Coping with Episodic Connectivity in Heterogeneous Networks

Rao Naveed Bin Rais
INRIA
Sophia Antipolis, France
nbrais@sophia.inria.fr

Thierry Turletti
INRIA
Sophia Antipolis, France
turletti@sophia.inria.fr

Katia Obraczka
UCSC
Santa Cruz, CA, USA
katia@cse.ucsc.edu

ABSTRACT

In this paper, we present an efficient message delivery mechanism that enables distribution/dissemination of messages in an internet connecting heterogeneous networks and prone to disruptions in connectivity. We call our protocol MeDeHa (pronounced "medea") for Message Delivery in Heterogeneous, Disruption-prone Networks. MeDeHa is complementary to the IRTF's *Bundle Architecture*: while the *Bundle Architecture* provides storage above the transport layer in order to enable interoperability among networks that support different types of transport layers, MeDeHa stores data at the link layer addressing heterogeneity at lower layers (e.g., when intermediate nodes do not support higher-layer protocols). MeDeHa also takes advantage of network heterogeneity (e.g., nodes supporting more than one network) to improve message delivery. For example, in the case of IEEE 802.11 networks, participating nodes may use both infrastructure- and ad hoc modes to deliver data to otherwise unavailable destinations. Another important feature of MeDeHa is that there is no need to deploy special-purpose nodes such as message ferries, data mules, or throwboxes in order to relay data to intended destinations, or to connect to the backbone network wherever infrastructure is available. The network is able to store data destined to temporarily unavailable nodes for some time depending upon existing storage as well as quality-of-service issues such as delivery delay bounds imposed by the application. We evaluate MeDeHa via simulations using indoor scenarios (e.g. convention centers, exposition halls, museums etc.) and show significant improvement in delivery ratio in the face of episodic connectivity. We also showcase MeDeHa's support for different levels of quality-of-service through traffic differentiation and message prioritization.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Network communications, C.2.2 [**Network Protocols**]: Routing protocols

General Terms

Algorithms, Design

Keywords

Disruption tolerance, Episodic connectivity, Heterogeneous networks, Node relaying, Store-carry-and-forward

1. INTRODUCTION

It is envisioned that the Internet of the future will be highly heterogeneous not only due to the wide variety of end devices (in terms of their capabilities, e.g., storage, processing time, battery lifetime, mobility, and traffic characteristics) it interconnects, but also in terms of the underlying networks it comprises. As illustrated in Figure 1, such networks range from wired- and wireless backbones (e.g. community wireless mesh networks) to wireless infrastructure-based and ad-hoc networks (MANETs). Furthermore, current and emerging applications, such as emergency response, environmental monitoring, smart environments (e.g., smart offices, homes, museums, etc.), and vehicular networks, among others imply frequent and arbitrarily long-lived disruptions in connectivity. The resulting disruption- or delay-tolerant networks (DTNs) will likely become an integral component of future internetworks.

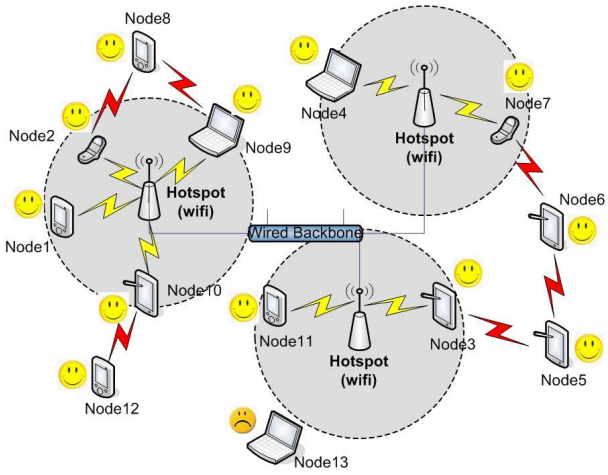

Figure 1 An example of a heterogeneous internetwork with a wired backbone, wireless infrastructure-based, and ad-hoc networks prone to episodic connectivity

As will become clear in Section 6, which describes related work, to-date, there are no comprehensive solutions targeting message delivery in heterogeneous networked environments prone to connectivity disruptions. Existing proposals either: (1) extend MANETs to handle episodic connectivity [1,2,3,4], (2) augment the coverage of access points in infrastructure-based wireless

networks by, for example, making use of multi-channel radios or switching from infrastructure mode in 802.11 [5,6,7,8], (3) provide MANETs with Internet connectivity by using special-purpose gateway nodes and a mechanism to discover them as part of route discovery in on-demand MANET routing [9], or (4) handle heterogeneity only at higher layers of the protocol stack (e.g., *Bundle Architecture* [10,11]).

In this paper we propose MeDeHa (Message Delivery in Heterogeneous, Disruption-prone Networks, pronounced "medea") – a general, yet efficient framework for data delivery in heterogeneous internets prone to disruptions in connectivity. To cope with arbitrarily long-lived connectivity disruptions, we use available storage within the network to save messages for destinations that are currently unreachable; once these destinations re-connect, messages destined to them are delivered. With respect to using in-network storage, MeDeHa is complementary to the *Bundle Architecture* proposed by the IRTF's Delay-Tolerant Networking Research Group (DTNRG) [10,11]. While the *Bundle Architecture* provides storage above the transport layer (in order to enable interoperability among networks that support different types of transport layers), MeDeHa, stores data at the link layer addressing heterogeneity at a lower layer (e.g., when intermediate nodes do not support higher-layer protocols). MeDeHa is also able to provide different levels of quality-of-service through traffic differentiation and message prioritization by controlling when messages are forwarded and for how long they are stored.

Besides, unlike existing proposals such as message ferries [12], data mules [13], or throwboxes [14], MeDeHa does not require any special-purpose nodes. Note that there is a difference between introducing special-purpose nodes in the network to perform the task of relaying (like message ferries [12], data mules [13], and throwboxes [14]) and the nodes with special capabilities that are integral part of underlying network (like APs in case of IEEE 802.11). Of course, whenever available, MeDeHa utilizes nodes with more resources and capabilities like APs, but they are part of the underlying network. Furthermore, we take advantage of the underlying heterogeneity (e.g., in the context of IEEE 802.11 networks, the nodes' ability to operate in infrastructure or ad-hoc modes) to enable message delivery across different networks.

Our current implementation of MeDeHa performs message delivery in an internet comprised of an infrastructure-based wireless network where mobile nodes roam freely between access point regions of connectivity, and become temporarily disconnected from the network. Simulation results obtained with a variety of mobility, traffic and connectivity conditions show that MeDeHa is able to improve message delivery ratio significantly. We performed simulations to analyze the behavior of MeDeHa in terms of delivery ratio as a function of data rates, buffer sizes and disconnection times and observed class-wise behavior of traffic according to some quality-of-service.

The remainder of this paper is organized as follows: Section 2 provides a detailed description of MeDeHa while MeDeHa's current implementation is presented in Section 3. The experimental methodology is briefly described in Section 4. In Section 5, we present simulation results reporting the performance of MeDeHa. Related work is reviewed in Section 6 and finally, concluding remarks and some future directions are given in Section 7.

2. MeDeHa OVERVIEW

MeDeHa's main functional components are:

***Message relaying*:** Unlike several DTN solutions, which employ specialized nodes to aid with message delivery [12,13,14], in MeDeHa any node in the network can relay messages under the *store-carry-and-forward* paradigm [11]. We thus avoid using any explicit discovery mechanism for finding specialized nodes (e.g., gateway to the backbone). Nodes may also take advantage of network heterogeneity to improve message delivery. For example, 802.11-capable nodes may periodically switch between infrastructure- and ad-hoc modes to get messages delivered across both networks.

***Buffering*:** In an environment with intermittent connectivity, it is necessary to use network nodes to store messages if a route to the intended destination(s) is not available. An important question is where to buffer these messages. In MeDeHa any node can relay messages and therefore needs to store messages whose destination(s) is(are) not available. However, we again try to take advantage of network heterogeneity. For example, Access Points (APs) in infrastructure-based wireless networks or mesh routers in the case of wireless meshes are perfect candidates to serve as temporary storage for undelivered messages as they usually exhibit higher resource availability. It is true that most current off-the-shelf APs do not typically come equipped with mass storage. We argue that adding this capability to next-generation APs is viable and will not considerably impact cost, especially if there is market demand. Furthermore, co-locating a general-purpose computing device with APs is another alternate given current AP technology.

Note that in the Bundle Architecture [10], buffering is performed above the transport layer; which in itself restricts the types of nodes that can perform this functionality. For instance, it rules out APs as buffering nodes, as APs usually run only the two lower protocol layers. In MeDeHa, buffering is done at layer 2 which enables almost any network-enabled device to relay and buffer messages. Moreover, in MeDeHa, quality-of-service is supported by enforcing application-specific requirements at the message forwarding and storage level. For instance, data belonging to real-time flows would be discarded after a pre-defined time interval specified by the application.

***Topology and content information exchange*:** Nodes periodically exchange information that is used in building their routing tables. This information includes a node's knowledge about the topology (e.g., its own neighborhood as well as what it knows about other nodes). Nodes also exchange a summary of their message buffer, their current state in terms of resources (e.g., how much storage left, remaining battery lifetime, etc.). This information is used by relay selection [15,16]. This signaling contributes to the overhead of MeDeHa. Therefore, there is a tradeoff between the overhead incurred by the protocol, how fresh paths are, and how well relay selection performs. Note that if neighborhood information is already made available by the underlying layer-2 protocol (e.g., beaconing, AP association/disassociation), MeDeHa simply makes use of it.

***Traffic differentiation*:** In order to satisfy application-specific requirements, MeDeHa uses message tags to carry information such as message priority, time-to-live (or TTL, which is the maximum amount of time the message should remain in the network), scope (e.g., maximum number of hops the message

should travel), etc. Besides performing traffic differentiation and supporting quality-of-service, message tags are also used for buffer management purposes. For instance, a message that has been stored pass its TTL would be discarded.

2.1 Overall Operation

Figure 2 illustrates MeDeHa's overall operation. By default, a node starts in *idle* state. It switches to *receive* state upon reception of a message, or to *forward* state if it has some message to send. This message can either be generated by this node, or can be the message that the node has stored for some unavailable destination. Thus, in *forward* state, if the destination is not found, the node stores the message and goes back to *idle*. Later if the destination is found, the node goes to *forward* state, delivers the message and changes its state to *idle*.

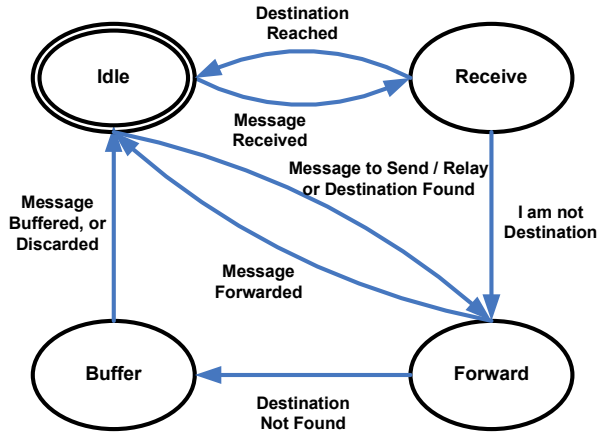

Figure 2 MeDeHa's Overall Operation

***Forward*:** When a node has a message to send either as the message originator or relay, it checks if it has a path to the destination, and if so, it sends the message along that path and switches to *idle* state. Otherwise, it tries to find a "suitable" relay. If it does not succeed, it switches to *buffer* state to store the message locally.

A number of heuristics can be used to select a relay for a *message-destination* tuple including: (1) when the node last encountered the destination (or age of last encounter), (2) how frequent the destination was encountered, (3) how mobile a node is, whether a node's mobility is "local" or "global", (4) how "social" a node is, etc. In MeDeHa, when selecting relays, we also account for the underlying heterogeneity among participating nodes, e.g., the amount of available resources such as storage, processing, and battery lifetime. In its current implementation, MeDeHa favors APs as message relays.

***Receive*:** When a node receives a message and it is not the intended destination, it switches to *forward* state and follows the steps described above.

***Buffer*:** Storing messages at relay nodes is based upon traffic differentiation and QoS requirements (e.g., message TTL, message priorities). So, when a relay node has a message to store and it doesn't have space available, it drops the oldest, lowest priority message. If all messages have the same priority, it then drops the oldest stored message. Otherwise, if the incoming packet has a priority, which is lower than all stored messages, the incoming message is discarded.

2.2 The Protocol

2.2.1 Receive

MeDeHa's receive functionality is shown in Figure 3. When a node receives a message, it switches to *receive* state and checks if it is the intended destination for the message. If so, it consumes the message (`ConsumeMessage()`) and switches back to *idle*. Otherwise, it switches to *forward* state.

```
state := Idle
if (Message is received) then
    state := receive
    if (Node is destination) then
        ConsumeMessage()
        state := Idle
    else
        Forward()
        state := forward
    endif
endif
```

Figure 3 Receive Function

2.2.2 Forward

The *forward* function is called either when a node has a message to send, or when a node that carries messages for a destination meets the destination, or meets another "suitable" relay node for that destination. In order to search for destinations for any of the stored messages, the *forward* function is called periodically. In *forward* state, a node first consults its routing (or contact) table to see if it has an entry for a destination. If the destination information is found, the message is forwarded to the destination (`SendMessageToDestination()`) and the node goes to *idle* state. Having not found the destination information, the node tries to find a route to the destination through its neighborhood (`SwitchNetworknCheckForDestInfo()`). For this purpose, the node may, for example, switch networks if it belongs to multiple networks. The message is sent to the destination if a route is found (`SendMessageToDestination()`). Otherwise, if a "suitable" relay node that can carry the message to the destination is found, the message is forwarded to the relay node (`ForwardMessageToRelay()`), and the current node changes its state to *idle.* If no information about the destination is found or no relay is selected and the message is not already buffered locally, the node changes its state to *buffer* and stores the message (`BufferMessage()`). Pseudo code for the *forward* function is shown in Figure 4.

2.2.3 Buffer

In this state, when a node has a message to store locally, it first checks if there is available storage, and then stores the message (`StoreMessage()`). In case the local buffer is full, the node examines the priority of the incoming message (`CheckMessagePriority()`). If the message has high priority, the node deletes the oldest and lowest priority message from its buffer. If all messages have the same priority, the oldest message is removed. If the incoming message has low priority and the buffer is already full with higher priority messages, the incoming message is discarded and the state is changed to *idle*. Figure 5 describes the pseudo code for the buffer function.

At the time of message origination, a TTL value (in seconds) is assigned to each message by the source of the message, according to its class of service. This TTL value indicates the amount of time this message is allowed to remain buffered at the storing

node, and is used for buffer management. The storing node discards the message when TTL for the message is expired. Note that the TTL mechanism doesn't require any synchronization amongst different nodes, and is used in order to avoid messages to remain buffered at nodes forever.

```
state := forward
ConsultRoutingOrContactTable()
if (Destination info is found) then
   SendMessageToDestination()
   state := idle
else
   SwitchNetworknCheckForDestInfo()
   if (Destination info is found) then
      SendMessageToDestination()
      state := idle
   else if (Relay is found) then
      ForwardMessageToRelay()
      state := idle
   else
      if (Message is already buffered) then
         state := idle
      else
         BufferMessage()
         state := buffer
      endif
   endif
endif
```

Figure 4 Forward Function

```
state := buffer
CheckStorageAvailability()
if (Buffer is not full) then
   StoreMessage()
else
   CheckMessagePriority()
   if (Message has higher priority) then
      CheckForOldestLowerPriorityMessage()
      if (Message is found) then
         RemoveOldestLowerPriorityMessage()
         StoreMessage()
      else
         RemoveOldestHigherPriorityMessage()
         StoreMessage()
      endif
   else
      CheckForOldestSamePriorityMessage()
      if (Message is found) then
         RemoveOldestSamePriorityMessage()
         StoreMessage()
      else
         DiscardMessage()
      endif
   endif
endif
state := idle
```

Figure 5 Buffer Function

2.2.4 Notification Protocol

In order to build its routing/contact tables, MeDeHa nodes use a notification protocol by which they exchange topology and content information. The main components of this protocol include vicinity discovery, neighborhood and content information exchange, and network switching. Neighborhood discovery is performed via the exchange of "hello" messages in order to see who is in the vicinity. This information could be provided by the underlying link layer protocol. Nodes engage in this exchange periodically so that if any node has messages stored for another neighboring node, these messages are delivered. Nodes also exchange "meta-information" about nodes they meet over time; this information is used as heuristics for relay selection.

MeDeHa tries to make use of node and network heterogeneity. For example, nodes that are able to participate in multiple networks, switch between them attempting to find a path to a destination or to find good relays. Switching is performed by utilizing different frequencies for each network. This can be achieved by making use of power-save mode, in case of IEEE 802.11. A similar kind of approach is proposed in [7]. MeDeHa also attempts to take advantage of more powerful nodes whenever available. For instance, MeDeHa's current implementation targets indoor scenarios (e.g. convention centers, exposition halls, museums etc.) consisting of an internet with a backbone connecting access points (APs). In these scenarios, the APs run the notification protocol to exchange information about associations and disassociations of mobile nodes.

2.3 Design Issues

MeDeHa raises a number of interesting design issues that are critical to the correct and efficient operation of the protocol. We discuss some of them here.

2.3.1 Relay node selection

As already mentioned in Section 2.1, several heuristics can be considered when choosing a suitable relay node. These include time since a node last saw the destination, frequency at which the node encountered the destination, total meeting time with destination, node's mobility pattern, node's social behavior, resources available at node including battery, storage etc. Some of these heuristics have already been reviewed in the literature [15,16], but their application is highly dependent upon the target environment. Messages can be replicated and carried by multiple relays. While having more relay nodes increases the chances of message delivery, it may cause buffer overflows and unnecessary traffic in the network. Moreover, wherever an infrastructure is available, MeDeHa gives preference to an infrastructure-based node (e.g. AP) over other relay nodes, if the destination is not directly reachable.

2.3.2 Buffer Management

Another important design decision is how to perform buffer management including how much storage space to be utilized at relay nodes, time to keep a message in storage at relay nodes, when to discard stored messages, etc. The storage space parameter depends upon the relay node's storage capacity (i.e., its storage and energy capabilities), as we can have heterogeneous devices in the system. In Section 5, the impact of varying buffer spaces is discussed and evaluated.

2.3.3 Switching Between Networks

For nodes that participate in more than one network (e.g., infrastructure and ad-hoc modes in IEEE 802.11), deciding when to switch between different networks is important. It should consider traffic demands, network and node capacity. Switching can be periodic by default with a specific amount of time dedicated to each mode. The switching operation can also be forced by a specific event, like the urgency of finding a

destination for a high priority message. Moreover, mechanisms can be built in order to adapt the switching time according to varying network conditions.

3. MeDeHa's CURRENT IMPLEMENTATION

Our current implementation of MeDeHa performs message delivery in an internet comprised of an infrastructure-based wireless network where mobile nodes roam freely and become temporarily disconnected from the network. This way, the messages are stored in the network and they are delivered as soon as a destination appears anywhere in the network. Moreover, a source, when moves and finds itself in a region of no connectivity, starts caching its messages for the destination. In this way, the source stores messages at its end, and as soon as it finds connectivity region, it starts forwarding the messages to the associated AP.

In terms of buffering in the backbone, we define two different strategies; centralized and distributed. In centralized buffering, we dedicate a machine on the backbone network to be responsible for data storage, whereas in distributed buffering, the responsibility of storing messages for a particular destination is assigned to the AP which has last seen the destination node. In this way, the responsibility of buffering keeps on changing as a node moves around in the network, which is a form of load balancing. While storing at a specific station in the backbone network requires a specific station to be dedicated for this purpose in the backbone, we have implemented this scheme to address the case when the APs are not capable of storing messages. Both these schemes (centralized and distributed) have their own advantages and disadvantages, and a comparison in between the two schemes is presented in Section 5.

A notification mechanism is required in between APs in order to send information about nodes' association/disassociation. This notification mechanism helps to find a route to a destination and to update the buffer status. The APs also collect information from their associated nodes and notify each other about the update periodically. In the following, we explain how this notification mechanism works between APs.

When a node is in the vicinity of an AP (associated with an AP), the AP informs all other APs about the presence of the node in its neighborhood by sending a *NODE_PRESENT_NOTIF* notification. This message contains the node's address, the AP's address, and notification ID. In this way, all other APs update their routes to that particular node and record the association of the node with the sending AP. Similarly, when a node leaves the vicinity of an AP (the node is disassociated from an AP), the AP updates all other APs by sending a *NODE_LEAVE_NOTIF*. The format of this message is the same as for the *NODE_PRESENT_NOTIF*.

After sending a *NODE_PRESENT_NOTIF* notification, the AP does a *pull* by requesting all others to send any stored messages for the node in question. This is done by sending a *FETCH_FRAMES_NOTIF* notification. In case of centralized buffering, this request is only sent to the central station, and for distributed buffering, the request is broadcasted to all APs. Upon reception of a *FETCH_FRAMES_NOTIF* request, the APs that have stored frames for the particular destination start sending messages. Message transmission is controlled by a timer so as to avoid sending all the stored messages at once leading potentially to congestion in the backbone. Note that it is possible that at a given point in time, more than one APs have messages for a single destination, in the case of distributed buffering mechanism. This can happen if the responsibility of storing switches to another AP before the previous AP delivers all the stored messages to the destination (an AP stops sending stored messages to a destination as soon as it receives a *NODE_LEAVE_NOTIF* notification).

The storing of messages at a central station or at any particular AP is done by sending a *STORE_FRAMES_NOTIF* notification. This message contains notification ID, destination node's address, sending AP's address, and the message to store. In case of distributed buffering, when a node goes out of the range of an AP, the AP declares itself responsible for storing messages for the node by sending a *NODE_LEAVE_NOTIF* notification. So, all the messages for the destination would be forwarded to this entity.

To be able to roam around, a node keeps on checking the received power levels of beacon frames from APs, and triggers a new association if the power level of other AP is 10% more than the power level of the current associated AP. This threshold is set in order to avoid oscillations of associations in between two APs. If a station receives beacons from only one AP, it compares the received power level with a threshold, and triggers an explicit disassociation as soon as it touches the minimum received power level. The management functionality of stations in IEEE 802.11 is modified to perform this task. Also, an explicit disassociation mechanism is incorporated in stations. In order to perform this roaming, and to enable a node to keep on listening to the beacons, all APs use the same channel and SSID.

4. EXPERIMENTAL METHODOLOGY

Today, network heterogeneity is not supported in most open-source network simulators. We use OMNET++ simulator [17], which provides basic network heterogeneity support. The version of INET Framework of OMNET++ that we are using is an extended version of the simulator[1]. We have added the handoff process in the simulator. Also, we have developed an explicit disassociation mechanism in which a station, before disconnecting from an AP, sends a disassociation frame to the AP, and then starts scanning all channels. This is done by comparing the received power with a threshold that is just above the minimum received power. The functionality of APs is also modified in OMNET++ to enable them to buffer the messages for unavailable destinations, and to implement notification protocol. We have also implemented two types of buffering mechanisms, centralized and distributed, as explained in Section 3.

The buffering policy is implemented to provide per flow and per destination priority mechanism. In this way, when an AP's buffer is full, the oldest message with lower priority is dropped. In this way, if a lower priority message is arrived and the buffer is full with higher priority messages, the incoming message is discarded (dropped).

5. RESULTS

We use packet delivery ratio (PDR) to show how MeDeHa improves message delivery in heterogeneous internets subject to connectivity disruptions. For this purpose, we consider a museum

[1] The modified version of INET Framework of OMNET++ can be found at http://planete.inria.fr/software/MeDeHa. Some scripts to perform the simulations are also available online.

environment where exhibit rooms/halls are equipped with APs. Visitors carrying portable devices move from one room to another to visit the museum and roam around between regions of coverage of the APs. These APs are connected to each other via an Ethernet switch. While changing rooms, visitors (nodes) get disconnected temporarily and hence loose some messages (announcements, etc.) destined to them. Here, we use the network to store messages temporarily. When no destination information is available, messages are stored using centralized and distributed buffering mechanism described in Section 2.3. In the first phase of implementation, the network only supports infrastructure mode with backbone network. Hence, we are able to deliver messages only when a node is associated with any of the APs. As previously discussed, there is some overhead associated with MeDeHa. This overhead includes 16-byte notification messages generated by (relays, APs etc). There is also storage overhead at each node.

To have results close to a realistic scenario, we employ Random Waypoint (RWP) mobility model with attraction points [18,19]. The attraction points are considered as rooms and the nodes move only in between these attraction points. Each attraction point is defined with zero mean and a specific standard deviation along with an intensity to select the attraction point by the RWP mobility model. The standard deviation acts as a radius for the region of influence for an attraction point. The nodes are made to move in between these attraction points at a speed that is uniformly distributed between 1 and 2.5 m/s. Also, while reaching within the coverage area of an attraction point, a node stays there for a time that is uniformly distributed between 10 and 90 seconds. A network of 9 APs is taken in consideration within a 1200x1200 area and there are 28 attraction points, each having an effective radius of 10 meters, indicating its region of influence. There are 60 nodes in the network and we have run the simulations for a duration of 40 minutes. In order to perform simulations, we create some mobility traces using random waypoint mobility model with attraction points, while utilizing BonnMotion Mobility Model tool [20].

5.1 Uniform and Non-uniform AP Distribution

In the first set of experiments, 20 sources send messages to 20 mobile destinations. Each source sends 2 flows; a high priority and a low priority flow. Messages are sent following an exponential distribution. We have observed similar behavior of the protocol with different mean exponential distribution rates. Here, we are showing the results for a mean rate of 5 packets/s and 1 packet/s, with a packet size of 1 Kbytes. There is no buffer limit at APs as the goal is to study the impact of data rates and the AP distribution.

First, we place the APs uniformly across the entire network. This means that the distance between all the APs is constant. This is done so as to have low disconnection times when nodes move representing an almost-connected network, comprised of connectivity “black holes”. The deployment of APs and that of attraction points is shown in Figure 6. The sources are static in this case, while the destinations move. The delivery ratio is shown in Figure 7.

We compare MeDeHa with the case when there is no buffering available. As is clear from the figure, with MeDeHa, 95% of nodes have more than 90% delivery ratio for the average rate of 5 packets/s (40 kbps), and 99% of nodes have more than 90% delivery ratio in case of 8 kbps. On the other hand, in case where buffering is not enabled, about 40% of nodes have less than 90% delivery ratio and 10% of nodes have even less than 50% delivery ratio, in case of 40 kbps data rate.

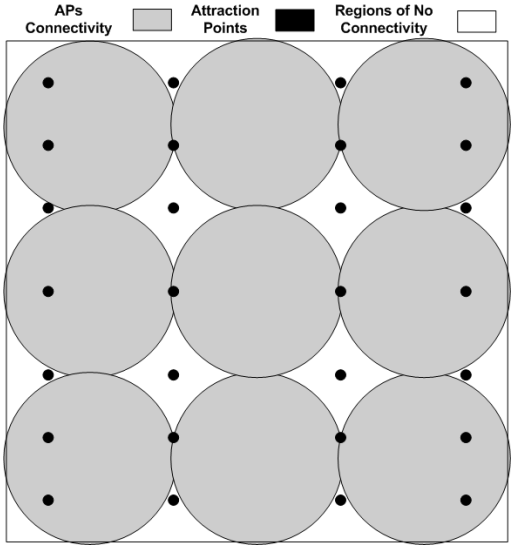

Figure 6 Uniform Deployment of 9 APs (28 Attraction Points)

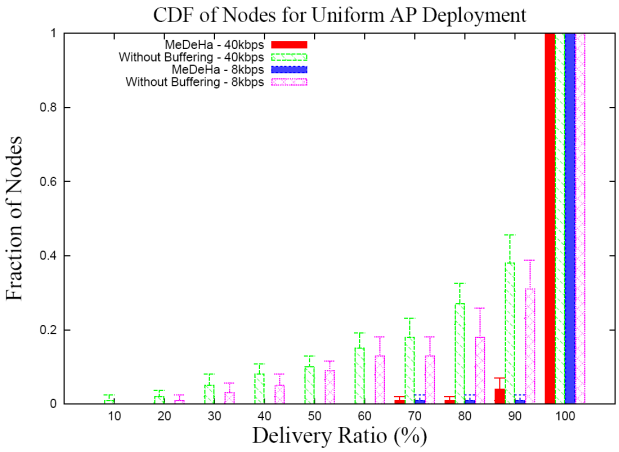

Figure 7 CDF of Nodes with Uniform APs Distribution

Next, we have considered the case when the APs are distributed in the network in such a way that the distance between APs is non-uniform. The idea was to simulate an environment where the average disconnection time is higher. Figure 8 shows the non-uniform deployment of APs for our simulations. All other simulation parameters are the same as for the previous case. The result for non-uniform deployment of APs is shown in Figure 9.

The impact of non-uniform distribution of APs on the delivery ratio for the case when the messages are not buffered is very high, as 75% of nodes have less than 80% delivery ratio, and 40% of nodes have less than 40% delivery ratio. Whereas using MeDeHa, we see that 97% of nodes have more than 90% delivery ratio, in case of 40 kbps. The behavior is similar in case of 8 kbps.

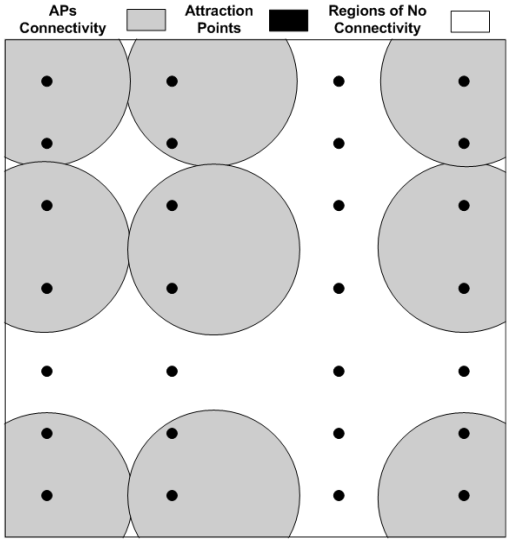

Figure 8 Non-uniform Deployment of 9 APs (28 Attraction Points)

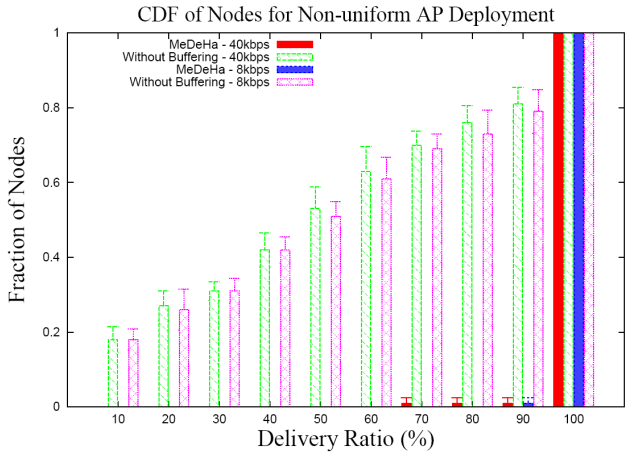

Figure 9 CDF of Nodes with Non-uniform APs Distribution

We also study the impact of source mobility on the performance of MeDeHa. If a source is mobile, it can also be disconnected from the network, and hence is not able to send any data to anyone. We have used two approaches for this case, namely: (1) caching messages at sources when they are disconnected, along with buffering in the network; and (2) disabling network buffering, and enable sources to buffer data while moving in the network. All the 20 sources are mobile, while all other parameters remain the same. We evaluate this scenario with non-uniform deployment of APs. The result for the average message rate of 40 kbps is shown in Figure 10. The behavior for other message rates is observed to be similar.

We see that with MeDeHa, when buffering is provided at sources and in the network, 96% of the nodes have more than 90% PDR. When the buffering is only present at the sources, 40% of the nodes have less than 70% delivery ratio, and when no buffering is present, only 20% of the nodes have more than 90% PDR, and 30% of the nodes have even less than 40% PDR.

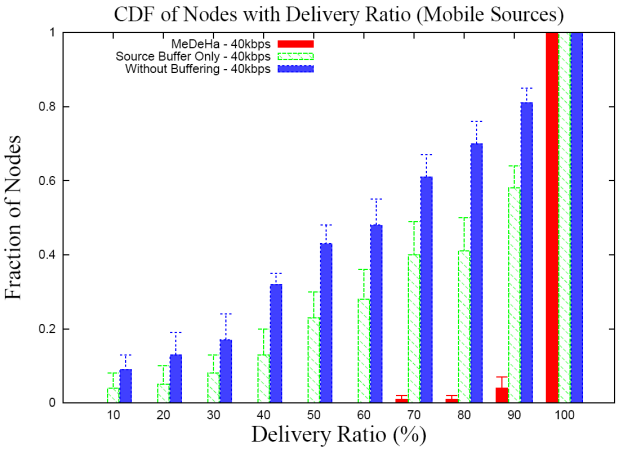

Figure 10 CDF of Nodes with Mobile Sources

5.2 Buffer Size

The choice of buffer size highly depends on application's message rates, as well as on delivery ratio requirements. We have evaluated the behavior of MeDeHa by observing the PDR as a function of different buffer sizes, both with centralized and distributed buffering schemes. It is also interesting to observe the impact of buffer sizes on traffic flows with different priorities. For this purpose, we use two flows per source (high and low priority), and the simulation parameters are the same as mentioned before. The impact of buffers sizes is also observed for the uniform and non-uniform deployment of APs. This is important as it becomes clear when we explain the obtained results in the following.

As mentioned in the previous subsection, average disconnection time of mobile nodes is directly related to the deployment of APs. More the nodes remain disconnected, the more is the buffer size required to store messages for these nodes. Hence, we say that in this case, we require more space to store messages. To analyze the impact of buffer sizes in centralized and distributed buffering, we take equal buffer sizes. This means that we divide the buffer size of centralized station equally among all APs, in case of distributed buffering. Thus, we say that:

$$S_c = \sum_i S_{d_i}$$

Where, S_c = the size of the buffer for centralized buffering, and

S_d = the size of the buffer for distributed buffering.

By evaluating the protocol for non-uniform AP deployment, we get the results shown in Figure 11. The results are taken for 20 source-destination pairs with mean message rate of 40 kbps per flow per source, and messages are exponentially distributed.

Here, in case of centralized buffering, for higher buffer sizes (e.g. 6 Mbytes), both low and high priority flows have obtained more than 95% PDR. But as we reduce the size of the buffer, the low priority traffic gets more affected than high priority traffic, until we reach at buffer sizes, where the buffering scheme has to drop some high priority messages; hence a reduction in PDR.

Same simulation is performed for distributed buffering scheme, but we see that the performance is not as good as in case of the centralized buffering. There are two main reasons behind this change in behavior. One is that the APs are not uniformly

deployed. Hence, for some APs, when they get the responsibility to store messages for a destination that gets disconnected for a longer period of time, it is likely that their buffer gets full and hence, they drop some messages. The impact is more at very low buffer sizes (1Mbyte for 9 APs would mean that each AP has only 111 Kbytes storage space, and can store only 111 messages). The second reason is that it is possible that some nodes remain disconnected for longer period of time, and hence they require more storage space at APs than others. So, it is possible that at a given time, one of the APs has more messages to buffer than its capacity while some other APs have all their storage space available. This case cannot be avoided in distributed buffering, and it doesn't occur when we store messages at a central place.

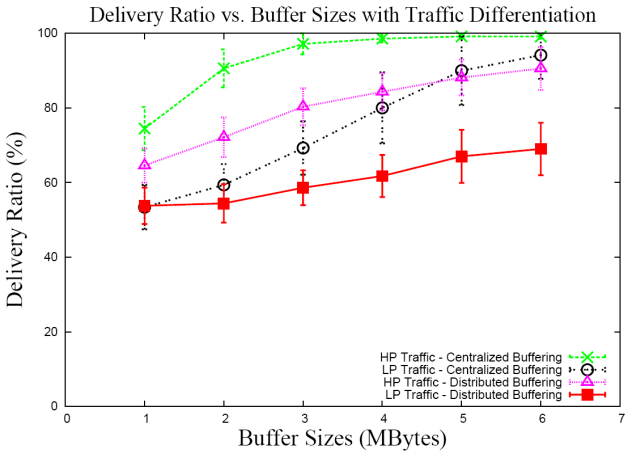

Figure 11 Buffer Size Impact on PDR (Non-uniform APs Deployment)

Next, the impact of buffer sizes has been observed in case of uniform APs deployment. When comparing centralized and distributed buffering schemes, similar behavior is observed with two main changes. First, the first reason that we described above is not present in this case. The second change is that the size of buffers required to store message is reduced, as the average disconnection time is reduced. The results are shown in Figure 12.

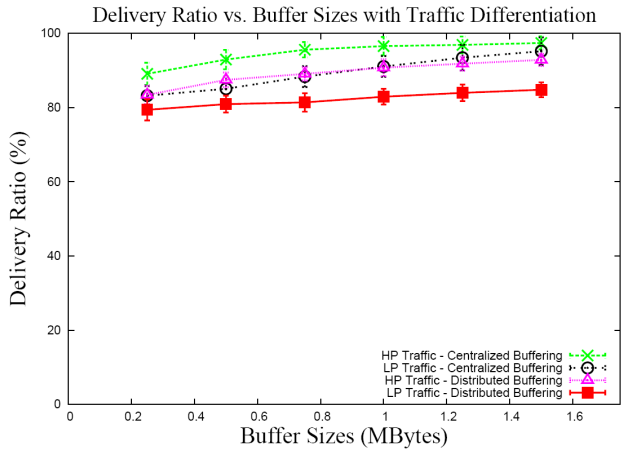

Figure 12 Buffer Size Impact on PDR (Uniform APs Deployment)

6. RELATED WORK

Most of the studies that target network heterogeneity in 802.11, aim towards extending connectivity area and increasing network capacity. To extend the network connectivity beyond regions covered by APs, these proposals use different mechanisms such as the use of different frequencies for each mode in Flex-Wifi [7], and a new layer between IP and link layer in MultiNet [8]. AODV+ [9] proposes a scheme to connect the Internet backbone to MANETs by introducing a gateway discovering mechanism. The common problem in all these schemes is the failure to deliver data in the presence of frequent network partitioning.

CAPWAP [21] and WINLAB [22] introduce the concept of enhancing APs with further functionality like caching, consistent management and configuration etc. These proposals argue in favor of extending the link layer features for APs. Another study [23] proposes combining the CAPWAP and WINLAB architectures by caching data at centralized entities (AC) when stations roam around. The architecture only targets infrastructure mode communications and handles frequent disruptions. The study also proposes to pre-fetch future data from ACs while connected, which may not be practical in most cases, including real-time scenarios. The deployment of these ACs is not trivial and management of these ACs for data storing and handling is problematic.

The seminal work of the IRTF's Delay-Tolerant Networking Research Group (DTNRG) pioneered research on DTNs with their delay-tolerant network architecture [11] a.k.a. Bundle Architecture. Their proposal is based on bundle switching with the ability to store bundles in transit for arbitrarily long periods of time. This is referred to as store-carry-and-forward. Storage is performed above the transport layer to provide interoperability among networks that support different types of transport layers. Our mechanism is orthogonal to the Bundle architecture that can be used with MeDeHa to support networks with different transport layers. In such cases, it is useless to store data at link layer of nodes that act as DTN routers or gateways. But the need to store messages at lower layers in other nodes of network would still be the same, and MeDeHa would be useful especially when intermediate nodes don't support higher layers, and where the Bundle layer mechanism cannot be incorporated.

Propositions exist to integrate DTNs with MANETs. Ott et al. [2] introduce specialized DTN capable end point nodes to bridge islands of networks, but this solution doesn't provide backbone connectivity. Natasa et al. [1] use the mobility patterns of the nodes over time to make nodes communicate in between different islands, but again, with the help of nodes that move in between these islands. Besides, some studies use the concept of node relaying in order to bridge otherwise partitioned networks. These propositions include message ferries [12], throwboxes [14], and use of data mules [13]. They suggest the use of specialized nodes, fixed or mobile that are used as data carriers, and/or forwarders. Specialized nodes are resourceful entities (storage space, battery power etc). The concept is very fruitful in increasing the delivery ratio, and in some cases, reducing the overall delay, but the problem of number of these special-purpose nodes in the network, and their routes is not trivial.

7. CONCLUSION & FUTURE WORK

Providing robust message delivery in heterogeneous internets subject to intermittent connectivity may be desirable in many scenarios, where late delivery is preferred over loss of data. This work is an important building block to enable current and upcoming applications in such scenarios. Our contributions are two fold. First, we address the problem of frequent and/or long-lived connectivity disruptions in heterogeneous networks. Current proposals targeting network heterogeneity don't deal with arbitrarily long connectivity interruptions. Second, with our scheme, there is no need to introduce special-purpose nodes in order to connect to the backbone network, or to support network heterogeneity. This is significant, as t having extra, more resourceful entities in the network (e.g., Access Points) is completely transparent.

We are currently extending MeDeHa with ad hoc network support. In the longer run, we will address multi-destination and QoS-based data delivery.

8. ACKNOWLEDGEMENTS

This research work is partially funded by French ANR Divine Project. Besides, the work has been partly supported by the Army Research Office (ARO) under a MURI project named DAWN, the Baskin Chair in Computer Engineering, NSF grants ANI 0322441 and CNS 0534129.

9. REFERENCES

[1] N. Sarafijanovic-Djukic, M. Piorkowski, and M. Grossglauser, "Island Hopping: Efficient Mobility-Assisted Forwarding in Partitioned Networks", Proc. of IEEE SECON, 2006.

[2] Jörg Ott, Dirk Kutscher, Christoph Dwertmann, "Integrating DTN and MANET Routing", Proc. of ACM SIGCOMM workshop on Challenged Networks (CHANTS), 2006.

[3] A. Vahdat and D. Becker, "Epidemic routing for partially connected ad hoc networks", Technical Report CS-200006, Duke University, 2000.

[4] T. Spyropoulos, K. Psounis, C.S. Ragha Vendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks", ACM SIGCOMM Workshops WDTN, Philadelphia PA, August 2005.

[5] J.-C. Chen, S. Li, S.-H. Chan, J.-Y. He, "WIANI: wireless infrastructure and ad-hoc network integration", in: Proc. IEEE International Conference on Communications, Seoul, Korea, 2005, pp. 3623-3627.

[6] J He J. Chen, S.-H. G. Chan and S.-C. Liew. "Mixed-mode wlan : The integration of ad hoc mode with wireless LAN infrastructure", IEEE Globecom 2003.

[7] Carlo Parata, Gabriella Convertino, Vincenzo Scarpa, "Flex-WiFi: a mixed infrastructure and ad-hoc IEEE 802.11 network for data traffic in a home environment", The First IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications, 2007.

[8] R. Chandra, P. Bahl, and P. Bahl, "MultiNet: Connecting to Multiple IEEE 802.11 Networks Using a SingleWireless Card", in IEEE Infocom, Hong Kong, 2004.

[9] U. Korner A. Hamidian and A. Nilsson. "Performance of internet access solutions in mobile ad hoc networks", Dagstuhl-Workshop "Mobility and Wireless in Euro-NGI", pages 189–201, 2005.

[10] K. Scott, S. Burleigh, "RFC 5050, Bundle Protocol Specifications", IRTF DTN Research Group, November 2007.

[11] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, H. Weiss, "RFC 4838, Delay-Tolerant Networking Architecture", IRTF DTN Research Group, April 2007.

[12] W. Zhao , M. Ammar , E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks", Proc. of ACM/IEEE MOBIHOC, 2004.

[13] R. Shah, S. Roy, S. Jain, W. Brunette, "Data MULEs: Modeling a Three-tier Architecture for Sparse Sensor Networks", IEEE SNPA Workshop, May 2003.

[14] Wenrui Zhao, Yang Chen, Mostafa Ammar, Mark Corner, B.N. Levine, and Ellen Zegura, "Capacity Enhancement using Throwboxes in DTNs", IEEE International Conference on Mobile Ad hoc and Sensor Systems (MASS), Vancouver, Canada, October 2006.

[15] T. Spyropoulos, T. Turletti, K. Obraczka, "Utility-based Message Replication for Intermittently Connected Heterogeneous Networks", The first International IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC), Helsinki, Finland, June 2007.

[16] Matthias Grossglauser, Martin Vetterli, "Locating Mobile Nodes with EASE: Learning Efficient Routes from Encounter Histories Alone", IEEE/ACM Transactions on Networking, Vol. 14, No. 3, June 2006.

[17] OMNET++, http://www.omnetpp.org.

[18] Christian Bettstetter and Christian Wagner, "The Spatial Node Distribution of the Random Waypoint Mobility Model", In Proceedings of the First German Workshop on Mobile Ad-Hoc Networks (WMAN), GI Lecture Notes in Informatics, P-11, 41–58.

[19] M. Feeley, N. Hutchinson, and S. Ray, "Realistic Mobility for Mobile Ad Hoc Network Simulation," LNCS 3158, pp. 324–329, 2004.

[20] BonnMotion, Univerysity of Bonn, "A mobility scenario generation and analysis tool", http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion.

[21] L. Yang, P. Zerfos, E. Sadot, "RFC 4118, Architecture Taxonomy for Control and Provisioning of Wireless Access Points CAPWAP)", Network Working Group, 2005.

[22] "Infostations: A case study of winlab's systems approach to research". http://www.winlab.rutgers.edu/. July 2007.

[23] Mazen Tlais, "Discontinuous Coverage Architecture, Challenges, Design & Evaluation", Phd Thesis, University of Rennes-1, 2007.