Opportunistic DTN Routing with Window-aware Adaptive Replication

Gabriel Sandulescu and Simin Nadjm-Tehrani*
Computer Science and Communication Research Unit
University of Luxembourg
{ gabriel.sandulescu, simin.nadjm-tehrani }@uni.lu

ABSTRACT

This paper presents ORWAR, a resource-efficient protocol for opportunistic routing in delay-tolerant networks. 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. As well as optimizing the use of bandwidth during overloads it helps to reduce energy consumption since partially transmitted messages are useless and waste transmission power. Another feature of the algorithm is the use of 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 the global accumulated utility is used as a system-wide performance metric. Simulations illustrate the benefit of our model and show that ORWAR provides lower overhead and higher delivery rate, as well as higher accumulated utility compared to a number of well-known algorithms (including Maxprop and SprayAndWait).

Categories and Subject Descriptors

C.2.2 [**Network Protocols**]: Routing protocols

General Terms

Algorithms, Design, Performance

*The second author was partially supported by Linköping University, Sweden, during preparation of this work.

AINTEC'08, November 18–20, 2008, Bangkok, Thailand.

Keywords

Routing, Delay-tolerant Networks, Opportunistic Networks

1. INTRODUCTION

Delay- or Disruption-tolerant Networks (DTN) aim to provide reliable communication in networks in which end-to-end connectivity cannot be assumed. Intermittent connectivity can be a result of high node mobility, low node density, intermittent power from energy management (on/off) schemes, short radio range, radio obstruction or malicious attacks [8].

DTN defines 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 the 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. Bundle layer is responsible for end-to-end delivery mechanism of messages, called virtual message forwarding [4, 22]. DTN uses a minimal conversational model so that DTN applications should be designed in a way to minimize the end-to-end transactions, using larger, self-contained messages. Although the large bundle size is beneficial in a store-and-forward context, this adds some new challenges when it comes to resource-constrained networking.

In this paper we provide effective use of resources at system level despite the large and variable sized messages at bundle layer. We demonstrate distributed mechanisms at node level which optimize the use of transmission power at relays, as well as bandwidth and memory during overloads. Note that bundle, message and packet are used interchangeably in the rest of the paper.

Our mechanism for efficient use of resources is based on two concepts: 1) minimising retransmissions, by relaying only the small enough messages at each encounter and 2) message differentiation, based on the notion of utilities. DTN architecture actually offers 3 relative priority classes (low, medium and high) which differenti-

ate traffic based upon an application's desire to affect the delivery urgency expressed by the message source. These have some impact on traffic contention as well as other resource allocation. For example, in the current reference implementation [6], 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 the 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 works by Spyroupoulos et al. [18]. 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 avoiding fragmentation. 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 [2], SprayAndWait [18], Prophet [13], among others. This paper evaluates our routing scheme in a city scenario, while evaluation of the scheme in disaster scenarios is a subject for future work.

In the next section we provide a background to routing in delay-tolerant networks and related problems. In section 3 we present ORWAR. Section 4 is devoted to comparative evaluation of the protocols and the paper ends with some conclusions and ideas for future work.

2. BACKGROUND AND RELATED WORK

2.1 General DTN routing strategies

Farell and Cahill in their survey of DTN routing place the algorithms on a closed circuit ranging over methods that assume full knowledge about the network and estimation-based techniques (Chapter 8.1, [8]). Jones et al. [11] describe two strategies for achieving higher delivery rates: a) knowledge about the network and b) replication. Knowledge about the network may vary between no knowledge at all and total knowledge. In the full knowledge case information about messages workload, contact schedule and buffer allowance is known in advance. In a no-knowledge scenario, a node can make decisions in a strictly opportunistic way, just by exploiting nodes in its vicinity - using neighborhood sensing. These strategies are simple, hard-coded in advance and similar for all nodes. Their implementations usually require minimal configuration and control messages. The disadvantage is that the strategy cannot adapt to different networks or conditions, so it may not make optimal decisions. A node might also need to know the complete future schedule of every contact in the network as possible in some interplanetary scenarios. Althrough this allows routing strategies to make very efficient use of the network resources by forwarding a message along the best path they are obviously not applicable to network with dynamic changes.

2.2 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. 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 [13], which makes use of historic encounters to estimate probability and only forward messages to some neighbor whose probability exceeds a certain threshold. The Epidemic protocol [20] was an early example where replication was used but this strategy works well only when message volume and node density are very low. Other protocols, like SprayAndWait [18] 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 the 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 [2], also uses multiple copies. Besides, Maxprop calculates the cost for each route as well as leveraging delivery notifications to purge old replicas. MaxProp is intended in storage and bandwidth-constrained environments; therefore, it purges messages that have a lower chance to be delivered.

2.3 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 [23] suggested in her survey, the success criteria of a routing scheme are still a research topic. Most protocols would still try to maximize delivery rate or to minimize transmission delay between source and des-

tination. However, there are other criteria to be taken into account. Reducing overhead, for example, is an important topic, especially in order to enforce energy efficiency. Even if bandwidth is assumed unlimited, 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 trying to deal with energy constraints in a DTN environment. One of the solutions proposed is to control radio wakeup intervals [9] or neighborhood sensing [21] by probing algorithms which trade off energy consumption against the probability of missing a contact. However, this will usually imply further degrading the connectivity. Another proposed solution is to architecture the network into multiple tiers as in Zebranet [12] or DataMules [10], thus maximizing energy savings at one single tier - albeit the most sensitive one - the sensor tier. However, such a network specialization is hard to implement in social, urban scenarios where heterogeneity is key property and association/dissociation of nodes to the network is very dynamic. Our approach to this problem is to select the message replica in an energy-aware manner. 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 [5] where construction of the route also enforces maximizing the accumulated utility for the whole network. In the context of DTNs, Balasubramanian et al. [1] use utility in order to optimize 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. Spyroupoulos et al. in [19] use utility to choose the fittest custodian nodes which would carry the message. However, in both papers in the context of DTN, there is no network-wide optimization of the accrued utility.

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

Another important aspect that we focus on is node density. Zhang [23] shows 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. On the other hand, Erramilli et al. [7] show that in homogenous and relatively dense setups found for example in Pocket Switched Networks different algorithms perform equally well in terms of delay and success rates. The corollary is that a protocol must be evaluated in the intended scenario.

2.4 Mobility implication for bundle size

Considering a network of nodes advancing at various speeds, it is obvious that the real available bandwidth is not solely driven by the device nominal bandwidth rate and that meeting dynamics matter. Ott and Kutscher [15], after carrying 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 there is a maximum size limit for the bundle to be exchanged depending on the relative speed. To transmit a bundle exceeding this size a node has no other alternative than waiting for a better contact opportunity (i.e. a node with lower relative speed) or to use proactive fragmentation. In a disaster scenario [17], 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 the rescue teams.

Another study by Ott and Kutscher [16] shows that application protocols are differently suited for mapping onto a DTN infrastructure; 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 in one message. In the absence of a DTN bundle layer limit for messages size, we can expect messages to get bigger in size. In our work we take the message size and relative speed into account when selecting the window for forwarding/replication.

3. CONTACT BASED ROUTING

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

3.1 Protocol design rationale

ORWAR uses local connectivity knowledge in order to route from source to destination. Connectivity knowledge is not known in advance but gathered from the vicinity on a peer-to-peer basis during contact. Specifically, we know neither message arrival rates nor meeting schedule, so routing is completely opportunistic. How-

ever, a node knows its own speed, direction, and its own location[1]. In presence of unlimited message sizes, we assume finite buffers for custodians. 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 an evaluation metric. Note that the unit for measuring utility is unimportant since it only reflects a global measure of benefit.
Second, we strive for high delivery ratio in partitioned networks. For this we use store-and-forward and replication mechanism in DTNs. In order to make the replication energy-efficient, we propose the decision to be based on both utility/bit for involved messages and local connectivity characteristics.

3.2 The ORWAR Algorithm

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 Spray and Wait mechanism presented by Spyroupoulos et al. in [18]. 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.

3.3 Message queues

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 2) a record of known delivered messages (kdm_i).

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 tail 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.

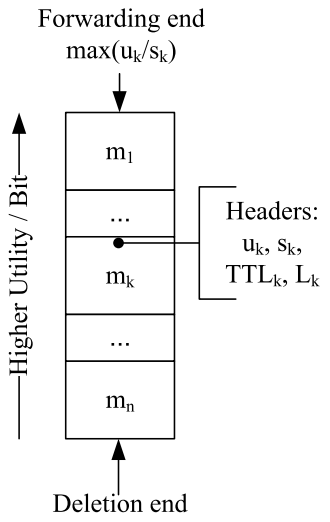

Figure 1: ORWAR queue

Every message header also includes L_k which denotes the intended number of message copies. Messages are replicated at each new hop and L_k is divided by 2 at each replication (similar to binary Spray and Wait [18]). The initial value of L_k is chosen according to Table 1, where L and Δ are algorithm parameters.

Table 1: 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

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.

[1]In fact, the algorithm can be easily changed in order to use relative location to all neighbors in radio range.

```
Constant: τ  //retention time in kdm
For each node i:
  Variables:
   x_i, y_i //node coordinates
   v_i   // node speed
   r_i   // node radio range
   kdm_i // known delivered messages
   mq_i  // current message queue

// original messages from applications
upon initiation of a message m_k
 insert m_k in mq_i (based on u_k/s_k)

at each meeting between i and j, at some
time t
  // Merge and delete delivered messages
  send kdm_i to j
  receive kdm_j
  remove from queue m_k ∈ kdm_i ∪ kdm_j
  remove from queue m_k with TTL_k > t
  kdm_i = kdm_i ∪ kdm_j
  remove m_k from kdm_i if TTL_k > t + τ
  s_max = compute_s_max (i,j)

while s_max > 0
// final delivery to j
  for each message in mq_i where dest(m_k)=j
      if s_k<s_max
        deliver m_k to j
        s_max = s_max - s_k
        // update kdm_j
        if m_k transmitted successfully
         insert m_k in kdm_i
         remove m_k from mq_i
  end for each message
// forward to j as custodian
  for each message in mq_i if s_k<s_max ∧ L_i>1
        if m_k ∉ mq_j then
        send m_k with L_k/2 to j
        L_ki= L_ki/2 in mq_i
         s_max = s_max - s_k
  end for each message
// receiving message m_k
  if mq_i is full
    if u_k/s_k > last(mq_i)
      replace last(mq_i) with m_k
  else
    insert m_k in mq_i (based on u_k/s_k)
  if destination(m_k) = i
     insert m_k in kdm_i
     remove m_i from mq_i
end at each meeting

Function compute_s_max(i, j)
  send (v_i, r_i, x_i, y_i) to j
  receive (v_j, r_j, x_j, y_j)
  return s_max
```

Figure 2: ORWAR pseudo-code

3.4 Contact window

Before sending/relaying a message the algorithm computes the size of largest transmittable message (s_{max}). This will be used to relay only messages that have a lower chance of transmission failure, thereby saving transmission power and bandwidth. Both energy and bandwidth are further optimized by selection of messages that fit into s_{max} in the order of utility/bit. Figure 2 shows the pseudo-code for the algorithm. s_{max} is computed based on the current connectivity context by estimating first the contact time window (t_{cw}) and the data rate (b):

$$s_{max} = b \times t_{cw}$$

DataRate (b) is given by the device radio properties (i.e. for Bluetooth 2.0 data rates are about 250kBps). ContactWindow (t_{cw}) is calculated from nodes' respective speeds ($\vec{v}_1$, $\vec{v}_2$), nodes' coordinates and transmit range (r_1, r_2) as shown in Figure 3, in which dashed trajectories denote movement of one node. Of course, mobility implies that nodes can change speed or movement path during a given transmission. If actual contact window is different from t_{cw} then it is possible that the transmission of some selected message will fail. 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. Nodes will be in contact as long as distance between them will not exceed the minimum radio range:

$$distance(node_1, node_2) \leqslant min(r_1, r_2)$$

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 2) conserving power as radio is not wasted for messages that cannot be sent anyway.

4. EVALUATION

4.1 Simulation setup

We evaluate the performance of ORWAR in comparison with five well-known delay-tolerant network routing protocols: MaxProp [2], SprayAndWait [18], Prophet [13], Epidemic [20] and DirectDelivery. We use ONE (Opportunistic Network Environment) [14], 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

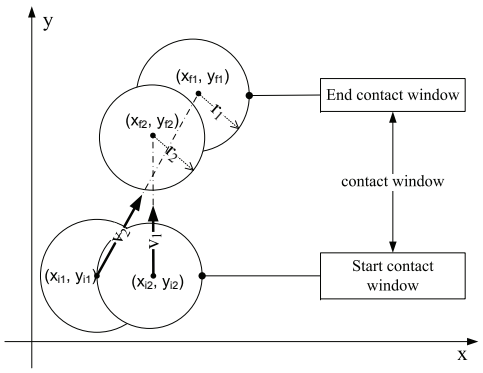

Figure 3: Estimation of the contact window

comes with 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 made sure both are run in the evaluation with the same replication factor (L=6). Since messages are evenly distributed between 3 priority classes in our experiment the total (maximum) number of copies in the system is not changed from SprayAndWait, thus giving us comparable results. ORWAR just applies 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 [13] is run with the following parameters: delivery predictability $P_{(a,b)} = 0.75$, the scaling constant $\beta = 0.25$ and aging constant $\gamma = 0.98$.

In evaluations below we considered a city setup with 126 nodes (80 pedestrians, 40 cars, 6 trams) sharing a 4500m x 3500m playground. We assumed each node has a network interface allowing a transmission range of 10m for pedestrians and 20m for cars and trams. For both we considered a transmission speed of 250kBps (2Mbps). Buffers are considered to be 5MB except for the trams which have 50 MB. The mobility pattern is very 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 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 as well as random time pause. Network is still very sparse, with the cumulated 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 bundles 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 the size classes, i.e., every size class (short messages, documents and multimedia) has equal number of bundles of utility 1, 2 and 3. In the following we refer to this size distribution as S

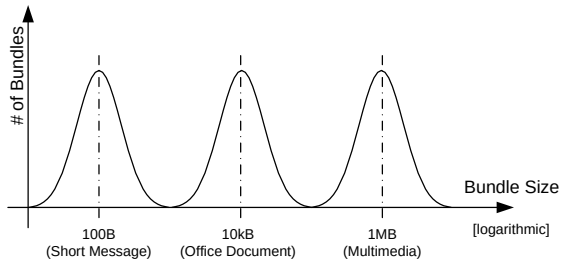

Figure 4: Size distribution in standard message set (S)

4.2 Message size implication

We run the first experiment in order to analyze the impact of the size of messages. 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.

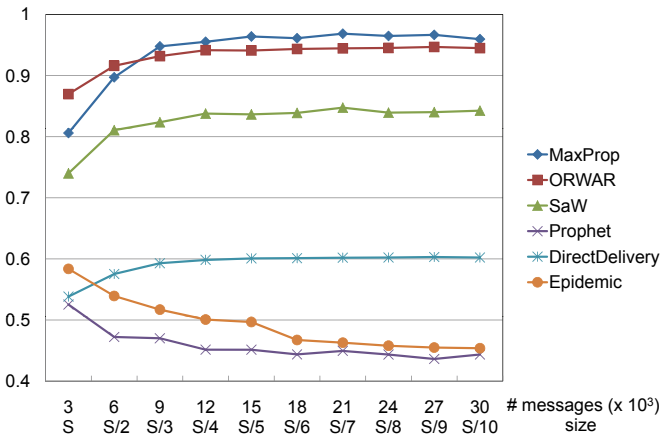

Figure 5: Delivery rate versus message size

Figure 5 shows that MaxProp provides the best overall delivery rate but only if message size is small enough. We note that ORWAR performs better when bigger

messages are injected into the system. When analyzing overheads, defined here as number of transmitted bundles divided by number of bundles 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 ORWAR diminishes partial transmissions.

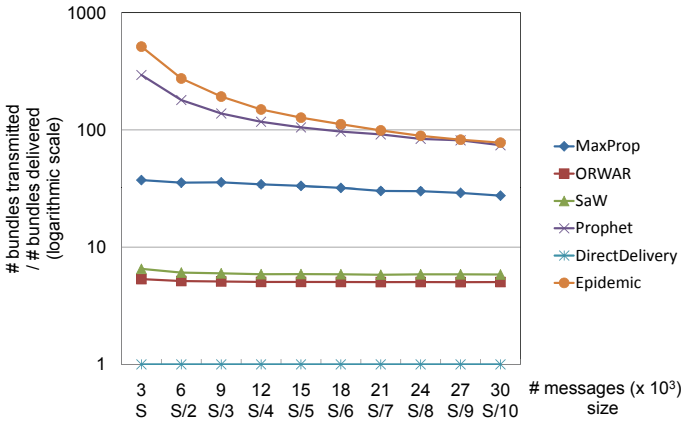

Figure 6: Delivery rate versus message size

From these 2 figures we can conclude that ORWAR has a good overall performance: very close to MaxProp in terms of delivery rate and second best overhead, after DirectDelivery. Moreover, when larger message are to be transmitted, ORWAR has better delivery rate than MaxProp. 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 rest of the paper we are going to present only the best performing protocols: MaxProp, ORWAR and SprayAndWait.

4.3 Energy implications

The main goal in design of 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 number of transmitted bundles divided by number of delivered bundles is widely used in the literature, simply counting number of bundles would potentially disadvantage protocols that use (small sized) acknowledgements, such as MaxProp over those who do not, such as SprayAndWait. Instead of focusing on 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 the "waste cost".

In the next experiment we increase network load by gradually growing 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.

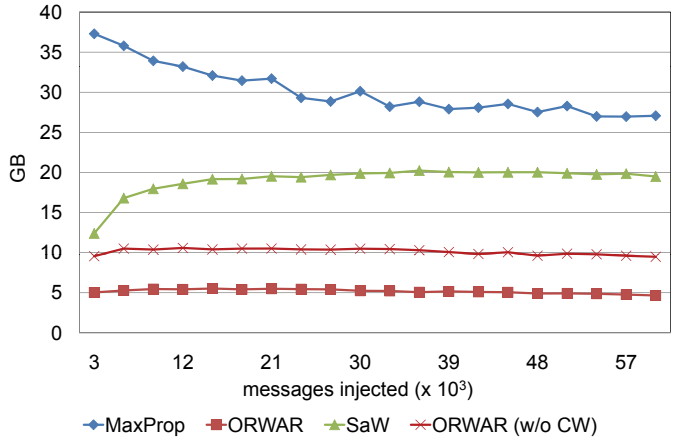

Figure 7: Partial transmission total size versus load

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 transmission over not calculating it at all. As ORWAR computes the most valuable message to be sent in a given meeting context, we expect that 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 realized during 12 hours.

It shows that ORWAR sends 10% to 50% more data than MaxProp and SprayAndWait over the same interval of time (12h). It also shows that throughput is

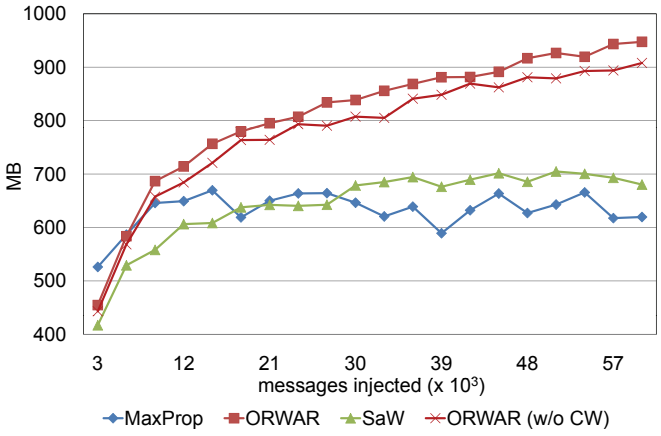

Figure 8: Throughput versus load

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.

4.4 Load implication

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.

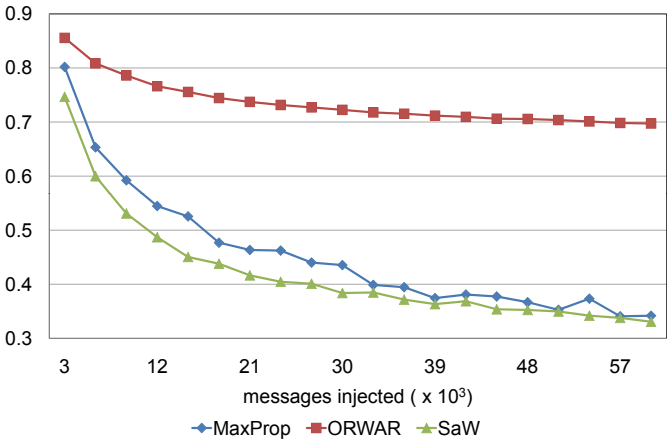

Figure 9: Delivery rate versus load

ORWAR not only has the best overall delivery rate but also its relative performance compared to other protocols increases at higher loads. The explanation is that ORWAR maintains a low overhead which pays off when network is congested. Other mechanisms, such as effective queue management, utility-based replication, usage of acknowledgements, contact window estimation are also contributing. 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.

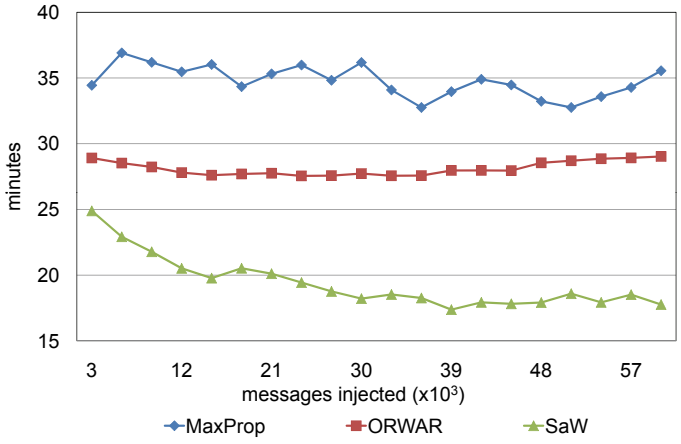

Figure 10: Median latency versus load

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

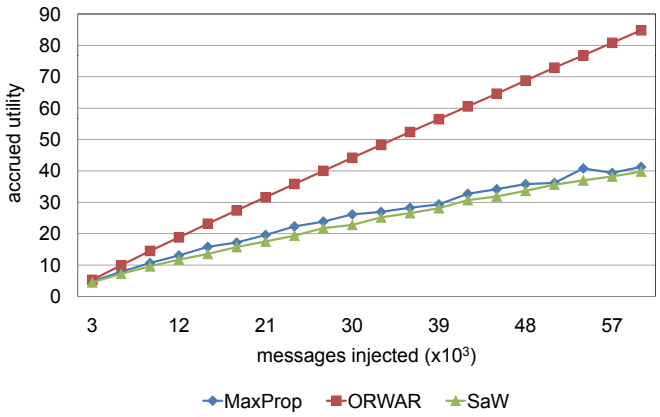

Figure 11: Delivery rate versus load

Note that utility is accounted for only if the respective bundle reaches the 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 number of messages delivered in each utility class, as follows:

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

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 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.

4.5 Mobility implications

We are interested in how the node speed affects ORWAR performance and more precisely when related to contact window estimation. We have shown in Figure 7 that by sending only messages that have a good chance to arrive within the contact window, we diminish partial transmissions without diminishing delivery rate and even increasing throughput. Those gains correspond to a medium speed in table 2. We are going to extend these measurements to other speeds.

Table 2: 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:

$$RRPT = 1 - S_{ORWAR}/S_{OwoCW}$$

where:
S_{ORWAR} = total data volume lost due to message abortions 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 20% and 58%. 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 50%).

5. CONCLUSION AND FUTURE WORK

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

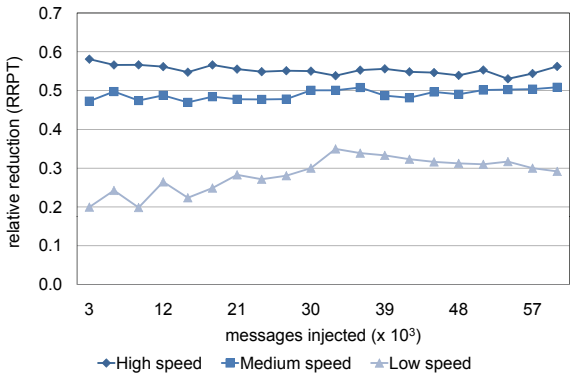

Figure 12: Partial transmissions versus load

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 acknowledgement 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. In a simulation setting we have illustrated the superior performance of the algorithm in comparison with five 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 delivery rate to MaxProp 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 paper proposing a routing scheme well-suited for large message sizes with no fragmentation, and taking account of resource optimizations 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. 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. Studying the algorithm performance within other DTN applications e.g. in a disaster scenario is also an interesting direction to pursue.

6. REFERENCES

[1] A. Balasubramanian, B. N. Levine and A. Venkataramani. *DTN Routing as a Resource Allocation Problem,* in Proc. SIGCOMM'07, ACM, volume 37, issue4, pp. 373-384, August 2007.

[2] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine. *MaxProp: Routing for Vehicle-Based Disruption- Tolerant Networks,* In Proc. IEEE Infocom, pp. 1-11, IEEE, April 2006.

[3] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. *Delay-tolerant networking: an approach to interplanetary Internet,* IEEE Communications Magazine, Volume 41, issue 6, pp. 128-136, June 2003.

[4] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. *Delay-tolerant networking architecture,* RFC4838, RFC Editor, April 2007.

[5] C. Curescu and S. Nadjm-Tehrani. *A Bidding Algorithm for Optimised Utility-based Resource Allocation in Ad hoc Networks,* to appear in Transactions on Mobile Computing, IEEE Communications Society, 2008.

[6] Delay-Tolerant Networking Research Group. *DTN reference implementation, Version 2.5,* Available at HTTP: http://www.dtnrg.org/docs/code, October 2007.

[7] V. Erramilli, A. Chaintreau, M. Crovella, and C. Diot. *Diversity of forwarding paths in pocket switched networks* in Proc. ACM IMC 07 pp. 161-174, October 2007.

[8] S. Farrell and V. Cahill. *Delay and Disruption Tolerant Networking,* Artech House, October 2006.

[9] J. Hyewon, M. Ammar, M. Corner and E. Zegura. *Hierarchical Power Management in Disruption Tolerant Networks with Traffic-Aware Optimization,* in Proc. SIGCOMM'06 Workshops, ACM, pp. 245-252, September 2006.

[10] S. Jain, R. Shah, W. Brunnette, G. Borriello, and S. Roy. *Exploiting mobility for energy efficient data collection in sensor networks,* Mobile Networks and Applications, Springer, Volume 11, number 3, pp. 327-339, June 2006.

[11] E. P.C. Jones, L. Lily, J. K. Schmidke, and P. A. S. Ward. In *Practical routing in delay-tolerant networks,* IEEE Transactions on Mobile Computing, IEEE, Vol. 6, No. 8, pp. 943-959, August 2007.

[12] P. Juang, H. Oki, Y. Wang, M. Martonosi, D. Rubenstein and L. Peh. In *Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet,* ACM SIGPLAN Notices, ACM, Volume 37, Issue 10, pp. 96-107, October 2002.

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

[14] Homepage of Opportunistic Network Environment (ONE). http://www.netlab.tkk.fi/%7Ejo/dtn/#one, Version 1, Accessed February 2008.

[15] J. Ott and D. Kutscher. *A disconnection-tolerant transport for drive-thru Internet environments,* INFOCOMM 2005, IEEE, Volume 3, pp. 1849-1862, March 2005.

[16] J. Ott and D. Kutscher. *Applying DTN to Mobile Internet Access: An Experiment with HTTP,* Technical Report TR-TZI-050701, Universitt Bremen, July 2005.

[17] N. Shank, B. Sokol, M. Hayes, and C. Vetrano. *Human Services Data Standards: Current progress and Future Visions in Crisis Response,* in Proc. ISCRAM'08, May 2008.

[18] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. *Spray and wait: an efficient routing scheme for intermittently connected mobile networks,* in Proc. ACM SIGCOMM Workshop on Delay-Tolerant Networking (WDTN'05), pp. 252-259, August 2005.

[19] T. Spyropoulos, K. Psounis, and C. S. Raghavendra. *Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility,* in Proc. PerCom Workshops apos, pp. 79-85, March 2007.

[20] TA. Vahdat and D. Becker. *Epidemic routing for partially-connected ad hoc networks,* Tech. Rep. CS-2000-06, Duke University, July 2000.

[21] W. Wang, V. Srinivasan and M. Motani. *Adaptive contact probing mechanisms for delay tolerant applications,* in Proc. MobiCom '07, ACM, pp. 230-241, September 2007.

[22] W. Wang, V. Srinivasan and M. Motani. *Delay-Tolerant Networks (DTNs), A Tutorial, V1.1.,* [Online document], Available at http://www.ipnsig.org/reports/DTN_Tutorial11.pdf, May 2003.

[23] C. Zhang. *Routing in Intermittently Connected Mobile Ad Hoc Networks and Delay tolerant networks: Overview and Challenges,* IEEE Communication Surveys, IEEE, Volume 8, Issue 1, pp. 24-37, Jan 2006.