Wireless Netw (2011) 17:1775–1794
DOI 10.1007/s11276-011-0377-0

Message delivery in heterogeneous networks prone to episodic connectivity

Rao Naveed Bin Rais · Thierry Turletti · Katia Obraczka

Published online: 17 August 2011

Abstract We present an efficient message delivery framework, called MeDeHa, which enables communication in an internet connecting heterogeneous networks that is prone to disruptions in connectivity. MeDeHa is complementary to the IRTF's Bundle Architecture: besides its ability to store messages for unavailable destinations, MeDeHa can bridge the connectivity gap between infrastructure-based and multi-hop infrastructure-less networks. It benefits from network heterogeneity (e.g., nodes supporting more than one network and nodes having diverse resources) to improve message delivery. For example, in IEEE 802.11 networks, participating nodes may use both infrastructure- and ad-hoc modes to deliver data to otherwise unavailable destinations. It also employs opportunistic routing to support nodes with episodic connectivity. One of MeDeHa's key features is that any MeDeHa node can relay data to any destination and can act as a gateway to make two networks inter-operate or to connect to the backbone network. The network is able to store data destined to temporarily unavailable nodes till the time of their expiry. This time period depends upon current storage availability as well as quality-of-service needs (e.g., delivery delay bounds) imposed by the application. We showcase MeDeHa's ability to operate in environments consisting of a diverse set of interconnected networks and evaluate its performance through extensive simulations using a variety of scenarios with realistic synthetic and real mobility traces. Our results show significant improvement in average delivery ratio and a significant decrease in average delivery delay in the face of episodic connectivity. We also demonstrate that MeDeHa supports different levels of quality-of-service through traffic differentiation and message prioritization.

Keywords Disruption tolerance · Episodic connectivity · Heterogeneous networks · Node relaying · Store-carry-and-forward · DTN routing

R. N. B. Rais (✉)
COMSATS Institute of Information Technology (CIIT), Lahore, Pakistan
e-mail: naveedbinrais@ciitlahore.edu.pk

T. Turletti
INRIA, Sophia Antipolis, France
e-mail: thierry.turletti@sophia.inria.fr

K. Obraczka
University of California, Santa Cruz, CA, USA
e-mail: katia@soe.ucsc.edu

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 it interconnects, but also in terms of the underlying networks it comprises. Figure 1 illustrates networks that range from wired- and wireless backbones (e.g. community wireless mesh networks) to wireless infrastructure-based and ad-hoc networks (e.g., MANETs). On the other hand, 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 important component of future internetworks.

Seamless interoperability among heterogeneous networks is a challenging problem as these networks may have very different characteristics. Node diversity may also

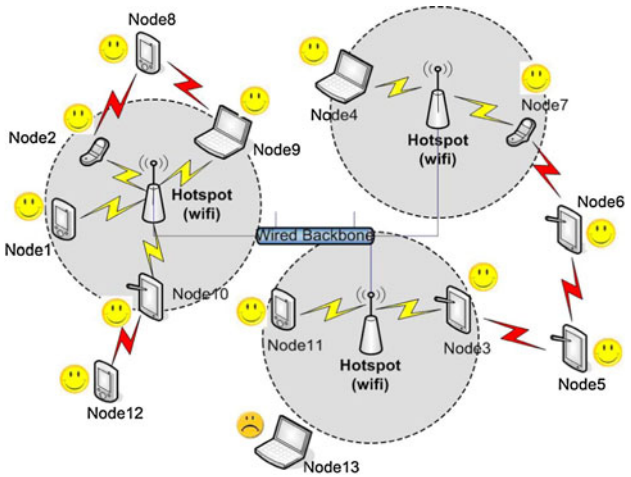

Fig. 1 An example of a heterogeneous internetwork with a wired backbone, wireless infrastructure-based, and ad-hoc networks prone to episodic connectivity

make routing difficult, as nodes must also take into account available resources at other nodes. Moreover, nodes require to avail contact opportunities (given that links are time-varying due the possibility of intermittent connectivity) to make correct routing decisions. For instance, in a buffer-constrained network where participating nodes may have different buffering capabilities, it is useless to forward a message to a neighboring node, if the latter is running out of buffer space.

There are no comprehensive solutions targeting message delivery in heterogeneous networked environments prone to connectivity disruptions.[1] Existing proposals either: (1) extend MANETs to handle episodic connectivity [1–4], (2) augment the coverage of access points in infrastructure-based (IS-based) wireless networks by, for example, making use of multi-channel radios or switching from infrastructure mode in 802.11 [5–8], or (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].

In this paper we propose MeDeHa (Message Delivery in Heterogeneous, Disruption-prone Networks)[2]—a general, 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, they get all messages destined to them. MeDeHa is complemetary to the Bundle Architecture [10, 11]: in addition to storing messages for unavailable destinations. MeDeHa also provides an integrated forwarding mechanism that bridges infrastructure-based and infrastructure-less networks. To-date the Bundle Architecture makes available, as external modules, forwarding mechanisms that are robust to connectivity disruptions. However, it does not have the capability of bridging infrastructure-based and multi-hop infrastructure-less networks. We use the terms infrastructure-less and ad-hoc interchangeably in this paper. The framework 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.

MeDeHa employs opportunistic routing to support nodes with episodic connectivity. In other words, nodes using the MeDeHa framework make a best effort to carry data towards the destination based on the contact opportunities that a source or a relay has. Moreover, any node in MeDeHa can act as a relay for any destination, and can serve as a gateway to bridge different networks that it is capable to connect. So, any node can provide backbone connectivity too. 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 making use of existing nodes with special capabilities (e.g., access points, or APs in the case of IS-based wireless networks) that are an integral part of the underlying network. Whenever available, MeDeHa utilizes nodes with more resources and capabilities like APs to perform message delivery more efficiently, but does not depend on them. Furthermore, we take advantage of the underlying heterogeneity (e.g., in the context of IEEE 802.11 networks, a node's ability to operate in infrastructure or ad-hoc modes) to enable message delivery across different networks.

This paper extends our preliminary work where scenarios of limited heterogeneity were addressed and evaluated [15]. In this paper, we explore significantly higher degrees of network heterogeneity including networks comprising wired as well as infrastructure-based and multi-hop ad-hoc wireless networks that are subject to intermittent connectivity. The contributions of this paper over our previous paper [15] are as follows: (1) We present comprehensive design of the MeDeHa framework that allows message delivery in disruption-prone networks as well as in infrastructure-based and ad-hoc networks. (2) We design a 2-hop notification protocol that incorporates existing DTN-based forwarding mechanisms in infrastructure-less networks. (3) We use more realistic scenarios to evaluate MeDeHa's performance using both synthetic and real mobility traces. We evaluated MeDeHa through extensive simulations using a variety of synthetic as well as real-world scenarios. Our results show that end-to-end delay can be improved significantly while maintaining high

[1] More details on the related work are presented in Sect. 6.

[2] This work has been done at INRIA, Sophia Antipolis.

delivery ratio. This is accomplished by selecting appropriate relays when forwarding data, taking advantage of in-network storage as well as node diversity and network heterogeneity (e.g., nodes with more resources, nodes that can switch between infrastructure and ad-hoc communication modes).

The remainder of this paper is organized as follows: Sect. 2 provides a overview of MeDeHa's framework while MeDeHa's protocol description is presented in Sect. 3. The implementation approaches as well as MeDeHa's current implementation is described in Sect. 4. Section 5 presents simulation results reporting the performance of MeDeHa using a variety of scenarios involving both synthetic—and real mobility traces. Related work is reviewed in Sect. 6 and finally, concluding remarks and some future directions are discussed in Sect. 7.

2 MeDeHa overview

MeDeHa allows message delivery across heterogeneous networks by accommodating a diverse set of characteristics in terms of mobility, connectivity, and resources. Heterogeneity is supported both at the network—and at node level. At the network level, MeDeHa allows co-existence of different types of networks like wireless IS-based as well as ad-hoc networks and intermittently connected networks. At the node level, nodes with diverse resources like battery power, buffering or mobility characteristics can be part of the network.

MeDeHa embraces node- and network heterogeneity and tries to make use of it whenever possible. For example, it tries to take advantage of more resourceful nodes (e.g., APs in IEEE 802.11 IS-based networks) whenever possible and feasible. Additionally, a node that participates in multiple networks will attempt to find a path (or suitable relays) to a destination in all networks of which the node is a member.

MeDeHa's main functional components are:

Message relaying and forwarding: MeDeHa nodes can relay messages under the store-carry-and-forward paradigm [11], and can be used to connect to the backbone network. We thus avoid using any explicit discovery mechanism for finding specialized nodes (e.g., gateway to the backbone). Message delivery is improved by taking advantage of network heterogeneity. This is achieved with the help of the MeDeHa-capable nodes that are able to connect simultaneously to more than one network (e.g., using multiple interfaces). Besides, the MeDeHa nodes may also switch between multiple networks using the same interface card. For example, 802.11-capable nodes may join different networks by switching between infrastructure- and ad-hoc modes by using different frequencies. This can be done, e.g., using the power save mode of the IEEE 802.11 standard [8].

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 act as a relay and therefore 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 IS-based wireless networks or mesh routers in the case of wireless mesh networks, are usually good candidates to serve as temporary storage for undelivered messages as they exhibit higher resource availability.[3]

In MeDeHa, buffering can be done at any layer of the communication stack, which enables almost any network-enabled device to relay and buffer messages. This feature allows MeDeHa to be implemented on nodes that run only the lower two or three protocol layers (e.g., AP bridges and routers). Moreover, quality-of-service can be 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 and contact 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). Routing tables are used to keep information on the connected nodes, whereas contact tables are maintained to keep a history of nodes encounters for a pre-defined period of time that may be used in the relay selection process. Entries in the contact tables are removed when expired. Nodes also exchange a summary of their message buffer and their current state in terms of resources (e.g., how much storage left, remaining battery lifetime, etc.). All this information is used in the relay selection process [16–19] and contributes to the overhead incurred by MeDeHa. Clearly, 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 in IEEE 802.11 infrastructure mode), MeDeHa simply makes use of it.

[3] 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 alternative given current AP technology.

Traffic differentiation: To satisfy application specific needs, 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), 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 the four states of MeDeHa's operation.

Idle: 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*.

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, 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 destination-dependent and destination-independent 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, and whether the scope of the mobility is "local" or "global", (4) how "social" a node is, etc. A number of these heuristics or utility functions has been presented recently [16]. MeDeHa's framework is flexible to employ any kind of utility function for choosing a relay to carry a message to a destination. When selecting relays, MeDeHa can also account for the underlying heterogeneity among participating nodes, e.g., the amount of available resources such as storage, processing, and battery lifetime. For instance, more resourceful entities (like APs) may be preferred when messages need to be stored.

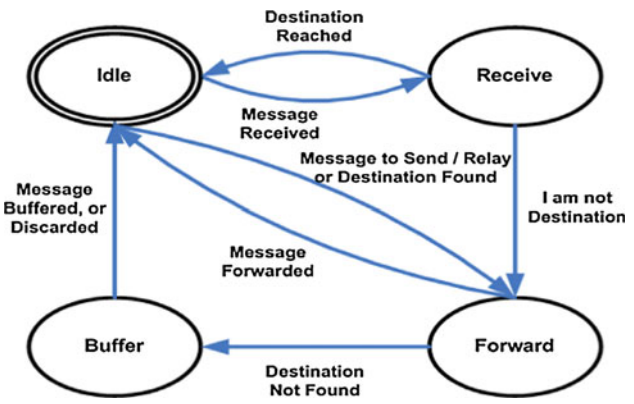

Fig. 2 State diagram showing MeDeHa's overall operation. A MeDeHa-capable node can be in one of the four states, *Idle, Receive*, and *Forward*

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

Buffer: A node is in *buffer* state when it has a message to store for an unavailable destination. MeDeHa's buffering mechanism is based on message priorities and time-to-live (TTL) values. The node goes back to *idle* state whether the message is buffered or discarded. The MeDeHa nodes make use of different buffer management strategies based, for example, on the application QoS requirements such as message priority and TTL.

2.2 MeDeHa's state diagrams

Receive state: MeDeHa's receive functionality is shown in Fig. 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()), i.e., forwards the message to the application layer, and switches back to *idle* state. Otherwise, it switches to *forward* state.

Forward state: 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. Thus, the forward function is called at each contact opportunity that the message carrier experiences. The function is also called when a node receives a message but it is not the intended destination of the message. In *forward* state, a node first consults its routing table to see if it has an entry for a destination. If the destination information is

```
1:  state ← idle
2:  if message received then
3:      state ← receive
4:      if node is destination then
5:          ConsumeMessage()
6:          state ← idle
7:      else
8:          state ← forward
9:          Forward()
10:     end if
11: end if
```

Fig. 3 MeDeHa receive function

found, the message is forwarded to the destination (*SendMessageToDestination()*) and the node goes to *idle* state. Otherwise, the node consults in contact table to see if some information is available to select a "suitable" relay or tries to find a route to the destination through its neighborhood, and if the relay is found, the message is forwarded to the relay (*SendMessageToRelay()*), 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 Fig. 4.

Buffer State: When a node has a message to store, it first checks if there is available storage, and then stores the message (*StoreMessage()*). In case the local buffer is full, it checks the priority of the incoming message by looking at the message tag (*CheckMessagePriority()*), and then checks for the oldest message having lower or equal priority. If it finds a message with lower or equal priority, it stores the incoming message by removing the oldest message from the buffer and switches the state to *idle*. If the incoming message has a 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 does not require any synchronization amongst

```
1: state ← forward
2: ConsultRoutingTable()
3: if destination info is found then
4:     SendMessageToDestination()
5:     state ← idle
6: else
7:     ConsultContactTable()
8:     if destination info is found then
9:         SendMessageToRelay()
10:        state ← idle
11:    else
12:        if message is already buffered then
13:            state ← idle
14:        else
15:            state ← buffer
16:            BufferMessage()
17:        end if
18:    end if
19: end if
```

Fig. 4 MeDeHa forward function

```
1: state ← buffer
2: if buffer is not full then
3:     StoreMessage()
4: else
5:     CheckMessagePriority()
6:     CheckForOldestLowerPriorityMessage()
7:     if message is found then
8:         RemoveOldestLowerPriorityMessage()
9:         StoreMessage()
10:    else
11:        CheckForEqualPriorityMessage()
12:        if message is found then
13:            RemoveEqualPriorityMessage()
14:            StoreMessage()
15:        else
16:            DiscardMessage()
17:        end if
18:    end if
19: end if
20: state ← idle
```

Fig. 5 MeDeHa buffer function

different nodes, and is used to avoid messages to remain buffered at nodes forever.

3 MeDeHa protocol description

This section describes in detail the protocol that implements MeDeHa's functional components.

3.1 Notification protocol

MeDeHa's notification protocol plays a key role in seamless message delivery across multiple heterogeneous interconnected networks. This is illustrated in the example of Fig. 6. It collects information about a node and its neighborhood and shares that information with other nodes by exchanging *notification messages* (described below). Neighborhood information is then used by MeDeHa-capable nodes to construct their routing or contact tables. In the current MeDeHa implementation, the notification protocol is run at the network layer and operates on more than one interface, where each interface may have a different network identifier such as an IP address. The implementation issues of MeDeHa will be discussed in Sect. 4.

Figure 6 illustrates a possible network topology. The access point (AP) gathers two-hop network information from the nodes that are associated to it. The AP then uses the associated nodes (node **G** in the example)[4] to forward a message to a node (in this case, node **D**) that is connected through one of the associated nodes (node **G**). This

[4] Note that node **G** can use a single interface card to connect to two different networks [8], or it can be connected to a cellular base station and uses 802.11 card to connect to an ad-hoc network.

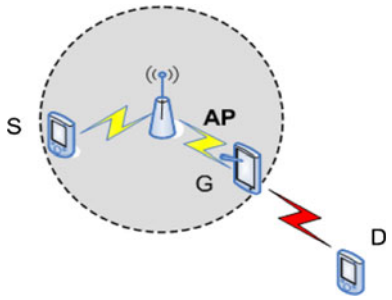

Fig. 6 Multi-hop message delivery involving IS-based and "ad-hoc" nodes that may be intermittently connected. Source *S* wants to send a message to destination *D*. This is made possible with the help of node *G* that acts as gateway between the two networks. *S* and *D* do not need to be connected to more than one network nor be part of the same network to send or receive messages

particular example shows that MeDeHa extends message delivery beyond the range of access points in IS-based networks to destinations that can only connect (intermittently) on ad-hoc mode.

MeDeHa's notification protocol has two main components: the *neighborhood sensing* and the *neighborhood information exchange*. Each component is described below.

Neighbor sensing: If neighbor detection is provided by the underlying network (e.g., beaconing and management messages in IEEE 802.11 IS-based networks), MeDeHa can take advantage of that information. For instance, in the case of IEEE 802.11 infrastructure mode, a node senses the presence of an nearby AP when it is *associated* with the AP at the link layer. This information is forwarded to the network layer as soon as the presence of a node (a station or an AP) is detected. On the other hand, a link disconnection is detected when a node is *disassociated* with an AP. Thus, in IS-based network, neighbor sensing is performed implicitly with the help of underlying link-layer protocol.

In MeDeHa-capable ad-hoc networks, neighbor (or link) sensing is done using *HELLO* notification message exchange. Nodes periodically broadcast *HELLO* notifications to inform other nodes in the neighborhood (if any) about each other's presence. In MeDeHa's current implementation, the *HELLO* notification interval is empirically set to 2 s, by default. In an effort to minimize the overhead incurred by the protocol, information in *HELLO* messages is kept to a minimum and may include:

- *Node identifier(s) (e.g., IP addresses)* Nodes may announce multiple identifiers if they have more than one.
- *Infrastructure affiliation indicator* A flag indicating whether transmitting node is currently affiliated (associated) with an IS-based network.
- *Identifier of IS-based node* In case of affiliation with an IS-based network, identifier of the associated IS-based node (e.g., AP).
- *Memory status* Available memory in number of bytes.
- *Energy level* An indication about the status of the node's current power capacity (e.g., remaining battery life).
- *Node utility* This metric is used to announce to other nodes for the set of utility functions that is supported by the transmitting node. It helps in making better decisions for selecting relays. For instance, this can be an indicator of the node's mobility behavior (e.g., bus, pedestrian, car etc.), or its affiliation to a particular community (e.g., city, village etc.) or an organization. Details are provided in Sect. 3.3.

Note that all fields are optional except the node identifier field.

Neighborhood information exchange: The *HELLO* notification only contains information about the *HELLO*-originating node, and not about its neighborhood. As previously mentioned, this is done to limit protocol overhead; this is especially beneficial in the case of highly partitioned networks. Having received a *HELLO* notification, a "hello handshake" process starts, where two nodes exchange their neighborhood information by sending a *NEIGHBOR_INFO* unicast notification, as shown in Fig. 7. In this way, the node with lower ID announced in its *HELLO* sends the *NEIGHBOR_INFO* notification first. This completes the handshake between two neighboring nodes and also eliminates uni-directional wireless links implicitly. A *NEIGHBOR_INFO* notification message may contain any combination of the following:

- *CURRENT_NEIGHBORS* List of 1-hop neighbor identifiers minus the identifier(s) of the node to which the message is being sent. If the transmitting node has no neighbors except the one to which the *NEIGHBOR_INFO* is sent, this notification is not included.

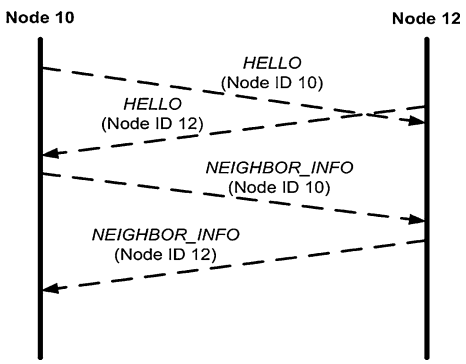

Fig. 7 Hello handshake mechanism between node 10 and node 12. Node 10 wins and sends the NEIGHBOR_INFO notification before Node 12

Table 1 Notification information exchanged for ad-hoc networks

Notification name	Includes	Contents	Description
HELLO		-Node IDs -FlagAssociated -Affiliated IS-based node's ID -Buffer level -Energy level	Periodically broadcasted by each node to inform neighboring nodes about its IDs
NEIGHBOR_INFO	CURRENT_NEIGHBORS	-IDs of neighbors	Sent in response to *HELLO* to inform receiving node about neighboring nodes
	RECENT_NEIGHBORS	-IDs of encountered nodes -Encounter time -Number of encounters *-Any other heuristic*	Sent in response to *HELLO* to inform receiving node about the nodes recently seen by the transmitting node
	MSG_VECTOR	-Sequence no. of messages -Source of messages -Destination of messages	Sent in response to *HELLO*, and contains sequence numbers of messages stored at transmitting node

- *RECENT_NEIGHBORS* List of node identifiers who have been encountered within a pre-defined period of time. It may also include additional information related to encountered nodes (e.g., number of encounters, encounter time, social affiliation of node, speed of nodes etc.), which are used in computing the *utility function* employed in relay selection. We provide the details of this function in Sect. 3.3. If the transmitting node has not encountered any node in the specified period of time, or all its contact table entries are expired, this notification is not included.
- *MSG_VECTOR* List of of application-layer message identifiers (sequence numbers, source and destination identifiers, and ports). This notification may be sent to avoid forwarding a message to a node (relay) that already has a copy of it. This is used with a multi-copy replication scheme to reduce unnecessary message duplication.[5] If the transmitting node buffer is empty, this notification is not included.

The *MSG_VECTOR* notification contains only a list of message identifiers (described above) for messages stored at the advertising node, instead of the actual messages. After exchanging the list of messages, the advertising node decides which message(s) the other node is missing. Then, they exchange only the missing messages that pass the relay selection criteria. Messages could also be identified by message digests that could also be used as a security mechanism to prevent message tempering by intermediate nodes. Note that *MSG_VECTOR*s are generated "on-the-fly" upon an encounter and are not stored at the nodes. The *MSG_VECTOR* notification is sent by the node that supports multi-copy. The node does not consider whether the peer node supports multi-copy or not. When two nodes meet, the node that sends the *MSG_VECTOR* notification supports multi-copy and will not receive duplicate messages from the peer node. This is because the purpose of the *MSG_VECTOR* notification is to prevent message duplication at the node that sends the notification.

Table 1 summarizes the different notification messages exchanged in MeDeHa-capable ad-hoc networks. We assume that each MeDeHa node recognizes each control notification, though it is not mandatory to send all control notifications in the *NEIGHBOR_INFO* message exchange. Note that the exchange of neighborhood information in ad-hoc mode allows each node to keep two-hop neighborhood information.

Note that the control messages of the MeDeHa's notification protocol are only meant to be exchanged between neighboring nodes, for example between nodes from an IEEE 802.11 Extended Service Set (ESS). We do not assume transmission of control messages over the Internet. In the case of IS-based networks, neighborhood information is exchanged between a node and its *associated* IS-based node (e.g., AP) and among IS-based nodes that are connected (either wired or wireless). The notification messages between IS-based nodes are triggered on the reception of a connection or a disconnection event (e.g., *NODE_PRESENT*, *NODE_LEAVE* etc.).[6] The notification

[5] Note that following traditional epidemic routing, two nodes, upon encountering each other, exchange the list of all the messages each one stores. Since local storage is limited, typically messages have a Time-to-Live associated with them, beyond which they are discarded. To prevent waste of memory resources, each stored message has an expiry time associated to it. Moreover, buffer management and flow control mechanisms for DTN such as HBSD [20] can be employed to schedule message transmission during contact time.

[6] These notifications are defined in Table 2.

Table 2 Infrastructure-based Notification Protocol Messages

Notification name	Originator	Destination	Description
ASSOC	Node	IS-based node	Notification sent to network layer as soon as a node is connected to an IS-based node
NODE_PRESENT	IS-based node	IS-based node	Upon arrival of *ASSOC*, this notification is sent to all other IS-based nodes to inform about a node's connection (association)
NODE_LEAVE	IS-based node	IS-based node	This notification may be sent when a disassociation process is completed (implicit or explicit)
FETCH_FRAMES	IS-based node	IS-based node	On the arrival of a *ASSOC*, an IS-based node may send this notification to other IS-based nodes asking about any stored messages
NEIGHBOR_PRESENT	Node	IS-based node	This notification is sent from an node to its affiliated IS-based node, and contains information about immediate neighbors of the transmitting station
INDIRECT_ASSOC	IS-based node	IS-based node	This notification is sent on the reception of *NEIGHBOR_PRESENT* to inform other IS-based nodes about an indirect association
NEIGHBOR_LEAVE	Node	IS-based node	As soon as departure of a neighboring node is detected, this notification is sent from an associated node to its IS-based node

messages between a node and its associated IS-based node may result from a link layer association of the node (e.g., *ASSOC* in Table 2), or based on sensing a neighboring node in ad-hoc mode (e.g., *NEIGHBOR_PRESENT*). Nodes that pass their one-hop neighborhood information to their associated IS-based nodes act as gateways to connect IS-based networks with nodes in ad-hoc mode. Notification messages that are exchanged in an IS-based network are presented in Table 2. In the specific case of IEEE 802.11 ESS, the notification protocol messages exchanged amongst APs are broadcast and confined to APs.

3.2 Routing and contact table management

In MeDeHa, each node maintains routing and contact tables which are built using information from neighbor sensing and neighborhood information exchange. MeDeHa routing tables contain forwarding information for nodes that are currently accessible. Using information from *HELLO* and *CURRENT_NEIGHBORS* messages allows nodes to maintain 2-hop routing information for other currently connected nodes. Routing information is updated after each *HELLO* notification exchange. If a node does not hear an update from a neighboring node (for which it has a routing entry) for as long as two times the period of *HELLO* exchange, it removes the routing entry from its routing table[7] and stops propagating the node's availability in subsequent *CURRENT_NEIGHBORS* notifications. All entries in the routing table for which the unavailable node is used as a gateway are also removed at this point. As soon as an entry for a node is removed from the routing table , the corresponding entry in the contact table is updated (or added) so that it can be used in the next *RECENT_NEIGHBORS* notification.

Routes are calculated in such a way that the routing loops are avoided. In this way, a direct hop to a node always has a priority over a 2-hop route to the node. Moreover, as nodes may use multiple interface identifiers (e.g., IP addresses), the routing table considers the ad-hoc interface identifier of a node as direct hop, and use all its other interfaces as accessible via the ad-hoc interface identifier of the node.

A node's contact table comprises information about other nodes that are encountered by this node over a pre-defined period of time. The contact table information is then propagated via *RECENT_NEIGHBORS* notifications. The information about a "contact" is entered into the contact table of a node when the node received a *HELLO* notification from a newly connected neighbor. This information comprises the time at which the contact occurred as well as an encounter counter. This counter is only incremented once during a contact duration (even if nodes exchange more than one *HELLO* notification), and is an indicator of the number of contact opportunities the two nodes have had with each other. Contact table entries of a node are removed when they time out. This timeout period is configurable, and depends on how long an information remains useful about a "contact" in a specific environment. A node stops propagating a contact information after this timeout.

3.3 Relay node selection and forwarding

In MeDeHa, selection of a relay node depends upon the information advertised by candidate relays (propagated as part of neighborhood information exchange) or by locally

[7] The node does not remove the neighboring node's entry from its contact table.

collecting the encounter information with other nodes. This information is used to compute the *utility* of the node as a relay. The choice of utility metrics for relay selection also depends upon the network environment, node heterogeneity, as well as application specific requirements.

For instance with IEEE 802.11, considering the case of an ESS where all APs within the ESS are connected to each other, providing an "almost connected" network, APs may have high utility as relays when compared to other nodes (see Sect. 5.2). This is because in such environments handing over a copy of a message to an AP means that the network now contains the number of copies of that application-layer message equal to the total number of neighboring APs within the ESS, even though only one AP has stored the message copy. This increases the probability of message delivery to a destination. Another advantage is that APs are expected to be more resourceful entities in terms of battery and storage space. Now consider an example where connectivity between different villages is only provided using buses that move between them. In this case, buses (or the people who ride the buses) would be given priority as relays to carry inter-village traffic (see Sect. 5.3). The affiliation to a particular community (e.g., village in this case) can also be used to choose a relay for carrying the traffic. The nodes detect the presence of these relays (such as buses) by the *utility* advertised by the relays. This information is propagated in *HELLO* messages under the field of *Node Utility*. The field *Node Utility* can also include information about the *trust rating* of the advertising node. This rating may be assigned by a central entity, and helps in avoiding malicious nodes.

Another important parameter in choosing a "suitable" relay node is the buffer capacity (e.g. in bytes) announced by a candidate relay node. If a node has more messages to send than the messages that can be accommodated by a candidate relay node, it could only forward a subset of stored message to the latter node and look for some other relay node to carry the other remaining messages. Similarly, a node's energy level is another parameter to be considered when choosing relay nodes as it may be useless to forward messages to a node who is going to die soon.

Two nodes may also exchange a summary of their stored application-layer messages (instead of the actual messages) using *MSG_VECTOR* notification as part of their *NEIGHBOR_INFO* message exchange. Also, before forwarding an application-layer message (or a set of messages) to a relay node, the corresponding route for the destination is entered in the routing table of the node that is forwarding the message with next hop set as the chosen relay. This route remains in the node's routing table until it times out.

To perform data forwarding, MeDeHa employs the hop-by-hop reliability mechanism as specified by the reference DTN architecture [11] that works as follows. When a source (or a relay) encounters a destination (or another relay) for which it carries a few messages, it forwards the messages and considers that a message is successfully received by the destination (or the relay) when it receives an acknowledgment (using TCP ack or explicit ack on top of UDP). This makes sure that the message is transfered reliably and that the number of messages transfered is proportional to the contact duration, thus avoiding any unnecessary message loss. This is even more beneficial for the scenarios where only one copy of a message exists in the network; thus, losing the only message has more drastic effect on the performance as compared to the scenario where multiple copies of a message co-exist in the network. Along with providing reliability, this mechanism also serves the purpose of controlling the number of duplicate messages flowing in the network.

3.4 Message delivery overview in MeDeHa

In this section, we present the overall mechanism of message delivery in MeDeHa by taking an example of IEEE 802.11 based networks for better understanding. Here, we consider APs as IS-based nodes, though any IS-based network can be used without the loss of generality.

A source, when having a message to send, consults its routing table to find the next-hop information for the message destination. If the information is found, it forwards the message through the specified interface. The message is stored locally on the node, if no information about the destination is found in the routing table. Nodes that are connected (associated) to an IS-based network may use as the corresponding AP as their default route, as it is assumed that the IS-based nodes such as APs are more resourceful nodes and are good candidates to store data. Moreover, as APs can be connected to each other in an ESS, storing a message at an AP increases the chances of message delivery as the message can be delivered as soon as the destination connects with any of the APs. Therefore, if a node is currently associated to an AP and has a message to send (forward), but no information about the destination is found in the node's routing table, the node should forward the message to its AP. The AP will then consult its routing table for the destination's information and if no information is available, the message will be stored locally until information about the destination is received: this information can either be that the destination is connected to the AP (directly or via an associated node), or that a connected node seems to be a better relay to carry a message to the destination. In an network where all APs are connected to each other, and there is only one copy per message, it may be better to keep the message stored at an AP and not forwarding the message from an AP to a relay, as keeping a message stored at an AP increases the chances

of message delivery, especially in a scenario where nodes are expected to be connected to the ESS at some point; the message is delivered as soon as the destination's information is found at any AP within the ESS. When more than one copy of a message exist in the network, a message carrier only forwards one copy of the message to the AP for buffering.

When a node (or a relay), carrying a message for a destination, encounters a more "suitable" relay (with the help of *RECENT_NEIGHBORS* exchange), it will add an entry in its routing table for the destination, declaring the relay as its next hop, and forwards messages for that destination to the relay. The routing table entries are refreshed periodically with the help of *CURRENT_NEIGHBORS* and *RECENT_NEIGHBORS* notifications, and all the entries for which there is no update, are removed from the routing table after a timeout. Each node maintains two types of tables: routing table and contact table. Forwarding a message to available nodes is performed by looking up the routing table entries. Contact tables are used to maintain utility function metrics for each encountered node within a specific time window. As soon as a node detects that a neighboring node has left its surrounding (i.e., if it does not hear from the latter for a period of two *HELLO* intervals), it removes the node's entry from its routing table, and updates its contact table entries for the departing station.

Advertising the addresses of all interfaces of a station in the *HELLO* notification allows message delivery to any of the available interfaces of a destination. Consider the scenario shown in Fig. 8. A source *S* with two interfaces, I.1 for infrastructure mode and A.1 for ad-hoc mode, and a destination *D* has two interface identifiers I.2 and A.2 for infrastructure and ad-hoc mode respectively. *S* is *associated* to AP *BS1* and has a message to be sent to I.2 address of *D*, but *D* is not currently *associated* to any of the APs in the network. A relay *R* meets *D* in ad-hoc mode, and is able to deliver message to *D* via *G*, because in its *HELLO* advertisement, *D* announces the possession of both I.2 and A.2, and *G* advertises to *R* that *G* is accessible. Thus, in ad-hoc mode, the message from *S* would be sent to A.2 address of *D* via *R*.

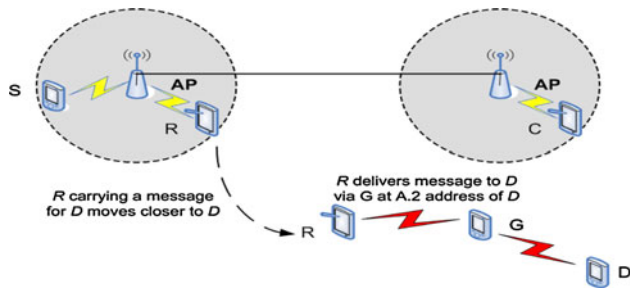

Fig. 8 An example of message delivery in heterogeneous networks

In case of cellular networks, the MeDeHa framework can be deployed within a Base Station Subsytem (BSS), which contains one Base Station Controller (BSC) and several Base Transceiver Station (BTS). The MeDeHa notification protocol can run on BTS stations such that they exchange connectivity information of mobile stations. Similarly, storage functionality can also be provided in BTS stations.

3.5 Security issues

Securing information is an important component of wireless communication. While application-layer messages can be secured using end-to-end security mechanisms such as encryption, it is also important to secure routing information exchange (e.g., *HELLO* advertisements and neighborhood information). Although we do not currently have any explicit security mechanisms in place, the MeDeHa framework can be extended to allow the integration of security-related mechanisms. For example, using message digests to ensure message integrity and authenticity (as mentioned in Sect. 3.1), adding security-specific criteria to the utility function (e.g., the "trustworthiness" of a node assigned by a trusted authority, as mentioned in Sect. 3.3).

On the other hand, it is also important to prevent unauthorized users to consume network bandwidth by sending unwanted traffic in the network. In regular networks, this can be easily managed using the principles of authentication, authorization and accounting (AAA), such that a designated node in the network is responsible for managing traffic. In networks that support intermittent connectivity of nodes, the solution is not straightforward. One solution is that each node has the public keys of all the nodes in the network [21]. But this solution is not scalable as it is difficult to configure and manage a large set of keys. If the network has a the connectivity to the backbone network, some nodes in the backbone can also be set responsible for the authorization of nodes in the network. The concept is similar to the super-node system [22]. Moreover, MeDeHa nodes can access the backbone through infrastructure-based networks to allow the implementation of a similar scheme. In the ad-hoc part, DTN-like security mechanisms [23] can be implemented where in addition to the sender authentication, each relay node also authenticates the message received from its neighboring node. For such mechanisms, key management is a critical issue [24].

4 MeDeHa's implementation

Our current implementation of MeDeHa performs message delivery in an internet comprised IS-based and ad-hoc

wireless networks where mobile nodes roam freely and may become temporarily disconnected. For the specific case of IEEE 802.11, IS-based nodes (e.g., APs) that are connected to each other form an ESS, and thus share network information using the notification protocol. While moving, nodes encounter each other in ad-hoc mode and exchange control information and data. The messages could be stored at APs as well as at relay nodes. Moreover, when a source moves and finds itself in a region of no connectivity, it starts caching its messages for the destination. In this way, the source stores messages at its end, and as soon as it finds either a destination, or a relay for the destination or an AP, it may start forwarding the messages.

The IEEE 802.11 standard does not define when and how a disassociation process should be initiated, except for the case when an authentification fails. In an ESS where AP regions do not overlap, a station would be disconnected from one AP before associating with another AP in the same ESS. Since in MeDeHa, APs need to know when a station leaves its connectivity region, we extended the IEEE 802.11 implementation to support explicit disassociations. Thus, a node keeps on checking the received power levels of beacons from its associated AP and triggers an explicit disassociation if the power level of the received beacon is less than a certain threshold. This threshold is currently set to 90% of the received power threshold of a station. To facilitate roaming, all APs that are part of the same ESS use the same channel and SSID.

Disassociation messages may get lost. This may be because the station is already out of the AP's communication range when sending this message, or because the corresponding frame collided with another frame. In this situation, the AP would still think that the station is associated, though the station has already left. This could cause data packet loss and could be minimized using an additional implicit disassociation mechanism at APs. Using implicit disassociation an AP keeps a timer running for associated stations, and in the case there is no data received from an associated station for a specific period of time, it sends a disassociation frame to the station and removed its entry from list of associated stations. It is possible that the station is still there but simply had no data to send data during that period of time. In this case, the station associates itself again with the AP.

In the current implementation of MeDeHa, we assume that the nodes identification is done using their IP addresses. In this way, nodes carrying multiple interfaces can have more than one IP address. So, a source may use one of a destination's IP addresses to communicate and the destination is able to receive messages destined to it on any IP address that it owns. We also assume that nodes in ad-hoc mode use pre-defined static IP address, and use these IP addresses to exchange control messages. These IP addresses are used in *HELLO* handshake process. Moreover, *CURRENT_NEIGHBORS* is used to exchange information about the nodes that are currently direct neighbors to the transmitting node. IP addresses of neighbors are sent in this control message, and we do not assume collision of identifiers in this case. We believe that even in an ad-hoc network where IP addresses are dynamically chosen by nodes, collisions avoidance of identifiers (IP address) can be achieved using the existing auto-configuration mechanisms for ad-hoc networks. We believe that while using IP addresses in this way has its own limitations, it is sufficient to showcase MeDeHa's functionality.[8]

We have also implemented MeDeHa on Linux as a user space daemon. Results from live experiments with MeDeHa [25, 26] corroborate the simulation results presented here.

4.1 Possible implementation approaches

The MeDeHa framework can be implemented at different layers of the communication stack. When implemented at the layer-2, it allows nodes with only two layers to be part of the MeDeHa set of devices. The advantage of this approach is that the MeDeHa's protocol could be implemented on nodes that only have two layers (e.g., AP bridges). Also, in an internet involving IS-based networks, it is easier to collect and use association or disassociation based information that is exchanged between APs. This implementation approach was chosen in our prior work [15]. The disadvantage is that message routing is more challenging.

Alternately, as we have done currently, MeDeHa can be implemented at layer-3. This facilitates the development of the routing function. On the other hand, in IS-based networks, association and disassociation information is passed to layer 3 from layer 2. But in layer-3 solution, all nodes in the network including IS-based nodes (e.g., APs) must run layer 3. An application-layer solution is also possible where application layer routing could be performed between MeDeHa nodes; in IS-based networks, the association (and disassociation) information could be passed from layer-2 to the application layer.

4.2 Implementation within ns-3 simulator

To date, network heterogeneity is not supported in most open-source network simulators. We use the ns-3 network simulator 3 (NS3) [27], which provides basic network heterogeneity support required for our framework. We had

[8] We recoginize that a longer-term naming solution is clearly needed. However, it is out of the scope of this paper and is being investigated as part of an ongoing work [28].

then to extend ns-3's heterogeneity support. As previously described, we developed explicit– and implicit disassociation mechanisms in the simulator. In explicit disassociation, 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. Whereas, in case of implicit disassociation, the AP keeps a timer for nodes associations and removes stations from its association list by sending them a disassociation frame when the timer expires. These functionalities are done at the simulator's layer-2.[9] We have also used a cross-layer information exchange to pass association or disassociation information from layer-2 to layer-3.

Buffer management policies have also been implemented to provide per-flow and per-destination priority mechanisms. For instance, when a node's buffer is full, the oldest message with lower priority is dropped. Or, if a lower priority message arrives and the node's buffer is full with higher priority messages, the incoming message is discarded (dropped).

5 Performance evaluation

We showcase MeDeHa's functionality and evaluate its performance through extensive simulations using a wide range of scenarios including traffic of different priorities. We used both synthetic traces with realistic mobility patterns as well as real mobility traces.

5.1 Performance metrics

To analyze the impact of MeDeHa on message delivery in heterogeneous internets subject to connectivity disruptions, we measure packet delivery ratio (PDR). Average delivery delay (AD) is also used as performance metric to show the benefits of embracing network heterogeneity. To this end, we compare scenarios where more than one network is supported against an infrastructure-only network [15]. The applications we considered for our framework's evaluation include file transfer and sms-like messages between nodes. Note that currently, the MeDeHa framework does not support application-layer fragmentation and reassembly. The framework only relies on the fragmentation and reassembly functionality of the network layer if available.

It is important to note that for a message delivery protocol like MeDeHa that involves wireless communication, performance of the protocol in terms of message delivery depends upon how quickly neighborhood changes are detected. *HELLO* messages are used for this purpose in ad-hoc networks, while beacons are utilized in IS-based networks. Message delivery can be improved by sending neighborhood detection messages such as *HELLO* more frequently, but on the other hand, it increases protocol overhead. This tradeoff needs to be considered when setting the protocol's parameters.

5.2 Case 1: Convention center type scenario

We consider a convention center type environment with different rooms and seminar halls where connectivity is provided by APs, but connectivity is not guaranteed everywhere (e.g., outside rooms or in hallways) in the convention center. Visitors carrying portable devices may move from one room to another and roam around across multiple AP coverage areas.[10] These APs are connected to each other via Ethernet or point-to-point links. Without MeDeHa, visitors (nodes) get disconnected temporarily while moving from one room to another and hence may loose some messages destined to them. With MeDeHa, the network stores messages temporarily. When no destination information is available, APs store messages temporarily. When using more than one network, a message can either be delivered to a destination in infrastructure mode, in ad-hoc mode, or the message can be handed over to a relay, which may carry the message to its destination.

This case is similar to the one we used in our previous work [15] in which we employed Random Waypoint (RWP) mobility model with attraction points [29, 30]. We use this scenario as a baseline to present the framework's functionality. Attraction points correspond to rooms and nodes move only in between these attraction points. Each attraction point is defined with a specific standard deviation along with an intensity to select the attraction point by the RWP mobility model. The standard deviation is of Gaussian distribution with zero mean and is used to specify the distances of nodes to the attraction point [31]. In other words, the standard deviation acts as a radius for the region of influence for an attraction point. Nodes are made to move in between these attraction point regions at a speed that is uniformly distributed between 1 and 2.5 m/s. Also, while within the coverage area of an attraction point, a node stays there for a time that is uniformly distributed between 0 and 60 s. A network of 9 APs is used spanning a 1,000 m $\times$ 1,000 m area; there are 16 attraction points, each having an effective radius of 20 m, indicating its

[9] The source code of the MeDeHa framework and our experiments can be downloaded from the URL http://planete.inria.fr/Software/MeDeHa.

[10] In our simulations, we assume that the APs have circular coverage areas. In practice, APs do not generally provide circular behavior. Changing APs coverage regions may change results obtained in this scenario, but has no effect on the functionality of MeDeHa qualitatively.

region of influence. There are 50 nodes in the network and we have run the simulations for a duration of 40 min. To perform simulations, we create some mobility traces using random waypoint mobility model with attraction points using the BonnMotion Mobility Model tool [31]. Note that all the mobility traces are taken for steady-state as the BonnMotion Mobility Model provides this feature by default. The results are obtained by running the simulation six times.

Uniform and non-uniform AP distribution: In the first set of experiments, 20 mobile stations exchange data, forming source-destination pairs. In other words, there are 20 sources and 20 destinations. Constant bit rate (CBR) traffic is generated using messages of 1 KBytes and different average data rates (in messages/mn). There is no buffer limit at APs as the goal is to study the impact of data rates and the AP distribution. For this scenario, each visitor node sends data traffic for a duration of about 20 min, and the average number of messages received by each node is represented by the average PDR for each case.

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 and uniform disconnection times when nodes move, representing an almost-connected network, comprised connectivity "black holes". The deployment of APs and that of attraction points is shown in Fig. 9.

Here, we compare two cases of MeDeHa: one where all stations support IS-based networks only (IS only), and the second case is where stations are able to connect to IS-based network as well as with other nodes in ad-hoc mode (IS + Ad-hoc). Our goal is to evaluate the impact on delivery ratio (PDR) and delivery delay (AD). In ad-hoc mode, we use the number of encounters with a station as relay selection strategy, and set its value to 2. In other words, a message is forwarded to a relay if it has seen the destination at least twice. Delivery ratio is shown in Fig. 10, while the average delivery delay is presented in Fig. 11.

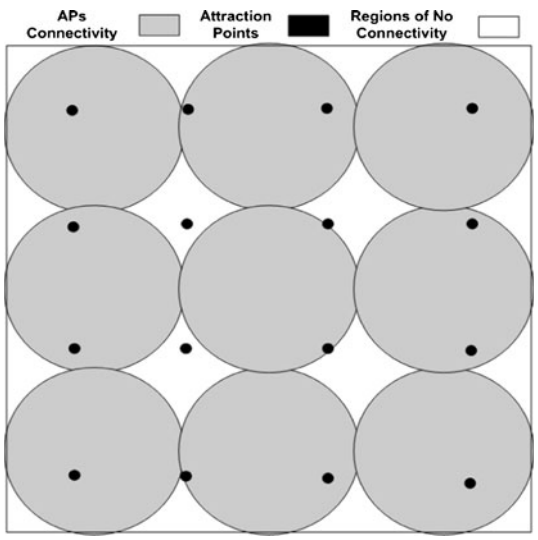

Fig. 9 Uniform deployment of APs and attraction points to have equivalent areas of no connectivity with respect to APs

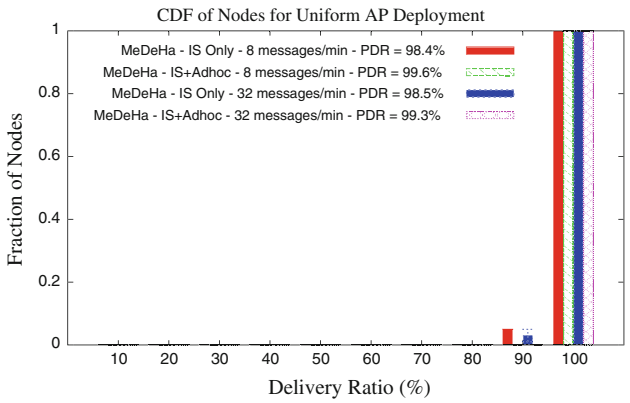

Fig. 10 Fraction of nodes versus Delivery Ratio for uniform deployment of APs

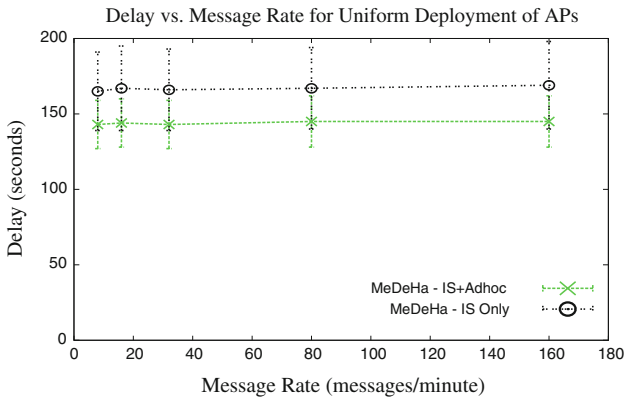

Fig. 11 Delay versus message rates for uniform deployment of APs

All stations exhibit more than 90% delivery ratio irrespective of whether they are member of one or two networks for the case of both 8 and 32 messages/mn.[11] While delivery ratio is not significantly affected, taking advantage of multiple networks decreases average delivery delay significantly irrespective of the data rate.

Next, we consider the case when the APs are distributed in the network in such a way that the distance between APs is non-uniform. The idea is to simulate an environment where the average disconnection time for stations is higher. Figure 12 shows the non-uniform deployment of APs for our simulations. All other simulation parameters are the same as the previous case. Delivery ratio and delay for the

[11] We used other values for average data rate from 1 to 160 messages/mn and observed similar performance trend.

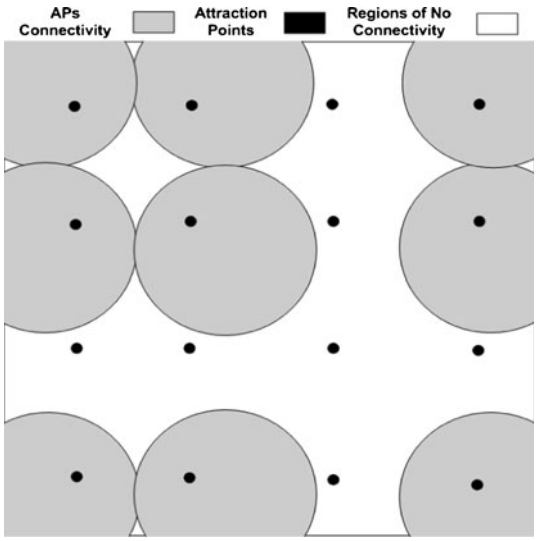

Fig. 12 Non-Uniform deployment of APs and attraction points to have non-uniform disconnection times and areas of no connectivity with respect to APs

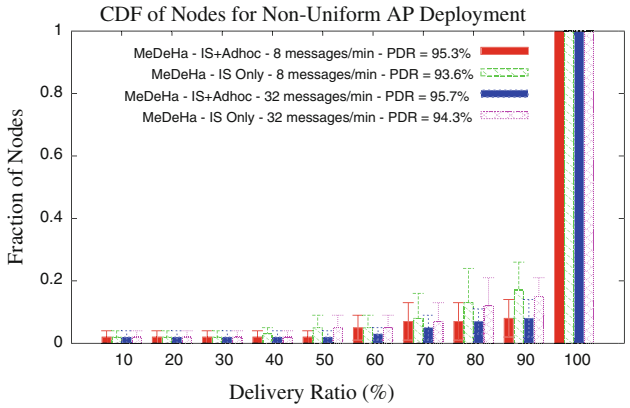

Fig. 13 Fraction of nodes versus delivery ratio for non-uniform deployment of APs

non-uniform AP deployment is shown in Figs. 13 and 14, respectively.

We observe that 80% of stations have more than 90% delivery ratio in case of stations using only Infrastructure-based networks (IS only), as compared to more than 90% of stations having more than 90% delivery ratio when stations support both IS and ad-hoc networks. Again, we can see that the average delay is higher as compared to the uniform AP deployment scenario, but we still observe an improvement in average delivery delay by using more than one network. The average delay is higher because the overall disconnection time is higher due to non-uniform AP positions, which also resulted in slightly lower PDR as compared to uniform deployment case.

Buffer Sizes: The goal of these experiments is to evaluate MeDeHa's performance when buffer capacity at

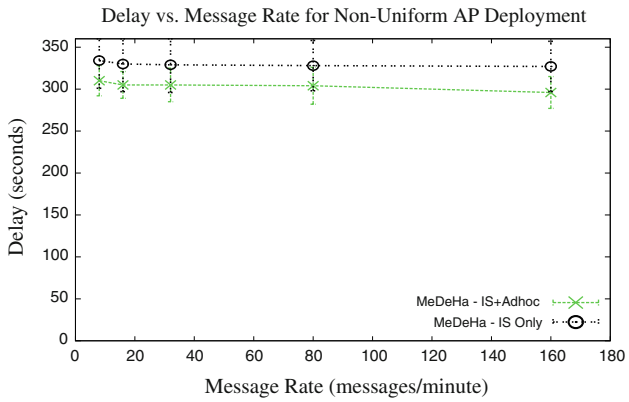

Fig. 14 Delay versus message rates for non-uniform deployment of APs

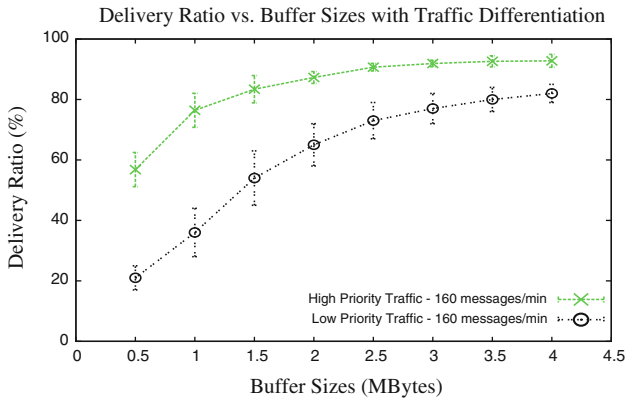

Fig. 15 Impact of varying buffer sizes on delivery ratio for high and low priority messages (message rate: 160 messages/mn)

nodes is limited. Further, we inject traffic of different priorities. We use the uniform AP deployment leaving all other parameters the same. The results are given for 160 messages/min and for stations supporting both 3 and ad-hoc networks. Delivery ratio for different buffer sizes and 2 traffic priorities (high and low) is shown in Fig. 15.

Our results confirm that MeDeHa gives preference to high priority messages, i.e., they achieve higher delivery ratio as compared to lower priority messages; this is especially true for the cases where buffer capacity is more limited.

5.3 Case 2: Communication between clusters of nodes

In this scenario, we simulate three clusters, each of which equipped with 3 APs connected to one other as part of an ESS. As shown in Fig. 16, within each cluster, there maybe some regions with no connectivity . Each cluster spans an area of 400 m $\times$ 400 m each and are placed well apart so they don't have overlapping coverage areas, i.e., they are disconnected from each other. Each cluster is configured

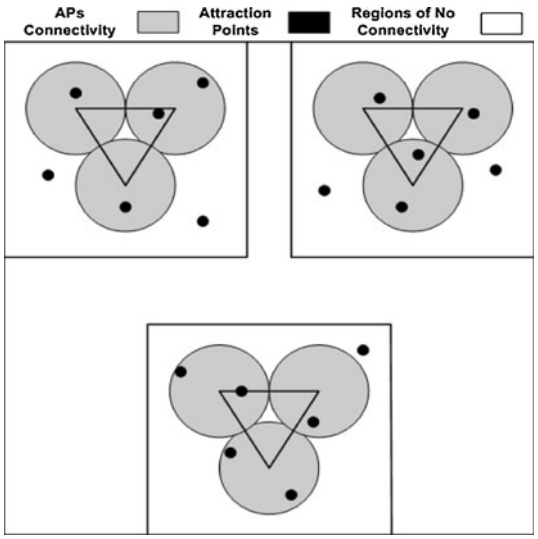

Fig. 16 Deployment of APs and attraction points in a scenario with 3 disconnected clusters

with 20 users carrying mobile devices: 14 of which only move within the boundary of their cluster at pedestrian speeds, while 6 visit other clusters with probability 0.4. These nodes are potential relays to carry and forward inter-cluster traffic and are assumed to move at vehicle speeds uniformly distributed between 30 and 60 km/h.[12] Pedestrians that move inside a cluster do so at speeds uniformly distributed between 3 and 6 km/h. Total simulation area is 3 km $\times$ 3 km, and total simulation time is 120 min. The performance metrics used are percentage of nodes that receive a certain delivery ratio, average packet delivery ratio, and average delivery delay. Figure 16 shows the map of the scenario and the corresponding AP locations.

Forwarding versus replication: For this scenario, we have chosen "community affiliation" as the relay selection strategy, where a community corresponds to a cluster. In other words, a source (or a relay) forwards a message to another node if the latter belongs to the same cluster as that of the destination. Here, we compare the behavior of forwarding (where there is only one copy of a message) with replication (where multiple copies per message exist in the network). For this scenario, we used 2 copies per message for the replication.

Additionally, traffic is divided into two parts: intra-cluster and inter-cluster traffic. Intra-cluster traffic corresponds to the case where both the source and the destination belong to the same cluster and thus both do not leave the cluster for the duration of simulation. Ten sources are chosen across all clusters to generate intra-cluster traffic, which is destined to nodes in their own cluster (more precisely 4 in cluster 1, 3 each in cluster 2 and 3). Inter-cluster traffic represents the traffic exchanged by nodes belonging to different clusters. For this traffic, 10 source-destination pairs are selected from all 3 clusters such that both the source and the destination do not move out of their clusters and belong to different clusters. The average message rate is 3 messages/mn and users send messages to other users for a duration of around 80 min. Figure 17 shows the CDF of the fraction of nodes as a function of delivery ratio using forwarding and replication for both kinds of traffic. The average number of messages received by each user is represented by the average PDR indicated in Fig. 17.

By comparing the results of forwarding and replication, we can see that in the case of forwarding, 33% of the nodes have less than 80% delivery ratio, whereas using 2-copy replication, only 20% of nodes have less than 80% delivery ratio, which is a significant improvement. A slight improvement is observed in the average PDR in the case of intra-cluster traffic. This slight improvement occurs because the traffic is local and any local node can become a relay node for a message, so the probability of message delivery is high. Hence, increasing number of copies from 1 to 2 does not help much as forwarding performs quite well, mainly because the nodes tend to see each other more, and the messages are also stored at the local APs. The minor increase in average delivery delay (AD) is due to the increase in PDR from 97.0 to 99.4%. For inter-cluster traffic, average PDR is greatly improved by using 2-copy *replication* as compared to forwarding (from 84 to 92%), but this increases the average delay as well (from 1259 to 1274 s). The increase in average delay (AD) is due to the

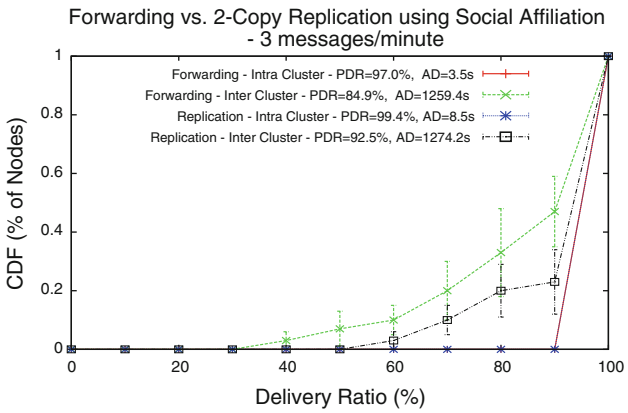

Fig. 17 CDF of fraction of nodes versus delivery ratio showing the comparison between forwarding and 2-copy replication for inter-cluster and intra-cluster traffic. Message rate is set to 3 messages/mn

[12] For this scenario, we assume that, while moving, users have their devices *on*. In real scenarios, users may turn their devices (e.g.,laptops) *off* while moving. For such cases, message buffering in the nodes must use persistent storage. When considering devices that can be turned off, there will be less opportunities for message forwarding due to less number of relay nodes. This may reduce the PDR and increase the AD.

significant improvement in PDR, as the messages that get delivered very late contribute towards increase in AD. These messages do not contribute in *forwarding* case as they are never delivered. The results are obtained by running the simulation experiments six times.

Relay selection strategy: Selecting a "suitable" relay to carry messages is an important component of MeDeHa and can have considerable effect on the performance of the protocol. One can employ different relay selection strategies depending upon a number of factors including network-, node-, and application characteristics as described in Sect. 2. We focus our attention on evaluating different criteria as the *utility* of a node to become a relay. First, we show the impact of using "number of past encounters" , which we refer as Encounter-Based Replication (ER).[13] Through ER, a source (or a relay) hands over a message to a node only if the latter has already encountered a destination at least twice, and it has seen the destination more recently. The idea behind this utility metric is that if a node has already seen a destination at least twice, there is a strong probability that it will again encounter the destination in the future. Depending upon the mobility pattern of nodes, this utility function may not be a good indication of the likelihood of future encounters.

Next, we choose "community affiliation" as the utility function for relay selection. In this way, a relay is chosen only if it belongs to the community of a destination. This utility function is meaningful here since to send traffic between different clusters, we have to rely on nodes that visit different clusters. Thus it is useful to forward a message to a visiting node for a destination if both destination and visiting node belong to the same cluster. We call this relay selection strategy as Social Affiliation-based Replication (SAR) scheme. We also combine the above two utility functions into the strategy which we refer as the Encounter and Social Affiliation-based Replication (ESAR). A relay is chosen to carry a message to a destination only if it belongs to the same community as that of the destination and if it has encountered the destination at least twice. A comparison is shown between ER, SAR and ESAR selection strategies for 2-copy replication is illustrated in Fig. 18. All other simulation parameters are the same as presented in Sect. 5.3.

Figure 18 further illustrates that for inter-cluster traffic, social affiliation-based replication (SAR) performs the best both in terms of average delivery ratio (PDR) and average delivery delay (AD). The reason is that the clusters are far away from each other and are not connected. Hence for message delivery, we rely only on the nodes that move between different clusters. SAR obtains the best results in this scenario because handing over a message to a node that belongs to the same cluster as that of destination increases the chances of message delivery, as compared to ER case which relies on the fact that the relay has to meet at least a few number of times (2 encounters in this case) before becoming a candidate for relay selection. Considering the size of the network and the nodes' speed, it is unlikely that nodes in different clusters tend to encounter each other too often. For the same reason, ESAR performs the worst, as the criteria for the relay selection is stricter in ESAR (hand over a message to a relay if the relay belongs to the same cluster as that of the destination and if the relay has seen the destination at least twice). This selection criteria adds the buffering delay for waiting for a suitable node and results in expiration of a lot of messages while being stored at nodes. For the messages that are delivered, ESAR yields the highest delay. Therefore, increasing the simulation time would not have helped improve PDR in this case because the messages are expired while stored at the nodes. Increasing the simulation time can improve the results only when message expiry time is also increased.

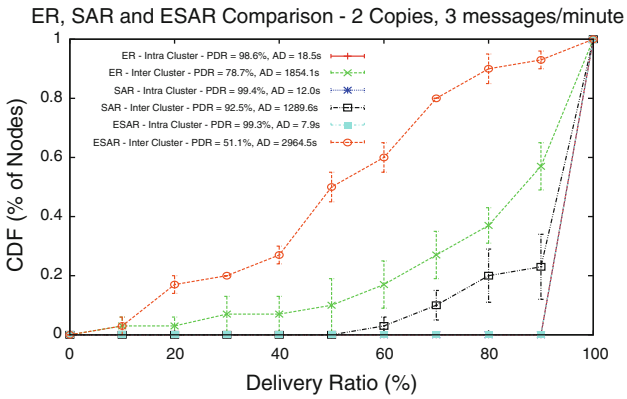

Fig. 18 CDF of nodes versus delivery ratio for 2-copy encounter replication (ER), social affiliation replication (SAR) and encounter and social affiliation-based replication schemes—(3 messages/mn)

On the other hand, for intra-cluster traffic, all relay selection strategies yield similar average delivery ratio, though ESAR performs slightly better than the other two strategies in terms of average delay. When both source and destinations are within the same cluster and do not move out, nodes tend to encounter other nodes more often; hence, ESAR yields the most accurate relay selection (it does not hand over a message to a node that belongs to a different cluster even if the node has already encountered the destination twice). This results in minimizing end-to-end delay as messages reach the destination in an efficient way.

When comparing the two traffic types, intra-cluster traffic has better PDR values with significantly low delay

[13] There are also some other relay selection strategies available such as EASE [17], EBR [18] and Prophet[19]. Here, we use simple strategies as the purpose is to show the validation of the framework functionality. Of course, using more sophisticated strategies may provide better results.

values, as both the source and the destination are present in the same cluster. On the other hand, PDR of inter-cluster traffic is relatively low and it has very high delivery delays, as the clusters are not directly connected and nodes has to carry the inter-cluster traffic for long periods of time before delivering them to the destinations.

We have also conducted experiments with a scenario that spans in an area of 1 km × 1 km involving 3 communities of 400 m × 400 m each; thus, the nodes encounter each other more. In that case, we observed that ESAR performs the best in terms of average PDR and the worst in terms of AD, while SAR performs better in terms of AD and has the lowest values of average PDR attained. We are omitting these results for space considerations. Thus, we can conclude that using communities, SAR performs better when nodes encounters are infrequent, while ESAR has better results when nodes encounter each other more often. In all the experiments, ESAR has obtains the worst AD.

5.4 Case 3: KAIST campus traces

We used a subset of real traces for the KAIST Campus available from CRAWDAD [32]. These traces record mobility of 50 students via their locations during a day. We took a 2 h window over the trace from 10 AM to 12 PM. We superimpose this mobility pattern on top of an area of 1.4 km × 2.4 km with 9 APs. All APs are connected to each other as in Case 1. Students visit different places of campus during the time and their speed change (students take shuttles while moving from one place to another, and move at pedestrian speed or not at all). Again, we evaluated this scenario for 20 source-destination pairs of students, sending each other messages at the average rate of 3 messages/mn, and obtained the CDF of nodes (students) attaining a particular delivery ratio (PDR) for the cases (1) where students can only connect to IS-based network (MeDeHa-IS only), (2) where students can use both IS-based and ad-hoc interfaces to communicate (MeDeHa-IS + Ad hoc) using both forwarding (1-copy per message) and replication (2-copy per message). We also observed average packet delivery ratio and average delay (AD). The result is shown in Fig. 19. Here, we used Encounter-based Replication (ER) for relay selection, and set the number of encounters value to two for students to be chosen as relays. In this scenario, each student sent messages for a duration of 40 min to the other student (destination), and the average number of messages received by each student is represented by average PDR achieved for each case.

Figure 19 illustrates that using network heterogeneity (IS + Ad hoc) improves the performance both in terms of delivery ratio (PDR) and delivery delay (AD). IS + Ad hoc *replication* reaches the best average PDR and AD values. In terms of fraction of nodes, we can see that only 6% of nodes have less than 90% delivery ratio for 2-copy heterogeneous network (IS + Ad hoc) as compared to 25% of nodes having less than 90% of delivery ratio when using only IS-based network (IS only).

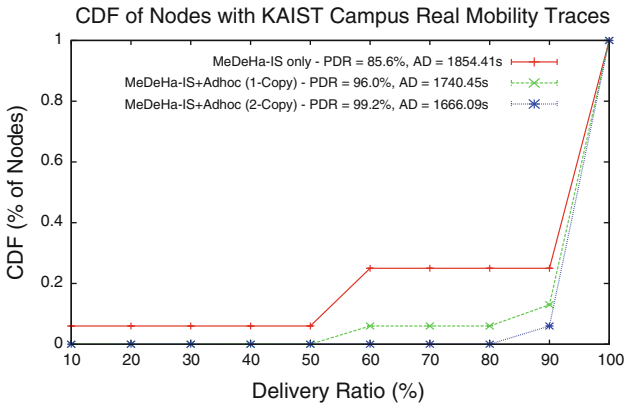

Fig. 19 CDF of nodes versus Delivery ratio for KAIST campus traces for two hours using IS only and IS+Ad hoc modes (message rate: 3 messages/mn)

6 Related work

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

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 generally performed above the transport layer to provide interoperability among networks that support different types of transport layers. The Bundle Protocol is intended to be compatible with different types of networks through the convergence layer

adapters. In this way, the protocol supports internetworking by allowing multiple convergence layers to be used for different networks. MeDeHa is orthogonal to the Bundle architecture and can be used with in cooperation with it. In such cases, it is useless to store data at lower layers 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 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 with the help of nodes that move in between these islands. Some studies use the concept of node relaying 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 in terms of storage space and battery power. The concept is very beneficial in increasing the delivery ratio, and in reducing the overall delay. But it is not trivial to find the optimal number of these special-purpose nodes in the network, and to find their routes.

Some initiatives target relay node selection in a disruption tolerant environment. One notable example is to use different utility functions for intermittent connected networks with different characteristics [16]. In Exponential Age Search (EASE) algorithm [17], a destination location is estimated by using the encounter database maintained locally by each node for every other node. EBR [18] is a similar approach where future rate of node encounters is predicted using number of past encounters with nodes. For this purpose, an encounter metric is computed locally by each node, and is used as utility metric when choosing a relay for a message.

7 Conclusion

This paper introduced MeDeHa, a robust and flexible message delivery framework targeting heterogeneous networks subject to intermittent connectivity. We believe that this work is an important building block to enable current and upcoming applications since; (1) future internets will likely become increasingly more heterogeneous and (2) in many scenarios/applications, late delivery is preferred over loss of data.

MeDeHa's contributions are two fold: the framework bridges the connectivity gap between infrastructure-based and infrastructure less networks, and it addresses the problem of frequent and/or long-lived connectivity disruptions in such heterogeneous networked environments. We showcase MeDeHa's features and performance in simulations using a variety of realistic scenarios, including user mobility traces. Finally, we show that MeDeHa helps to significantly improve message delivery ratio, while reducing delivery delay.

References

1. Sarafijanovic-Djukic, N., Piorkowski, M., & Grossglauser, M. (2006). Island hopping: Efficient mobility-assisted forwarding in partitioned networks. In *Proceedings of IEEE SECON'06 (pp. 226–235)*.
2. Ott, J., Kutscher, D., & Dwertmann, C. (2006). Integrating DTN and MANET routing, In *Proceedings of ACM SIGCOMM Workshop on Challenged Networks (CHANTS)* (pp. 221–228). Pisa, Italy.
3. Vahdat, A., & Becker, D. (2000). *Epidemic routing for partially connected ad-hoc networks*, Technical Report CS-200006, Duke University.
4. Spyropoulos, T., Psounis, K., & Ragha Vendra, C. S. (2005). Spray and wait: An efficient routing scheme for intermittently connected mobile networks, In *Proceedings of ACM SIGCOMM Workshops WDTN* (pp. 252–259). Philadelphia, PA.
5. Chen, J.-C., Li, S., Chan, S.-H., & He, J.-Y. (2005). WIANI: Wireless infrastructure and ad-hoc network integration, In *Proceedings of IEEE International Conference on Communications (ICC)* (pp. 3623–3627). Seoul, Korea.
6. He, J., Chen, J., Chan, S.-H. G., & Liew, S.-C. (2003). Mixed-mode Wlan: The Integration of ad-hoc mode with wireless LAN infrastructure. In *Proceedings of IEEE Globecom* (pp. 231–235).
7. Parata, C., Convertino, G., Scarpa, V. (2007). Flex-WiFi: A mixed infrastructure and ad-hoc IEEE 802.11 network for data traffic in a home environment. In *Proceedings of the First IEEE WoWMoM Workshop on Autonomic and Opportunistic Communications (AOC)* (pp. 1–6). Finland.
8. Chandra, R., Bahl, P., & Bahl, P. (2004). MultiNet: connecting to multiple IEEE 802.11 networks Using a singleWireless card. In *Proceedings of IEEE Infocom* (pp. 882–893). Hong Kong.
9. Hamidian, A., Korner, U., & Nilsson, A. (2005). *Performance of internet access solutions in mobile ad-hoc networks*. Wireless Systems and Mobility in Next Generation Internet (pp. 189–201). Berlin: Springer.
10. Scott, K., & Burleigh, S. (2007). *RFC 5050, Bundle Protocol Specifications*, IRTF DTN Research Group, November.
11. Cerf, V., Burleigh, S., Hooke, A., Torgerson, L., Durst, R., Scott, K., Fall, K., & Weiss, H. (2007). *RFC 4838, Delay-Tolerant Networking Architecture*, IRTF DTN Research Group, April.
12. Zhao, W., Ammar, M., & Zegura, E. (2004). A message ferrying approach for data delivery in sparse mobile ad-hoc networks. In *Proceedings of ACM/IEEE MOBIHOC* (pp. 187–198). Japan.
13. Shah, R., Roy, S., Jain, S., & Brunette, W. (2003). Data MULEs: Modeling a three-tier architecture for sparse sensor networks. In *Proceedings of IEEE Workshop on Sensor Network Protocols and Applications (SNPA)* (pp. 30–41). Seattle, WA.
14. Zhao, W., Chen, Y., Ammar, M., Corner, M., Levine, B. N., & Zegura, E. (2006). Capacity enhancement using throwboxes in DTNs. In *Proceedings of IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)* (pp. 31–40). Canada.

15. Rais, R. N. B., Turletti, T., & Obraczka, K. (2008). Coping with episodic connectivity in heterogeneous networks In *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. pp. 211–219. Canada.
16. Spyropoulos, T., Turletti, T., & Obraczka, K. (2009). Routing in delay tolerant networks comprising heterogeneous node populations. *IEEE Transactions on Mobile Computing (TMC), 8*(8), 1132–1147.
17. Grossglauser, M., & Vetterli, M. (2006). Locating mobile nodes with EASE: Learning efficient routes from encounter histories alone. *IEEE/ACM Transactions on Networking, 14*(3), 457–469.
18. Nelson, S., Bakht, M., & Kravets, R. (2009). Encounter-based routing in DTNs. In *Proceedings of IEEE Infocom'09*, Brazil.
19. Lindgren, A., Doria, A., & Scheln, O. (2004). *Probabilistic routing in intermittently connected networks*, Lecture Notes in Computer Science, Vol. 3126, pp. 239–254.
20. Krifa, A., Barakat, C., & Spyropoulos, T. (2008). Optimal buffer management policy for delay tolerant networks. In *Proceedings of SECON, San Francisco, June, CA*.
21. Raya, M., & Hubaux, J. P. (2007). Securing vehicular ad hoc networks. *In Proceedings of Journal of Computer Security 15*(1), 39–68.
22. Samuel, H., & Zhuang, W. (2009). Preventing unauthorized messages in DTN based mobile ad hoc networks. In *Proceedings of IEEE Globecom* (pp. 1–6).
23. Farrell, S., Symington, S. F., Weiss, H., & Lovell, P. (2009). *Delay-tolerant networking security overview*, Internet Draft, IRTF, September.
24. Bindra, H. S., & Sangal, A. L. (2010). Considerations and open issues in delay tolerant networks (DTN) security. In *Proceedings of Wireless Sensor Network 2*(8), 645–648.
25. Mendonca, M., Rais, R. N. B., Turletti, T., & Obraczka, K. (2010). *Message delivery in heterogeneous disruption-prone networks*, demo presentation in ACM Mobicom.
26. Rais, R. N. B., Mendonca, M., Turletti, T., & Obraczka, K. (2011). Towards truly heterogeneous networks: Bridging infrastructure-based and infrastructure-less networks. In *Proceedings of 3rd IEEE/ACM International Conference on Communication Systems and Networks (COMSNETS)* (pp. 1–10). India.
27. Network Simulator and Network Animator Project (NSNAM), *Network Simulator 3 (ns-3)*, http://www.nsnam.org.
28. Rais, R. N. B., Abdelmoula, M., Turletti, T., & Obraczka, K. (2011). Naming for heterogeneous networks prone to episodic connectivity. In *Proceedings of IEEE WCNC*, Mexico.
29. Bettstetter, C., & Wagner, C. (2002). 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*, Vol. 11, pp. 41–58.
30. Feeley, M., Hutchinson, N., & Ray, S. (2004). *Realistic mobility for mobile ad-hoc network simulation*, Ad-Hoc, Mobile, and Wireless Networks, Lecture Notes in Computer Science, Vol. 3158. Berlin: Springer.
31. BonnMotion, Univerysity of Bonn, *A mobility scenario generation and analysis tool*, http://web.informatik.uni-bonn.de/IV/Mitarbeiter/dewaal/BonnMotion.
32. Rhee, I., Shin, M., Hong, S., Lee, K., Kim, S., & Chong, S. (2009). *CRAWDAD data set ncsu/mobilitymodels (v. 2009-07-23)*, Downloaded from http://crawdad.cs.dartmouth.edu/ncsu/mobilitymodels, July.

Author Biographies

Rao Naveed Bin Rais received his M.S. and Ph.D. degrees in Computer Engineering from University of Nice, Sophia Antipolis, France in 2007 and 2011 respectively. Before that, he received his B.E. in Computer Systems from National University of Sciences and Technology (NUST), Pakistan in 2002. He is currently working as Assistant Professor in the Electrical Engineering Department at COMSATS Institute of Information Technology (CIIT), Lahore Pakistan. Before his M.S. in France, he has four years of industrial experience in Pakistan. He has done his Ph.D. research at INRIA, Sophia Antipolis, France under the supervision of Dr. Thierry Turletti and Prof. Katia Obraczka and worked on adaptive communication mechanisms for networks with intermittent connectivity. His research interests include future internet architecture, and network protocol design for heterogeneous networks and mobile ad hoc networks (MANETs).

Thierry Turletti received the M.S. (1990) and the Ph.D. (1995) degrees in computer science both from the University of Nice—Sophia Antipolis, France. He has done his PhD studies in the RODEO group at INRIA Sophia Antipolis. During the year 1995–1996, he was a postdoctoral fellow in the Telemedia, Networks and Systems group at LCS, MIT. He is currently a senior research scientist at the Planète group at INRIA Sophia Antipolis. His research interests include multimedia pplications, congestion control and wireless networking. Dr. Turletti serves on the Editorial Board of the Wireless Communications and Mobile Computing (WCMC), Wireless Networks (WINET) and Advance on Multimedia (AM) journals.

Katia Obraczka received the B.S. and M.S. degrees in electrical and computer engineering from the Federal University of Rio de Janeiro, Brazil, and the M.S. and Ph.D. degrees in computer science from the University of Southern California (USC). She is currently Professor of Computer Engineering at the University of California, Santa Cruz. Before joining UCSC, she held a research scientist position at USC's Information Sciences Institute and a research faculty appointment at USC's Computer Science Department. Her research interests include computer

networks, more pecifically, network protocol design and evaluation in wireline as well as wireless networks, distributed systems, and Internet information systems. She has been a PI and a co-PI in a number of projects sponsored by government agencies (NSF, DARPA, NASA) as well as industry. Dr. Obraczka has authored over 100 technical papers in journals and conferences.