BATMAN Store-and-Forward: the Best of the Two Worlds

Laurent Delosières and Simin Nadjm-Tehrani
Department of Computer and Information Science,
Linköping University SE-581 83 Linköping, Sweden
{*laurent.delosieres, simin.nadjm-tehrani*}*@liu.se*

***Abstract*—The need for communication is highest in disaster scenarios when the infrastructure is also adversely affected. A recent protocol for ad hoc communication, the BATMAN protocol, is dependent on minimal infrastructure, in the form of mesh nodes that are used as access points, or nodes acting as an intermediary in a multi-hop connection. While BATMAN works well in a scenario in which there is a multihop path from senders to receivers at all times, it will drop the packets in intermittently-connected networks. Moreover, although implementation on a device is essential as a proof of concept, performing large scale evaluations requires a simulation platform in which variations in the operating environment can be studied. This paper is about adding the store-and-forward mechanism to the routing component in BATMAN nodes, to overcome intermittent connectivity through mobility. We describe an extension of the protocol, SF-BATMAN, that has been implemented in an interoperable manner with BATMAN, i.e. with no added signaling, and no change of basic BATMAN settings. We have implemented SF-BATMAN in a packet level simulator (NS3), and demonstrated its performance in a scenario that consists of two regions of connectivity: a well-connected mesh network and a set of sparser subnetworks. We show that the added capability enhances the performance of BATMAN, through an increase of the delivery ratio by 20% with a lower overhead, while it exhibits a similar latency in comparable network scenarios.**

***Keywords*-mesh network, delay-tolerant routing, BATMAN, store-and-forward, NS3.**

I. Introduction

Communication in disaster area networks is vital for essential services that create situation awareness and facilitate disaster relief. Since the need for communication is often highest when the infrastructure is most overloaded or indeed damaged, there is a growing evidence that ad hoc networks have a role to play in post-disaster communication [1], [2], [3]. Such infrastructure-free networks are a complement to the existing technologies such as satellite phones, TETRA-based devices, mobile base stations etc., which are too expensive to deploy in a large area for mass communication.

A recent deployment of a mobile ad hoc network (MANET) routing protocol, the BATMAN protocol, in so called Bat-phones [4], is dependent on minimal infrastructure [5]. The idea is to use mesh nodes as access points, or use them as peers which rely on every phone acting as an intermediary in a multi-hop connection. While BATMAN works well in a scenario in which there is a multihop path from senders to receivers at all times, it will drop the packets in intermittently-connected networks. Moreover, although implementation on phones is essential as a proof of concept, performing large scale evaluations requires a simulation platform in which variations in the operating environment can be studied.

This paper is about adding the concept of store-and-forward from delay-tolerant networks (DTN) to the routing component in BATMAN nodes, in order to overcome intermittent connectivity through mobility. We describe an extension of the protocol implemented in a packet level simulator (NS3) and demonstrate its performance in a scenario that consists of two regions of connectivity: a well-connected mesh network and a set of sparser subnetworks that are intermittently connected to some elements of the mesh network. We show that the added capability enhances the performance of BATMAN both measured as delivery ratio and overheads.

The BATMAN store-and-forward version enables to route packets both in mesh and delay-tolerant subnetworks by using only one routing protocol. As opposed to other works [6], [7], [8], [9] in which one routing protocol was used for delay-tolerant networking and another one for mesh networking, our approach provides interoperability with an existing performant mesh network routing protocol.

The contributions of this paper are (1) implementation of the BATMAN protocol in Network Simulator 3 (NS3), (2) adding the store-and-forward capability in an interoperable manner with no extra signaling and (3) evaluation of the extended protocol in a hybrid scenario, a mix of mesh nodes and mobile DTN nodes, showing an added value.

The paper is structured as follows: Section II introduces the concepts of mesh and delay-tolerant networks, and explains how the routing protocol BATMAN works. Section III explains the modifications brought to BATMAN in order to incorporate store-and-forward. Finally, Section IV evaluates BATMAN and its store-and-forward version in a mixed scenario composed of DTN and mesh nodes.

II. Background

A MANET is a self-configuring and infrastructureless network in which mobile devices may be used as relays to facilitate the communication between unreachable pairs of devices. The assumption on such a network is the existence of a multi-hop path between each pair of devices at all times.

978-1-61284-936-2/12/$26.00 ©2011 IEEE

A mesh network is very similar to a MANET except that the nodes composing the infrastructure tend to be stationary or have limited mobility, and some can be used as dedicated routers. This network is quite reliable since there are many paths to connect two devices.

A DTN is composed of devices which may not be able to communicate with each other because of temporary disruptions. To cope with these disruptions, a mechanism called "store-and-forward" is central to DTN protocols. This mechanism enables to store the messages when no paths are available towards the destination and forward it later to the destination or to custodians that will transmit the message towards the destination.

BATMAN [5] is a protocol originally developed for mesh network routing based on the ant pheromones. The pheromones created by every node or originator in the network are called Originator Messages (OGMs). Each node periodically broadcasts an OGM to its neighbours, which in turn rebroadcasts the OGM to its neighbours. While some OGM messages will be lost due to contention or mobility, most nodes will get an update of the information about the nodes in the network on a regular basis.

Upon receiving a new OGM, each node stores it in a buffer to keep track of the previous OGMs received. Each node also updates the last aware time of the originator which corresponds to the latest time of the reception of the OGM created or forwarded by this originator. The node having the highest quantity of pheromones from a given originator is likely to be the first hop on the best path to reach this originator. Each node keeps a timer for each originator in its list of originators in order to remove the obsolete originators from its list.

In order to avoid one-way communication, there is added information in the OGM to check whether or not the link between two nodes is bidirectional. This process is done in three steps. First, a node broadcasts its new OGM. Secondly, the node in its proximity upon receiving this OGM broadcasts it in turn. Thirdly, the first node will receive its own OGM and will know that the communication is bidirectional between it and the node that has sent back the OGM. In other words, these nodes are neighbors. A node *A* considers the link as bidirectional with a node *B* as long as the number of new OGMs that are self-originated by the node *A* since the last establishment of the link bidirectionality with node *B* is below a certain threshold. This threshold is called the bidirectional link timeout.

When a node wants to send a message to a given destination, it will forward the message to its *best* neighbor. The best neighbor is the one that has forwarded the highest quantity of OGMs issued by the destination within a sliding window as described below.

Since each neighbor may forward different OGMs from a given originator, a unique sliding window of constant size is used to keep track of the freshest OGMs rebroadcasted by each neighbor. This window is updated each time a new OGM is received from a neighbor. The notion of best neighbor towards a given destination at a given time can thus replace the classic routing table in proactive MANET routing. In the descriptions below, we will use the primitives *GetNextHop* and *UpdateRoutingTable* for the routing operations that are performed on the OGM lists.

Reineri et al. [10] have used a smart sliding window which counts the OGMs according to their weights. These weights are computed by an exponential function based on the rank of OGMs contained in the sliding window. In other words, the latest OGMs received are more prioritized than old ones. This version of BATMAN is referred to as SW-BATMAN. From now on, when we refer to BATMAN, we intend the optimized version (SW-BATMAN).

III. BATMAN AND ITS EXTENSION

In this section, we describe the processing of packets with BATMAN that has been modified and extended to create a delay-tolerant version of BATMAN (SF-BATMAN). We begin by describing the skeleton of the original protocol, and then go on to extend it in order to include the partition tolerance mechanism.

A. BATMAN

Procedure 1 describes how hop-by-hop forwarding works. When an OGM packet arrives from a bidirectional link, or an originator in the list of originators becomes obsolete, the routing table will be updated.

When an application packet needs to be sent, we first get the chosen neighbor from the routing table via the function $GetNextHop$ and then we send the packet via the function $SendPacket$ if there exists a chosen neighbor. Otherwise we drop the packet.

When a forwarded packet arrives, we first check if it is intended for us via the function $GetPacketDestination$. If so, we deliver the packet via the function $DeliverPacket$ otherwise we find the chosen neighbor, if it exists, and send the packet to it. If it does not exist, we simply drop the packet.

B. Store-and-Forward BATMAN

Procedure 2 describes how hop-by-hop forwarding works with SF-BATMAN. SF-BATMAN is similar to BATMAN except that in the point where BATMAN would send a packet SF-BATMAN tries to send it via the function $TrySending$ (Procedure 3) which saves the packet in case the chosen neighbor does not seem to be reachable. More specifically, the operations *GetNextHop*, *UpdateRoutingTable*, *SendPacket*, *DeliverPacket*, and *GetPacketDestination* are identical to the corresponding BATMAN operations. This means in particular, that a packet can be forwarded towards a destination for which the path information is

Procedure 1 BATMAN running at node i

When an application packet arrives
  $chosen\_neighbor$ = GetNextHop($packet$)
  if $chosen\_neighbor \neq NULL$ **then**
    SendPacket ($packet$, $chosen\_neighbor$)
  else
    DropPacket($packet$)
  end if
EndWhen
When a forwarded packet arrives
  if GetPacketDestination($packet$) $\neq$ i **then**
    $chosen\_neighbor$ = GetNextHop($packet$)
    if $chosen\_neighbor \neq NULL$ **then**
      SendPacket($packet$, $chosen\_neighbor$)
    else
      DropPacket($packet$)
    end if
  else
    DeliverPacket($packet$)
  end if
EndWhen
When a new OGM packet arrives from a bidirectional link
  UpdateRoutingTable()
EndWhen
When an originator becomes obsolete
  UpdateRoutingTable()
EndWhen

somewhat stale. But this is also a property of BATMAN and our goal is not to redefine the original protocol.

Another difference is the attempt to send the packets stored in the buffer SF_{buffer} via the function $IteratePackets$ (Procedure 4) when a packet to forward or an OGM packet arrives.

Procedure 3 describes how we try to send the message. First we get the last aware time of the next hop via the function *GetMostRecentAwareTime* as well as the current time. If the difference between the current time and the last aware time of the chosen neighbor is less than T_C (explained below) then the chosen neighbor is likely to be in our transmission range, thus we forward the message to it. If the packet is not likely to succeed in forwarding and it is new, we store it in the SF_{buffer}.

DTN protocols typically use replication to reduce the risk of losing a packet at the MAC Layer [11], [12]. Since BATMAN has a single forwarding policy, we need to be careful about partial transmissions resulting in lost packets at the MAC layer [1]. To cope with this, we only forward to neighbors that we believe are currently in our range. The nodes that are currently in contact are most likely those that have sent OGMs in the latest OGM period. If movements are slow the duration of the contact window could be larger than the OGM period. In our work, the approximation of the contact window is a configurable parameter T_C. We have chosen not to estimate the contact window as in earlier works [12] to preserve interoperability with BATMAN (i.e. avoiding a change of the OGM structure by adding speed, position, etc). In addition, using this method we keep the standard BATMAN timers for removing originators from the list which is another interoperability argument. Even though the SF-BATMAN protocol tries to limit the loss of partial transmissions, some may still occur due to interferences or node mobility subsequent to transmission. While retransmissions at the transport layer or application layer are not ruled out, our goal here is not making any assumptions on their existence, and try to reduce the partial transmissions at IP layer with minimum overhead.

[1]Note that changing the MAC layer would make the protocol non-interoperable.

Procedure 2 SF-BATMAN running at node i

When an application packet arrives
  $chosen\_neighbor$ = GetNextHop($packet$)
  TrySending($packet$, $chosen\_neighbor$)
EndWhen
When a forwarded packet arrives
  if GetPacketDestination($packet$) $\neq$ i **then**
    $chosen\_neighbor$ = GetNextHop($packet$)
    TrySending($packet$, $chosen\_neighbor$)
  else
    DeliverPacket($packet$)
  end if
  IteratePackets()
EndWhen
When a new OGM packet arrives from a bidirectional link
  UpdateRoutingTable()
  IteratePackets()
EndWhen
When an originator becomes obsolete
  UpdateRoutingTable()
  IteratePackets()
EndWhen

Procedure 3 TrySending

Input: $packet$, $chosen\_neighbor$
  T = GetMostRecentAwareTime($chosen\_neighbor$)
  T_{now} = GetTimeNow()
  if $T_{now} - T \leq T_C$ **then**
    SendPacket($packet$, $chosen\_neighbor$)
  else
    $SF_{buffer} = SF_{buffer} \cup packet$
  end if

Procedure 4 IteratePackets

```
for each packet ∈ SF_buffer do
    chosen_neighbor = GetNextHop(packet)
    if chosen_neighbor not NULL then
        TrySending (packet, chosen_neighbor)
    end if
end for
```

IV. Simulation

In this section, we show that the interoperable operation of BATMAN and SF-BATMAN in a mixed scenario has an added value. By a mixed scenario, we mean one in which there is a mesh network deployment in some part of the area and there are other nodes moving in a surrounding DTN. The hypothesis is that nodes that are equipped with SF-BATMAN will make the best of the mesh infrastructure in a BATMAN like fashion, and when they are in a DTN they exploit their SF-BATMAN capabilities.

A. Scenario

The scenario is composed of 60 nodes spread all over Helsinki. Amongst them 30 are stationary and grouped together to form a mesh network located in the center and 30 others composed of pedestrians, cars and trams representing DTN nodes spread elsewhere in the area (see Figure 1). The mobility pattern has been generated by the Helsinki scenario of simulator ONE [13] which is well-known as a platform for DTN evaluations, and publicly available. These mobility traces were adapted to the NS3 input format.

The pedestrians dispose of a short transmission range reflecting mobile phones compared to the other type of nodes such as base stations which have a much larger transmission range. In order to reflect the interferences between the mesh nodes, we have chosen a transmission range larger than the distance between two neighbors. Table I describes the characteristics of the nodes.

The UDP traffic is split equally between the mesh subnetwork and DTN subnetwork where half of the traffic was issued and intended for the moving nodes, and the other half by the mesh nodes and for the mesh nodes. Sender and recipient are randomly selected in their respective subnetwork.

Four experiments have been done lasting 3 hours and 20 minutes each, with the number of messages sent ranging from 3.000 messages to 60.000 messages of 1.500 bytes each, in each subnetwork. In each experiment, the respective load is 0,67 messages per second, 2,68 messages per second, 4,69 messages per second, and 6,70 messages per second over all the network. Each point in the following figures represents the average of twelve simulations along with their 95% confidence interval.

Concerning the BATMAN protocol, the sending period of OGMs has been fixed to one second. This duration was also used for the contact window T_C. The other parameters have been fixed according to the BATMAN rfc [14]. The first hour of simulation has been used for building up the routing table and getting the network started up so that BATMAN is in a stable state to be sure to be fair when isolating the effects of the SF. As for the counting process of OGMs, it has been done according to the earlier work by Reineri et al. [10]. For the sake of a complete picture we restate these parameters in Table II. Here, we also include the parameters that are specific to SF-BATMAN, i.e. T_C.

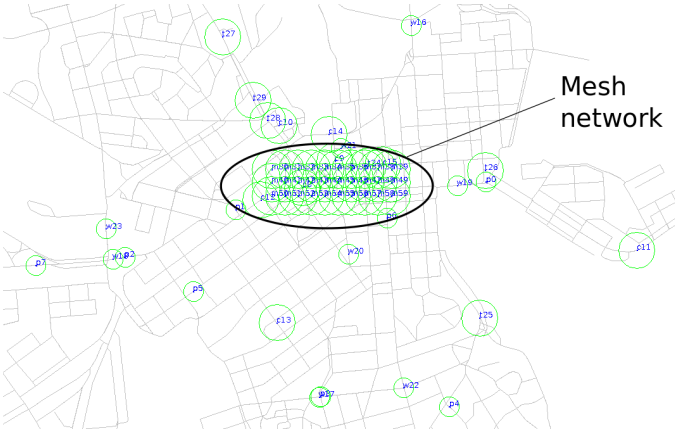

Figure 1. Helsinki - mesh network and DTN nodes

B. Results

Concerning the delivery ratio for BATMAN and SF-BATMAN, we expect SF-BATMAN to have a higher delivery ratio than BATMAN since SF-BATMAN stores the packets that are dropped due to partitions in BATMAN. This is confirmed by our results shown in Figure 2. We can also notice that when the number of messages injected into the network increases, the number of messages received decreases. However, the difference of delivery ratio is constant as the number of messages sent increases which may be explained by the fact that the buffer SF_{buffer} is large enough and therefore packets are not dropped due to memory limitation (in the worst case, 1,54 MB of buffer space was used). Moreover, since 50% of the UDP traffic is inside the mesh network, we would intuitively expect a delivery ratio of 50% (or higher) for the two protocols. The lower delivery ratio is explained by higher interferences in the mesh network since the DTN traffic is also passing through the mesh for certain destinations.

An analysis of the overhead between SF-BATMAN and BATMAN follows next. We define the overhead as the total number of packets (i.e. OGM packets as well as data packets) sent in the network over the total number of messages received. We expect to have a lower overhead for SF-BATMAN compared to BATMAN. This is based on the intuition that SF-BATMAN has a higher delivery ratio than BATMAN and the same number of OGM packets

Table I
CHARACTERISTICS OF NODES

Node type	Number	Transmission range (meters)	Node speed (km/h)	Pause (seconds)	Buffer Size (MB)	Maximum bitrate (Mbps)
Pedestrian	16	51	1.8-5.4	5-120	5	6
Car	8	95	10-50	5-120	50	6
Tram	6	95	10-50	10-30	50	6
Static node	30	102	0	0	50	6

Table II
SIMULATION PARAMETERS

Time To Live (in number of hops)	128
Sliding window size	128
Timer (to remove obsolete originators)	1280 seconds
OGM sending interval	1 second
Birectional link timeout (consecutive OGMs)	10
T_C	1 second

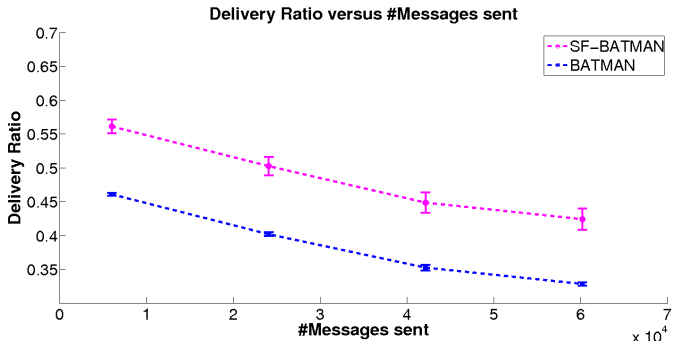

Figure 2. Delivery ratio

which represent the overwhelming majority of packets in the network [2]. Moreover, since the number of messages received increases as the number of messages injected in the network increases while the total number of packets remains almost constant, we expect to have a decrease in overhead. This is confirmed by our results in Figure 3.

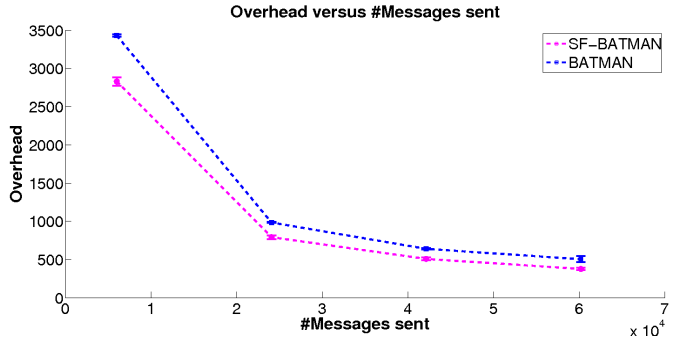

Figure 3. Overhead

[2]Note that this also implies that computing overhead only based on the OGMs would not change the picture or add new insights.

We have also analyzed the average latency defined as the average time taken for the packets to reach their destination excluding the packets that were lost on the path for both protocols. It turned out that BATMAN has a very low latency of 0.2 seconds in contrast with the SF-BATMAN which shows a higher average latency (Figure 4). This difference is created by the mobility, and sparseness induced by network partitions, and the time taken for buffered packets to move towards destination. In other words, the latency is influenced by all the waiting times for each packet at each hop. Considering that many more messages are delivered by SF-BATMAN, a higher average latency is expected. We also note that the average latency for messages exchanged only in the mesh network for SF-BATMAN (i.e. 0.4 seconds) is not that far from BATMAN (i.e. 0.2 seconds).

As a first conlusion, we can say that SF-BATMAN outperforms BATMAN while having a lower overhead and an average latency below 1 second for routing packets in the mesh network. Similar results were observed when the messages were addressed for random destinations and not only for the nodes belonging to the same subnetwork.

V. RELATED WORKS

One of the first communication protocols for disaster area networks with a store-and-forward capability was suggested by Asplund et al. [1]. That protocol was intended for energy-efficient manycast in a network with no knowledge of present node addresses. This paper is about directed communication via unicast.

Johnsson et al. [5] have compared the performance of BATMAN created by Neumann et al. [14] with the Optimized Link State Routing (OLSR) [15]. They showed that in a static grid composed of forty nine routers, BATMAN outperforms OLSR on almost all performance metrics (i.e. routing overhead, best overall throughput, delay, etc).

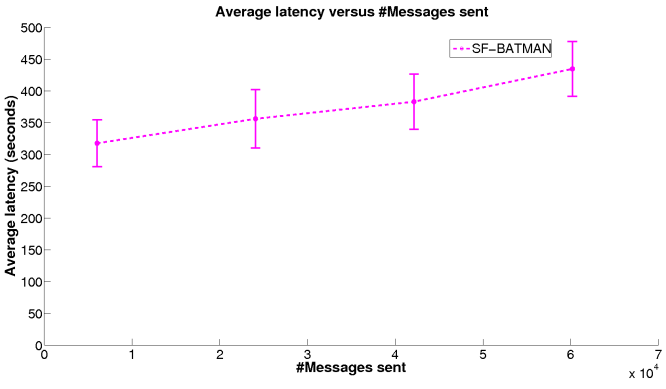

Figure 4. Average latency

Reineri et al. [16] have compared the performance of their modified BATMAN (i.e. SW-BATMAN) with OLSR, OLSR-ETX and BATMAN for routing MPEG video packets with and without background traffic. Their SW-BATMAN version outperforms the other routing protocols in terms of rate loss, jitter, etc. These works motivate our interest for the basic BATMAN protocol.

To route a packet in both a connected and disconnected network, Musolesi et al. [6] have used the routing protocol CAR which is the combination of the Destination-Sequenced Distance-Vector (DSDV) routing protocol and their own DTN routing protocol based on the delivery probability. DSDV was used for the intra-partition communications and a DTN routing protocol for the inter-partition communications. To limit the number of messages in the network while ensuring a high delivery probability for each packet, they were replicating the messages according to their delivery probability. As opposed to their work, we only use one type of signaling packet (i.e. OGM).

Liu et al. [9] have proposed a new routing protocol called Efficient Adaptive Routing (EAR) which is composed of DSDV and Spray and Wait [11]. They form logical clouds which are defined as the set of nodes being able to communicate with each other. However, they limit the number of nodes per logical cloud in order to limit the bandwidth used by each node for maintaining the DSDV shortest paths. Indeed, nodes belonging to the same logical cloud only need to maintain paths inside this cloud. Between the clouds, nodes use the Spray and Wait routing protocol. Whitbeck et al. [7] have created their own hybrid routing protocol Hybrid DTN-MANET (HYMAD). They split the network in small clusters where nodes use a distance vector routing protocol for intra-partition routing and Spray and Wait for inter-partition routing. In our case, we do not impose any logical clouds or small clusters.

Ott et al. [17] propose a hybrid routing protocol composed of AODV and a DTN routing protocol where AODV discovers nearby DTN routers and uses DTN router information for getting route hints. The nodes choose between routing packets with AODV and end-to-end TCP if there is an existing path to the destination and routing packets with hop-by-hop DTN and bundles otherwise. In contrast with them, we do not need to modify the signaling packets.

Lakkakorpi et al. [8] have used AODV for the MANET routing and the epidemic routing for the DTN routing. They switch from MANET to DTN routing based on the message size, the node density and the path length to destination. The epidemic routing protocol replicates unlimitedly the messages until they arrive at the destination which causes memory depletion of the nodes.

VI. CONCLUSIONS AND FUTURE WORK

We have embedded the store-and-forward functionality in BATMAN and compared it with its basic version in a scenario with a mixture of mesh nodes and DTN nodes. It meant adding a buffer to store the packets which did not have any routes towards the destination or packets whose next hop seemed unreachable. To limit the number of packets lost because of node mobility, we have used a time window parameter to approximate a contact window in which the nodes were allowed to send messages to their next hop or to the destination.

We have illustrated the benefits of the extended protocol by showing that the delivery ratio of SF-BATMAN is 20% higher than BATMAN with a lower overhead. Even though the average latency for the whole traffic is larger when running SF-BATMAN than the one experienced in BATMAN, the packets which are routed inside the mesh network have a similar average latency for the two protocols.

Future works include implementation of the protocol on a device, its comparison with other hybrid protocols in more detail, and the exploration of the replication mechanism for higher delivery ratio.

ACKNOWLEDGEMENTS

This work was supported by the EU project SecFutur and the use case with embedded devices for disaster communication.

REFERENCES

[1] M. Asplund and S. Nadjm-Tehrani, “A partition-tolerant manycast algorithm for disaster area networks.” in *SRDS*. IEEE, 2009, pp. 156–165. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/SRDS.2009.16

[2] “The serval project making communications available anywhere, anytime.” Dec. 2011. [Online]. Available: http://www.servalproject.org/

[3] “Lifenet: Organization.” Dec. 2011. [Online]. Available: http://thelifenetwork.org/org.html/

[4] “Open-mesh.” Dec. 2011. [Online]. Available: http://www.open-mesh.org/

[5] D. Johnsson, N. Ntlatlapa, and C. Aichele, "A simple pragmatic approach to mesh routing using BATMAN," in *2nd IFIP International Symposium on Wireless Communications and Information Technology in Developing Countries*. Pretoria, South Africa: CSIR, 2008. [Online]. Available: http://wirelessafrica.meraka.org.za

[6] M. Musolesi and C. Mascolo, "CAR: Context-aware adaptive routing for delay-tolerant mobile networks," *IEEE Transactions on Mobile Computing*, vol. 8, pp. 246–260, 2009. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TMC.2008.107

[7] J. Whitbeck and V. Conan, "HYMAD: Hybrid DTN-MANET routing for dense and highly dynamic wireless networks," *Computer Communications (Butterworth-Heinemann)*, vol. 33, pp. 1483–1492, August 2010. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2010.03.005

[8] J. Lakkakorpi, M. Pitkänen, and J. Ott, "Adaptive routing in mobile opportunistic networks," in *Proceedings of the 13th ACM international conference on Modeling, analysis, and simulation of wireless and mobile systems*. New York, NY, USA: ACM, 2010, pp. 101–109. [Online]. Available: http://doi.acm.org/10.1145/1868521.1868539

[9] C. Liu and J. Wu, "Efficient adaptive routing in delay tolerant networks," in *IEEE International Conference on Communications*. IEEE, 2009, pp. 1–5. [Online]. Available: http://dx.doi.org/10.1109/ICC.2009.5198967

[10] M. Reineri, C. Casetti, and C.-F. Chiasserini, "Routing protocols for mesh networks with mobility support," in *6th International Symposium on Wireless Communication Systems (ISWCS)*. IEEE, sept. 2009, pp. 71 –75. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ISWCS.2009.5285344

[11] 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 (WDTN)*. New York, NY, USA: ACM, 2005, pp. 252–259. [Online]. Available: http://doi.acm.org/10.1145/1080139.1080143

[12] G. Sandulescu and S. Nadjm-Tehrani, "Adding redundancy to replication in window-aware delay-tolerant routing," *JCM*, vol. 5, no. 2, pp. 117–129, 2010. [Online]. Available: http://dx.doi.org/10.4304/jcm.5.2.117-129

[13] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE simulator for DTN protocol evaluation," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Simutools)*. Brussels, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 55:1–55:10. [Online]. Available: http://dx.doi.org/10.4108/ICST.SIMUTOOLS2009.5674

[14] A. Neumann, C. Aichele, M. Lindner and S. Wunderlich, "Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.)," RFC 1, Internet Engineering Task Force, Oct. 2008. [Online]. Available: http://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00

[15] T. Clausen, G. Hansen, L. Christensen, and G. Behrmann, "The optimized link state routing protocol, evaluation through experiments and simulation," in *Proc. of IEEE Symposium on Wireless Personal Mobile Communications*, September 2001. [Online]. Available: http://hipercom.inria.fr/olsr/wpmc01.ps

[16] M. Reineri, R. Rubino, C. Casetti, and C. Chiasserini, "Experimental performance assessment of WMN routing protocols with mobile nodes," in *Wireless Communications and Mobile Computing Conference (IWCMC)*. IEEE, 2011, pp. 1010 –1015. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/IWCMC.2011.5982679

[17] J. Ott, D. Kutscher, and C. Dwertmann, "Integrating DTN and MANET routing," in *Proceedings of the 2006 SIGCOMM workshop on Challenged networks (CHANTS)*. New York, NY, USA: ACM, 2006, pp. 221–228. [Online]. Available: http://doi.acm.org/10.1145/1162654.1162659