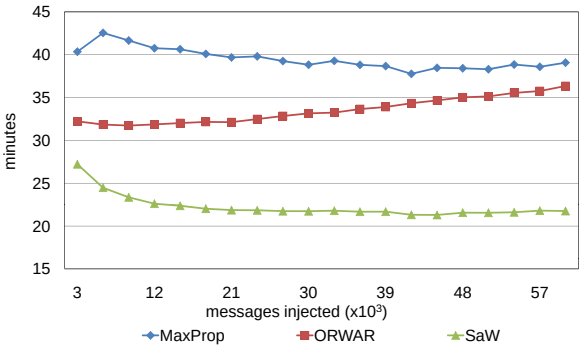

Figure 10. Median latency versus load

A major benefit of ORWAR is demonstrated in Figure 11 where we plot the accumulated utility.

Note that utility is accounted for only if the respective bundle reaches destination. Because messages are treated differently according to their utility, that is, more resources are available for high utility messages, ORWAR obtains a higher accumulated utility over the same interval of time. Note that we compute accumulated utility in the same way for all algorithms. We recall that 3 utility classes are used in these experiments, thus accumulated utility is computed as a function of the number of messages delivered in each utility class, as follows:

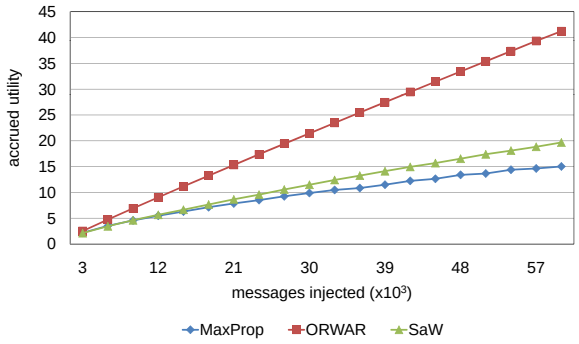

Figure 11. Accumulated utility versus load

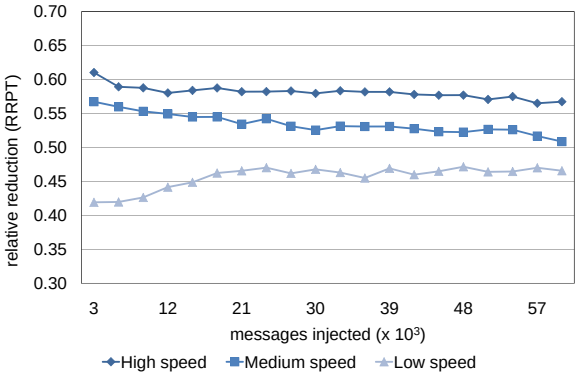

Figure 12. Partial transmissions versus load

$$AccUtility = \sum_{3}^{1} u_i \times n_i \quad (3)$$

where: u_i = message utility class (see table 1)
n_i = # messages delivered within the class

The higher accumulated utility for ORWAR is explained by a higher replication rate for high utility messages, and deleting low utility messages first. All in all, ORWAR shows a better performance compared to the best 2 of the alternative protocols.

E. Mobility implications

We are interested in how node speed affects ORWAR performance and more precisely when related to contact window estimation. We have shown in Figures 7 and 8 that by sending only messages that have a good chance to arrive within the contact window, we diminish partial transmissions without diminishing delivery ratio and even increasing throughput. Those gains correspond to a medium speed in table II. We are going to extend these measurements to other speeds.

TABLE II.
DIFFERENT SPEEDS TEST BED

Speed	Pedestrians	Cars and trams
High	3.9-10.8 km/h	20-100 km/h
Medium	1.8-5.4 km/h	10-50 km/h
Low	0.8-3.7 km/h	5-25 km/h

Figure 12 shows the relative reduction of partial transmissions (RRPT) when using the contact window, defined as:

$$RRPT = 1 - S_{ORWAR}/S_{OwoCW} \quad (4)$$

where:
S_{ORWAR} = total data volume lost due to message abortion using ORWAR with Contact Window.
S_{OwoCW} = total data volume lost due to message abortion using ORWAR without Contact Window.

Irrespective of speed and message load we see that the relative reduction of partial transmissions is between 40% and 60%. It also appears that gains are more significant at higher speed. At medium speed - which is the most likely situation to find in a city scenario - the gain is still significant (around 55%).

V. FRAGMENTATION IN A HYBRID SETTING

In the previous section we have presented ORWAR as an efficient routing solution when fragmentation is not available. We have also argued why we expect big messages to be the rule rather than the exception in DTN.

In this section we consider what gains can be made if fragmentation is indeed an option and some or all nodes support it. We will show that delivery ratio can be improved by 30-35% just by using fragmentation/redundancy with ORWAR. These figures are in absence of another complementary infrastructure network. We then go on and study the benefit of having some help from another infrastructure (e.g. cellular) to boost the benefits of fragmentation in a hybrid DTN context.

In many papers on DTN in a mobile context we find delivery ratios below 90%, and this is barely acceptable in real life scenarios. From a practical point of view, most mobile devices nowadays come with two or more network interfaces. Interfaces such as WIFI or Bluetooth can be used in a first phase on a partitioned mobile ad-hoc base. Others, such as UMTS/HSPA, WIMAX can then be used only to complete what was not delivered in the first phase. This section presents a scheme that combines benefits from a cheap but less reliable ad-hoc network (DTN) and a high-cost but reliable alternative network (infrastructure).

We consider a system consisting of mobile nodes (cars, pedestrians or trams), each having 2 interfaces, one for access to the partitioned mobile ad-hoc network and the other to the infrastructure network. We assume that all nodes move and are within the reach of the infrastructure access point. Figure 13 explains the proposed mechanism materialized in 4 steps:

1) **Fragmentation/Coding**: A bundle of size s is encoded into n, s/m sized data-blocks (where $n \geq m$). Encoding is based on Reed-Solomon or Tornado codes [27] which allow reconstruction of the original message from any m different fragments

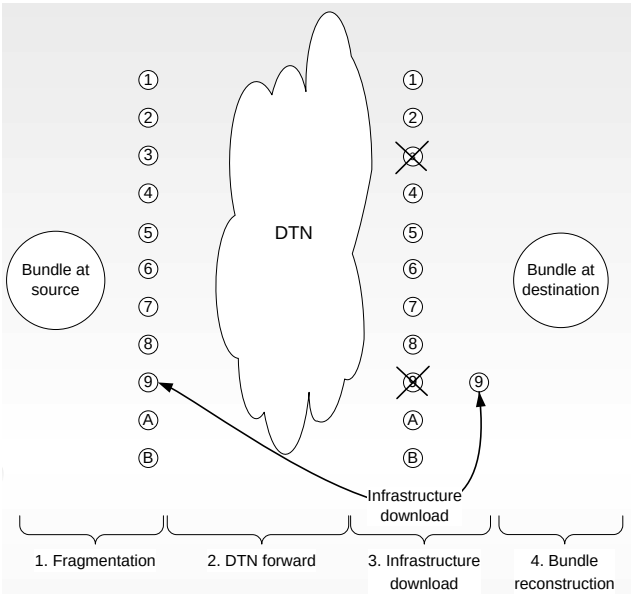

Figure 13. Hybrid mechanism

arrived at destination. In our example, a 500kB message is encoded in $n = 11$ fragments (identified in Figure 13 as messages 1 to B, in hexadecimal) each of 50kB, meaning that a 10% redundancy is used ($m = 10$).

2) **DTN forward**: All these fragments are forwarded independently over a DTN network; some can arrive at destination whilst others cannot. In our example fragments 3 and 9 failed to arrive at destination.
3) **Infrastructure download**: Destination node requires m different fragments for reconstructing the message and decides to download the rest via infrastructure (the pull phase). In our example only 1 fragment is still needed (10 different messages at destination) and fragment 9 is chosen.
4) **Reconstruction**: Message is reconstructed from the m different fragments available ($m = 10$).

Note that most of the transmissions are done over the DTN network and that infrastructure is used only to complete when the first fails to deliver. Fragmentation and coding are the key to the scheme, allowing infrastructure download to be kept at a minimum. Steps 3 and 4 require some glue code to be added below the application layer at each end. It is responsible for selective pulling at the receiver end and response to pull at the sender end.

This scheme is orthogonal to the choice of routing algorithm but in the next subsection we will combine it with ORWAR and later return to a comparison with SprayAndWait where the same mechanism of fragmentation/redundancy was also adapted.

A. Fragmentation size

We use proactive fragmentation, meaning fragmentation is decided in advance at the source node. We have identified 3 criteria when selecting fragmentation size:

1) Fragmentation size (s_f) should not exceed expected maximum transferable size determined by the contact window. This corresponds to the s_{max} already introduced in *ORWAR* and this limit is relevant for every contact in the network. In other words:
$$s_f < min(s_{max,i,j})$$
where: s_f = size used for fragmentation
$s_{max,i,j}$ = estimated max. transferable message between node i and node j
2) Using fine grained fragments helps minimize infrastructure download. This scheme allows downloading via the infrastructure of only the fragments missing. As the size of one fragment decreases, the system will download a smaller volume through the infrastructure, which is helped by a higher fragmentation degree.
3) Too much fragmentation increases latency. The bundle is reconstructed at destination when *the last of the* m fragments arrives at destination. When many small fragments are used, more fragments implies multiplications of paths, whereby end-to-end latency corresponds to the highest latency path.

From a DTN perspective one of the following 3 cases may occur:

- **Complete deliveries (CD)** - when the bundle can be reconstructed at destination from fragments arriving only through DTN (at least m out of n fragments were delivered at destination)
- **Partial deliveries (PD)** - when k fragments arrived at destination ($1 \leq k < m$), meaning that the bundle can be reconstructed only by downloading the remaining necessary fragments via infrastructure
- **No delivery (ND)** - meaning that no fragment was delivered at destination. This case is the worst case because the destination node does not know that a bundle has been sent, therefore it does not know when to initiate an infrastructure download.

We evaluate the hybrid scheme in ONE [28] in which we have implemented fragmentation. In the following 3 plots we inject 1000 messages where size distribution is the previously defined *S*. We vary fragmentation size from no fragmentation at all up to 1kB fragment size whilst redundancy remains constant at 10%. Node mobility is map-based and corresponds to the half-day model defined in ONE. All other settings are those referred to in section IV except the replication factor L and the protocol parameter Δ. In this section we use: $L = 12$ and $\Delta = 4$. This is due to the fact that with smaller fragments we can afford to have more copies.

In the following 3 Figures we are going to highlight in light colours the best results (50 kB fragments) and in dark colours the results in absence of fragmentation. All results are the average of 10 runs with different seeds (initial node positioning, direction and speed).

Figure 14 shows the total size of all completed bundles CD arrived at destination via DTN mechanisms, meaning that partial deliveries PD are excluded from this count. We note that the delivery ratio reaches a maximum when

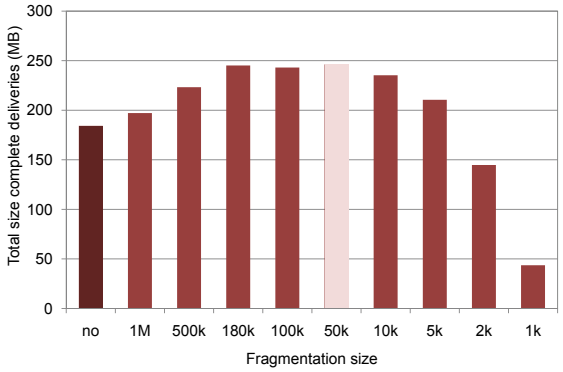

Figure 14. Complete deliveries (CD) Total Size vs. Fragmentation size

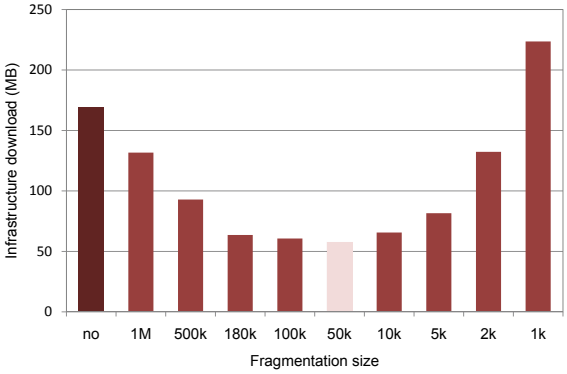

Figure 16. Infrastructure download vs. Fragmentation size

fragment size is between 180kB and 50 kB. Indeed, calculating what would be the worst case contact window in this scenario, we find that this corresponds to 2 cars running in opposite directions at 50 km/h, which also corresponds to 180kB.

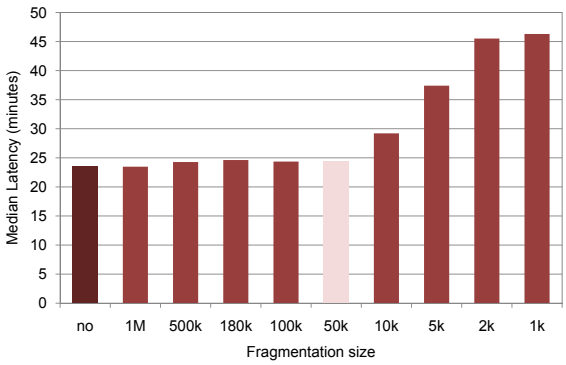

Figure 15. Latency vs. Fragmentation size

In Figure 15 we confirm that, due to path multiplication, latency is increased as fragmentation increases. With such an increase in latency and in presence of a bounded TTL we would get more downloads via the expensive infrastructure.

In Figure 16 we show the total size of fragments downloaded via the infrastructure and we note that a fragmentation range from 50kB to 180kB remains interesting but, when fine grained fragmentation is used, this diminishes the part retrieved via infrastructure, as expected.

The benefits of fragmentation are obvious here: instead of downloading 170MB via the infrastructure (which would be the total amount of lost messages in absence of fragmentation - as shown by the chartbar in Figure 16) only a third of this is pulled when using a hybrid scheme with 50kB sized fragmentation.

B. Load implications

In previous sections we have shown that ORWAR performs well against other protocols when increasing load is injected in the system. This is due to taking into account message utility when forwarding decisions are made. However, fragmentation implies that the contact window has now a negligible effect as we are dealing with small fragments (50kB) where almost no partial transmissions are encountered. Therefore, when fragmentation is used, we would like to demonstrate that the competitive advantage of ORWAR is at least maintained.

In order to evaluate this in the fragmentation context, we compare ORWAR with SprayAndWait. Load is simulated by increasing the inital message size by factors of up to 10 and by keeping fragmentation size constant (50kB). As load grows, the number of partial deliveries (PD) increases at the expense of complete deliveries (CD). Although fewer and fewer complete bundles are delivered at destination, by using the hybrid DTN /Infrastructure scheme, finally all of them can be reconstructed at destination by pulling, despite the expensive infrastructure download.

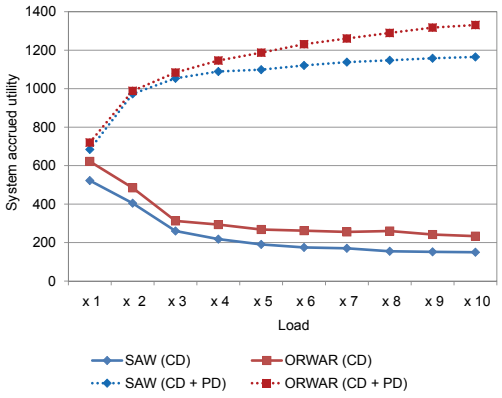

Figure 17. System level accrued utility vs. load

In Figure 17 we plot system utility taking into account complete deliveries (CD) but also total deliveries (CD + PD). Partial deliveries are also beneficial as they allow diminishing costly download via infrastructure. We notice that at higher load, fewer and fewer bundles are completely delivered to destination. Instead, there are increasingly more bundles in PD. ORWAR shows better resilience to load than SprayAndWait. System-wide utility increases with added load, with roughly 15-25% improvement over SprayAndWait. This shows that utility driven

routing decisions are worthwhile, especially in highly loaded scenarios.

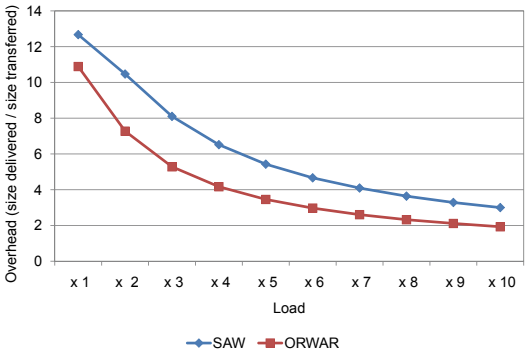

Figure 18. Overhead vs. load

ORWAR was designed with the goal of diminishing energy consumption by limiting overheads. Next we look at overhead (defined as total volume transmitted / total volume arrived at destination) retaining only the DTN part. Figure 18 shows that although ORWAR delivers more in terms of both delivery ratio and utility, the number of transmissions remains relatively small. The competitive advantage against SprayAndWait is also about 20% and rather independent of load.

In this subsection we have shown that, even when fragment size is smaller than biggest transferable message over the given contact window so that contact window calculation makes less benefit, other ORWAR built-in mechanisms remain of interest.

VI. Conclusion

The massive body of work on routing protocols in mobile ad hoc networking assumes end-to-end connectivity, a property that is believed to be absent in several applications of such networks. In this paper we address the routing problem in sparse networks and propose a new routing algorithm that exploits the store-and-forward mechanism in delay-tolerant networking. The paper proposes an algorithm, ORWAR, that combines selected replication and delivery acknowledgment from existing routing algorithms with two novel features in the DTN context: (1) the use of message utility as a parameter in the selection of replicated messages as well as buffer management, and (2) the use of estimated contact window for selecting the optimal message to forward at any opportunity. Moreover, we analyze effects of fragmentation on ORWAR and show that this substantially improves performance especially when combined with pulling of the last few fragments via a complementary infrastructure network.

In a simulation setting we have illustrated the superior performance of the algorithm in comparison with existing algorithms (Direct Delivery, Epidemic, Prophet, MaxProp and SprayAndWait), including detailed studies in relation to the closest algorithms (SprayAndWait and MaxProp). We added the notion of utility to the messages generated by ONE, where three classes of messages have been generated with equal probability. The analysis shows that ORWAR has a similar or better delivery ratio as compared to MaxProp and SprayAndWait, while creating far less overhead. It also shows a 10% higher delivery ratio compared to SprayAndWait with an overhead that is around 10% lower. Because of producing little overhead, ORWAR relative performance will increase at higher loads. We show that the benefit of using ORWAR is especially significant when having to deal with larger messages. To our knowledge, this is the first routing scheme well-suited for large message sizes, and taking account of resource optimization at the same time. Another gain from the use of ORWAR results when the accumulated utility is used as a metric for evaluation, whereby a gain of 25-50% is demonstrated compared to baseline algorithms. The added benefit increases as the load increases in the network.

A further study of a hybrid DTN/Infrastucture scheme was demonstrated to lead to most of the traffic being routed via a DTN network, allowing pulling of remaining lost fragments via infrastructure.

This work can be extended in several directions. An obvious extension of the work is the validation of our approach on an emulated network of physical nodes. In particular, whether the real-time estimation of the contact window indeed leads to optimized packet transmission is the next step of the study. Second, the algorithm can be made more adaptive to changing network conditions and traffic characteristics by producing replications of messages at different quality of service levels according to dynamic network conditions. One approach would be to take account of the diminishing TTL as messages are selected for forwarding. Another would be to take account of time varying utility functions.

Studying algorithm performance within other DTN applications e.g. in a disaster scenario is also an interesting direction to pursue.

Acknowledgments

We thank Mikael Asplund and Erik Kuiper for valuable feedback concerning this work and the reviewers for their detailed and valuable comments.

References

[1] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss., "Delay-tolerant networking architecture (RFC4838)," April 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4838.txt

[2] F. Whartman, "Delay-Tolerant Networks (DTNs), A Tutorial, V1.1," May 2003. [Online]. Available: http://www.ipnsig.org/reports/DTN Tutorial11.pdf

[3] R. Kravets and P. Krishnan, "Power Management Techniques for Mobile Communication," in *Proceedings of IEEE International Conference on Mobile Computing and Networking (MobiCom 1998)*, 1998, pp. 157–168.

[4] "Delay-Tolerant Networking Research Group. DTN reference implementation, Version 2.5," 2007. [Online]. Available: http://www.dtnrg.org/docs/code

[5] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, August 2005, pp. 252–259.

[6] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "MaxProp: Routing for Vehicle-Based Disruption- Tolerant Networks," in *Proceedings of the* 25^{th} *IEEE International Conference on Computer Communications*, April 2006, pp. 1–11.

[7] A. Lindgren, A. Doria, and O. Schelén, "Probabilistic routing in intermittently connected networks," *Lecture Notes in Computer Science*, vol. 3126, pp. 239–254, January 2004.

[8] G. Sandulescu and S. Nadjm-Tehrani, "Opportunistic Routing with Window-aware Adaptive Replication," in *Proceedings of the* 4^{th} *Asian Conference on Internet Engineering*, November 2008, pp. 103–112.

[9] J. Ott and D. Kutscher, "A disconnection-tolerant transport for drive-thru Internet environments," in *Proceedings of* 24^{th} *Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM05)*, vol. 3, March 2005, pp. 1849–1862.

[10] N. Shank, B. Sokal, M. Hayes, and C. Vetrano, "Human Services Data Standards: Current progress and Future Visions in Crisis Response," in *Proceedings of International Community on information systems for crysis response and management (ISCRAM08)*, May 2008.

[11] J. Ott and D. Kutscher, "Applying DTN to Mobile Internet Access: An Experiment with HTTP," Universität Bremen," TRTZI-050701, July 2005.

[12] E. Jones, L. Lily, J. K. Schmidke, and P. Ward, "Practical routing in delay-tolerant networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 8, pp. 943–959, August 2007.

[13] T. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks," Duke University," CS-2000-06, July 2000.

[14] C. Zhang, "Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay tolerant networks: Overview and Challenges," *IEEE Communication Surveys*, vol. 8, no. 1, pp. 24–37, 2006.

[15] J. Hyewon, M. Ammar, M. Corner, and E. Zegura, "Hierarchical Power Management in Disruption Tolerant Networks with Traffic-Aware Optimization," in *Proceedings of the 2006 SIGCOMM workshop on Challenged networks*, September 2006, pp. 245–252.

[16] W. Wang, V. Srinivasan, and M. Motani, "Adaptive contact probing mechanisms for delay tolerant applications," in *Proceedings of the* 13^{th} *annual ACM international conference on Mobile computing and networking (MobiCom07)*, September 2007, pp. 230–241.

[17] P. Juang, H. Oki, Y.Wang, M. Martonosi, D. Rubenstein, and L. Peh, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 10, pp. 96–107, October 2002.

[18] S. Jain, R. Shah, W. Brunnette, G. Borriello, and S. Roy, "Exploiting Mobility for Energy Efficient Data Collection in Wireless Sensor Networks," *Mobile Networks and Applications*, vol. 11, no. 3, pp. 327–339, June 2006.

[19] C. Curescu and S. Nadjm-Tehrani, "A Bidding Algorithm for Optimised Utility-based Resource Allocation in Ad hoc Networks," *IEEE Transactions on Mobile Computing*, vol. 7, pp. 1397–1414, December 2008.

[20] A. Balasubramanian, B. N. Levine, and A. Venkataramani, "DTN Routing as a Resource Allocation Problem," *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 373–384, August 2007.

[21] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility," in *Proceedings of the* 5^{th} *IEEE International Conference on Pervasive Computing and Communications Workshops*, March 2007, pp. 79–85.

[22] V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot, "Diversity of forwarding paths in pocket switched networks," in *Proceedings of the* 7^{th} *ACM SIGCOMM conference on Internet measurement*, October 2007, pp. 161–174.

[23] K. Fall and S. Farrell, "DTN: an architectural retrospective," *IEEE Journal on Selected Areas in Communications*, vol. 26, pp. 828–836, June 2008.

[24] M. Pitkänen, A. Kernen, and J. Ott, "Message Fragmentation in Opportunistic DTNs," in *Proceedings of International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2008)*, 2008.

[25] S. Jain, M. Demmer, R. Patra, and K. Fall, "Using Redundancy to Cope with Failures in a Delay Tolerant Network," *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 109–120, October 2005.

[26] M. Pitkänen and J. Ott, "Redundancy and Distributed Caching in Mobile DTNs," in *Proceedings of the* 2^{nd} *ACM/IEEE international workshop on Mobility in the evolving internet architecture*, August 2007.

[27] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *ACM SIGCOMM Computer Communication Review*, pp. 56–67, 1998.

[28] "Homepage of Opportunistic Network Environment (ONE), version 1.3." [Online]. Available: http://www.netlab.tkk.fi/%7Ejo/dtn/

Gabriel Sandulescu is currently a Ph.D. candidate at the University of Luxembourg. He obtained a Diploma in Electrotechnical Engineering from the Politehnica University of Bucharest in 1992, and has worked since as a software engineer and consultant with various companies, mainly in the Luxembourg financial sector. His research interests include new service paradigms in conjunction with network convergence, resource allocation in mobile networks, and distributed systems.

Simin Nadjm-Tehrani is a full professor at Linköping University (LiU), Sweden. She leads the real-time systems laboratory (RTSLAB) at the department of Computer and Information Science, LiU, since 2000. She obtained her BSc from Manchester University, England, and her PhD from Linköping University in 1994. Her current research interests lie in the area of dependable distributed systems and networks, including analysis of safety and fault tolerance, reliable communication and measuring availability in presence of failures, attacks and overloads.