HAL open science

# Adaptive Forwarding to Match Mobility Characteristics in Delay Tolerant Networks

Cigdem Sengul, Aline Carneiro Viana, Roy Friedman, Marin Bertier, Anne-Marie Kermarrec

## To cite this version:

Cigdem Sengul, Aline Carneiro Viana, Roy Friedman, Marin Bertier, Anne-Marie Kermarrec. Adaptive Forwarding to Match Mobility Characteristics in Delay Tolerant Networks. [Research Report] RR-6816, INRIA. 2009, pp.24. inria-00356601v2

**HAL Id: inria-00356601**

**https://inria.hal.science/inria-00356601v2**

Submitted on 24 May 2009

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

# *Adaptive Forwarding to Match Mobility Characteristics in Delay Tolerant Networks*

Cigdem Sengul — Aline Carneiro Viana — Roy Friedman — Marin Bertier — Anne-Marie Kermarrec

**N° 6816**

May 2009

Thème COM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--6816--FR+ENG

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche SACLAY - ÎLE-DE-FRANCE

# Adaptive Forwarding to Match Mobility Characteristics in Delay Tolerant Networks

Cigdem Sengul* , Aline Carneiro Viana , Roy Friedman† , Marin Bertier , Anne-Marie Kermarrec

Thème COM — Systèmes communicants
Équipes-Projets Asap

Rapport de recherche n° 6816 — May 2009 — 20 pages

**Abstract:** In this paper, we propose an adaptive and opportunistic forwarding mechanism that is able to match mobility characteristics in Delay Tolerant Networks (DTNs). Our protocol, Seeker, empowers nodes with the ability to estimate future contact opportunities based on a history of pair-wise contacts. Furthermore, Seeker is able to adjust and rewind forwarding and buffering decisions on the fly. This ensures a good trade-off between reliability and resource-efficiency, even when disconnections are frequent and hard to predict. Essentially, the novelties of Seeker are (1) its ability to estimate good relays without having a global network view and (2) its flexibility to operate under different conditions. These features are particularly important as in DTNs, devices are further restricted by strict resource constraints and a contemporaneous path may never exist between two nodes in the network. Using simulations based on a synthetic mobility model and real mobility traces, we show that Seeker is able to adapt its forwarding accordingly in diverse scenarios and achieves high performance with low overhead.

**Key-words:** opportunistic forwarding, buffer management, contact prediction, contact history maintenance, delay tolerant networks

* TU-Berlin.
† Technion.

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université
4, rue Jacques Monod, 91893 ORSAY Cedex
Téléphone : +33 1 72 92 59 00

# Acheminement Adaptive aux Caractéristiques de Mobilité dans des Réseaux Tolérants aux Délais

**Résumé :**

Ce travail porte sur la proposition d'un mécanisme d'acheminement adaptive et opportuniste, capable de correspondre aux caractéristiques de mobilité dans les réseaux tolérants aux délais (DTN). Notre proposition, Seeker, permet aux nIJuds d'estimer les possibilités futures de contact en se basant sur une histoire de paires de contacts. En outre, Seeker, est capable de s'adapter et de réévaluer à la volée les décisions d'acheminement et de stockage. Cela assure un bon compromis entre la fiabilité et l'efficacité des ressources, même lorsque les déconnexions sont fréquentes et difficiles à prévoir. Essentiellement, les nouveautés de Seeker sont (1) sa capacité d'estimer la bonne relais sans avoir une vue globale du réseau et (2) sa flexibilité pour fonctionner dans des conditions variées. Ces fonctionnalités sont particulièrement importantes dans DTNs, où les appareils ont des ressources limités et où un chemin peut ne jamais exister entre deux noeuds du réseau. En utilisant des simulations basées sur des modèles synthétiques de mobilité et des traces réelles de mobilité, nous montrons que Seeker est en mesure d'adapter conformément son acheminement dans des différents scénarios et possède une performance élevée en générant de faible surcharge.

**Mots-clés :** routage opportuniste, acheminement adaptive, mobilité, réseaux tolérants aux délais, estimation de contactes

# 1 Introduction

*Delay Tolerant Networks* (DTNs) are wireless networks in which a contemporaneous routing path may never exist between a sender and a receiver [12]. Examples of DTNs include disaster response, underwater sensor, vehicular networks, and pocket-switched networks, which provide connectivity to users that carry their portable devices from one connectivity island to another. However, due to mobility of nodes in these networks, *store-and-forward* techniques can ensure eventual communication between any two nodes. Additionally, DTNs are frequently characterized by strict resource constraints in terms of memory, CPU and energy. Given these characteristics, any forwarding mechanism for DTNs must be frugal in its usage of computation and communication resources.

Note that an immediate solution that comes to mind for data forwarding in DTNs is to always relay through the node that is likely to meet the destination first. However, some nodes may never meet the destination, yet can still be good relays since they frequently meet other nodes that often meet the destination. Thus, it makes sense to evaluate the contact histories of all pairs of nodes. However, keeping track of all this information and propagating it to every node is too costly. Therefore, the main challenge becomes *to adjust forwarding decisions on the fly and recover from bad decisions to perform under different mobility conditions with a low cost*. This is essential to ensure reachability even when disconnections are frequent and hard to predict. Hence, current research on DTNs focuses on understanding mobility patterns, especially, human mobility and social network characteristics [5, 6, 4]. These efforts are motivated by the inefficiency or unsuitability of the existing protocols (e.g., MANET routing protocols) to DTNs. Recently, routing using knowledge of underlying mobility patterns is explored in [15, 11, 25, 1]. However, these approaches either do not consider any resource constraints or present low adaptation capabilities.

Adjusting forwarding decisions on the fly while respecting resource constraints is thus an interesting challenge in DTNs and is the main focus of our research. To counter this challenge, Seeker, empowers nodes with the ability to make contact predictions that rely on local information (e.g., wireless connectivity and neighborhood variations). Hence, the novelty of Seeker is *its ability to estimate good relays without having a global view.* Nodes use simple online predictors, which estimate next contact opportunities based on a history of pair-wise contacts and their current context. This history includes both the quality of the connection between the nodes, whenever they meet, as well as the likelihood of successfully forwarding messages towards the destination. *Nodes then make on the fly forwarding and data buffering decisions* based on these estimations. Hence, Seeker takes advantage of the bias among different contacts [26] for more reliable communication. This behavior is especially advantageous, when the network is stable and the mobility patterns are highly predictable.

In addition, although a large amount of effort has been invested in designing routing algorithms for DTNs [15, 11, 23], the effect of buffer and bandwidth constraints in these algorithms has not received similar attention. Contrarily, Seeker respects these constraints and *does not assume infinite buffers or bandwidth*. Using contact-history-based estimations, Seeker propagates messages in a headlight fashion towards the destinations and so, maintains a reasonable memory, computation, and messaging overhead.

Finally, our protocol also considers that contact histories might not always be up-to-date. Seeker drives opportunistic operation by taking advantage of new meetings and hence is not too tied to predictions. Nodes are able to *rewind forwarding decisions* and thus, save messages, based on local observations and through well-designed buffer

management mechanisms. We assert this flexibility is the key to the *adaptive operation of Seeker to different mobility conditions*.

We study the performance of Seeker using simulations based on a synthetic mobility model and real mobility traces. The simulation results show that even if a simple prediction technique is used, Seeker is able to make good enough decisions to guarantee high reliability and acceptable delay in significantly different environments in terms of mobility. This flexibility to operate under different conditions renders Seeker invaluable for DTNs. Specifically, Seeker, with its adaptive behavior, *is able to imitate the performance of best-performing option* for each mobility scenario: maintains low overhead in static scenarios and high reliability in environments where connection opportunities are low. Finally, Seeker achieves these results with low control, buffering and forwarding overhead.

The remainder of this paper is structured as follows. In Section 2, we present the motivation for adaptive forwarding for DTNs and explain the design of Seeker. Sections 3, 4, 5 and 6 describe the main components of our protocol. Section 7 presents our evaluation study. Section 8 discusses the related work. Section 9 concludes with future work.

## 2 Emphasizing flexibility

The main goal of our research is to provide adaptive forwarding for matching mobility characteristics in DTNs. Additionally, when there are no disruptions due to mobility, it should be possible to take advantage of this and incur lower costs for higher reliability. To meet these challenges, we propose Seeker, which empowers nodes with the capability to learn and adapt based on past observations. Essentially, if a pattern exists among past contacts, future contacts can be estimated. Such estimations would be invaluable for managing message buffering and transmissions. On the flip side, however, when predictions do not match future behavior, it is necessary to rewind bad forwarding decisions to improve performance. Therefore, in the Seeker design, we emphasize flexibility to adapt to different situations to achieve a good trade-off between performance and cost.

We assume that a network may exhibit different levels of mobility, including being disconnected at certain times. Nodes, including destinations, might be static or mobile. All nodes are identical and have limited resources. Each node is able to communicate with a subset of neighbors that are in its transmission range. We do not assume symmetric communication. Nodes do not know their location or any topological information, such as where the destinations are. Given this model, our objective is to design a protocol that propagates messages to destinations in a headlight manner, thereby achieving high reliability and timely delivery with an acceptable overhead.

Clearly, these goals can be contradicting. The simplest way to achieve high reliability and timely delivery is epidemic routing [29]. However, the good performance also comes with a high cost. Furthermore, if buffer and bandwidth constraints are taken into account, this performance might not be realized in practice. For instance, in a highly dense network with low mobility, epidemic routing might incur too much overhead, degrading also reliability (see Section 7). Current approaches for reducing this overhead require network topology awareness [23] or highly predictable contact patterns [13]. Other approaches try to reduce the cost by selecting a few but good relays, by comparing the last meeting time with the destination or contact frequency, among other metrics [11]. However, this might lead to being too conservative and losing good for-

warding opportunities. To strike a balance, a protocol must be able to adapt to changes in the the node availability. To this end, Seeker enables nodes to learn their contact potential and base their forwarding decisions on this knowledge.

Intuitively, Seeker operates as follows. When node $j$ receives messages from node $i$ to deliver to a destination $m$, $j$ takes charge of these messages probabilistically based on the expected connectivity to and connection quality with $m$. In other words, the more $j$ believes it is on a good path leading to $m$, the more likely $j$ is to carry and forward the message for $i$. This way, messages can be pulled towards destinations through higher quality paths. Paths with low connection quality or lower chance to meet the destination are weeded out since the nodes on these paths are gradually attached a low forwarding probability. Of course, in a generic DTN, connectivity and connection quality is not known ahead. Hence, Seeker needs to make estimations and be able to recover from bad decisions. This is achieved through four mechanisms:

- Contact history maintenance
- Contact prediction
- Opportunistic forwarding
- Buffer management

In Seeker, each node maintains a *contact history*. Based on this history, nodes make simple predictions about future meetings with their contacts. These *predictions* drive the *opportunistic forwarding mechanism*. Essentially forwarding decisions are left to the receivers of a message. Depending on their path quality to the destination of a message, nodes only forward the message at a time when they expect to meet their best contact for this job. However, when this time comes, nodes still locally broadcast the message to take advantage of other nodes in the vicinity. Finally, *buffer management* allows executing and rewinding the forwarding decisions. We present the details of these mechanisms in Sections 3 to 6.

# 3 Maintaining Contact History

In Seeker, a node builds a contact history as it meets other nodes in the network. By not limiting the contact information to destinations, nodes are able to exploit other communication opportunities through multi-hop paths. This information is essential for nodes that never come into contact with a destination [20, 26]. Next, we present how nodes detect connections and disconnections to other nodes.

## 3.1 Detecting connection

A node discovers it has come into contact with another node if it receives a message from the node. Since we do not assume symmetric communication, the reception of a message does not guarantee a bidirectional link, but a potential contact. Seeker utilizes the following messages:

- *Data:* The message to be sent to the destination.
- *Ack:* Sent by a *destination* for a received data message to the previous hop.
- *Promise:* Sent by a *non-destination* node to the previous hop, if it decides to forward the data message.

- *Hello:* Sent by all nodes to announce their presence.

*Hello* messages are sent periodically. Nevertheless, to reduce the control overhead, a node defers a *hello* message if it already sent other messages during the last period. In addition to well-used Hello messages for neighbor discovery in DTNs, we introduce two new messages, *Ack* and *Promise*, as indicators of path quality. An *ack* message, sent to the previous hop by the destination, confirms that it is able to receive the data messages. This is also important for determining connectivity quality to the destination. Similarly, *promise* messages enable nodes to verify connectivity with their non-destination contacts. Note that a *promise* only indicates a contact's willingness to forward the message but does not guarantee it.

More specifically, on receiving a message from node $j$, a node $i$ updates the following information in its contact table: (1) the time the first message is received, $t_{start}$ (2) the time the last message is received, $t_{end}$ and (3) for a specific destination $m$, the ratio of promise (or ack) messages sent by $j$ to the number messages sent by $i$, $d^i_{j,m}$. Hence, for each neighbor $j$ the following tuple is maintained: $< t_{start}, t_{end}, .., (m, d^i_{j,m}), .. >$. While during the connection time, the node updates the same tuple for the given contact, a new tuple is started after a disconnection. We next present how a node detects a disconnection of a contact.

## 3.2 Detecting disconnection

Once connectivity is established with neighboring nodes, a node monitors if it is still connected to a given neighbor observing the time period between heard messages (i.e., hello, data, promise, ack). As the time interval between consecutive messages from a neighbor increases, it is assumed that the neighbor is moving away (and hence, its *availability* decreasing) and potentially a longer disconnection will ensue. A node $i$ decides a disconnection has occurred if the availability for a neighbor node $j$, $a^i_j$ falls below a threshold, $\delta$. More specifically, availability is calculated as $a^i_j = \max\{\delta, 1 - silence/\widehat{silence}\}$ where $silence$ is the time between observed messages and $\widehat{silence}$ is maximum silence that can be tolerated by a node. Node $i$ decides that node $j$ disconnected, if $a^i_j = \delta$.

**Discussion:** We use fixed values for the $\widehat{silence}$ and $\delta$. We observed in our simulations that adapting $\widehat{silence}$ based on the running average of inter-message times creates high dependency to traffic patterns and degrades performance. Therefore, $\widehat{silence}$ is set as two times the hello interval. Additionally, when $silence$ approaches 90% of $\widehat{silence}$ (i.e., $\delta = 0.1$), we suspect a disconnection. The reason behind not trying to set $\delta$ optimally is as follows. If $\delta$ is too low, nodes lose packets due to late disconnection detection and if $\delta$ is too high, nodes decide a disconnection too early and underestimate the contact quality. However, when calculating the contact quality, which is explained next, we take some measures to reduce these effect and hence, $\delta = 0.1$ suffices.

As nodes detect connections and disconnections, they update their tables accordingly. In case the contact history table is full, the new contact might replace an old contact with low quality value. The size of the contact history and the number of tuples maintained for each contact are design parameters, which depend on the prediction method in use. Based on the recorded information, nodes estimate the quality of their contacts, which is the topic of the next section.

# 4 Estimating Contact Quality

In Seeker, nodes that have good contacts have a high probability to forward a message. To evaluate contact quality, each node $i$ uses its contact history to calculate (1) estimated remaining time to meet a contact $j$, $t^i_{j,wait}$ and (2) expected *service quality* from the contact $j$, $c^i_{j,m}$ (i.e., the rate of responding with ack or promise messages to data messages for destination $m$). Then, using $t^i_{j,wait}$ and $c^i_{j,m}$, a node $i$ calculates a quality value for each neighbor $j$ for a given destination $m$, denoted as $q^i_{j,m}$, as follows:

$$q^i_{j,m} = e^{-\frac{t^i_{j,wait}}{t^i_j(k+1))^P}} \cdot c^i_{j,m} \tag{1}$$

As desired, $q^i_{j,m}$ increases with increasing service quality and decreasing $t^i_{j,wait}$. Instead of using $t^i_{j,wait}$ directly, $t^i_{j,wait}$ is normalized with $(t^i_j(k+1))^P$, the expected inter-contact meeting time between $i$ and $j$, detailed hereafter. This is performed in order to avoid problems with varying time granularity with different mobility patterns.

In this section, we first present how $t^i_{j,wait}$ and $c^i_{j,m}$ are calculated. Second, we present how estimations that are necessary to calculate these values are made.

## 4.1 Calculating remaining time and service quality

The remaining time to meet a node, $t^i_{j,wait}$ is calculated as follows. We denote the expected inter-contact meeting time between node $i$ and node $j$ for the $(k+1)^{th}$ time as $(t^i_j(k+1))^P$ (where $P$ indicates that this is a predicted value). At time $t$, given $(t^i_j(k+1))^P$:

$$t^i_{j,wait} = \begin{cases} t^i_{j,last} + t^i_j(k+1))^P - t & \text{if disconnected} \\ 0 & \text{if connected} \end{cases} \tag{2}$$

where $t^i_{j,last}$ is the last recorded meeting time of node $j$ in node $i$'s contact history table. However, if node $i$ has not detected a disconnection to node $j$, then $t^i_{j,wait} = 0$.

The service quality between nodes $i$ and $j$ for destination $m$, $c^i_{j,m}$, on the other hand, is determined by several factors. For instance, $c^i_{j,m}$ depends on whether $j = m$, whether node $i$ is currently connected to node $j$ or not. Additionally, the number of messages sent in a row without hearing any promise or ack in response, referred as $loss$, is also indicative of low service quality. More specifically, if node $j$ is not the destination and $loss$ is greater than a threshold $\Delta$, then node $i$ concludes that it has a bad connection to node $j$. Essentially, this also allows saving messages when $\delta$ is selected too small, and hence, disconnection detection is slow. Finally, if node $i$ is connected to node $j$, it uses the current delivery ratio, $d^i_{j,m}$, as a quality measure. The delivery ratio, $d^i_{j,m}$, is the ratio of promise or ack messages to the sent messages. If node $i$ and node $j$ are disconnected, node $i$ relies on the predicted value of the delivery ratio, $(d^i_{j,m})^P$ as a service quality measure. The following formula summarizes the calculation of $c^i_j$:

$$c^i_{j,m} = \begin{cases} 0 & , j \neq m, loss \geq \Delta, \\ d^i_{j,m} & , j \neq m, loss < \Delta, t^i_{j,wait} = 0, \\ (d^i_{j,m})^P & , j \neq m, loss < \Delta, t^i_{j,wait} > 0 \\ \min\{1, \beta \cdot d^i_{j,m}\} & , j = m, t^i_{j,wait} = 0, \\ \min\{1, \beta \cdot (d^i_{j,m})^P\} & , j = m, t^i_{j,wait} > 0 \end{cases} \tag{3}$$

Note that, using $\beta > 1$, the delivery ratio for a destination is artificially increased. Essentially, such boosting of the service quality of the destination allows prioritizing destinations over non-destination contacts. Hence, governed by the $\beta$ parameter, node $i$ improves its chances for waiting for the destination to send in one hop, unless the quality of the neighbor nodes are sufficiently high to motivate multi-hop communication.

### 4.2 Estimating remaining time and service quality

In this section, we discuss how we estimate inter-contact time, $(t_j^i)^P$ and delivery ratio, $(d_{j,m}^i)^P$. Our estimations use exponential weighted moving average (EWMA), which gives higher priority to recent observations. This type of prediction is widely used by the networking community, for instance, for round trip time (RTT) estimation in TCP [21].

**Expected inter-contact meeting time:** We calculate $(t_j^i(k+1))^P$ as follows:

$$(t_j^i(k+1))^P = (1-\alpha) \times (t_j^i(k))^P + \alpha \times t_j^i(k), \tag{4}$$

$0 \leq \alpha \leq 1$ is the constant smoothing factor, $t_j^i(k)^P$ is the previously estimated inter-contact meeting time, and $t_j^i(k)$ is the observed inter-contact meeting time. Estimating $(t_j^i(k+1))^P$ more precisely may also be possible by using the results of current work on human mobility models, which show that pair-wise inter-contact meeting times exhibit a good fit with exponential or log-normal distributions [6]. Nevertheless, we chose to use EWMA for its simplicity. Furthermore, the current studies find a fit after examining the entire connectivity pattern. In our case, the parameters of the distribution need to be determined online as new contacts are established. Therefore, we leave the study of such estimators for inter-contact meeting time as future work.

**Expected delivery ratio:** To calculate the expected delivery ratio for a destination $m$, the current delivery ratio of the neighbor, $d_{j,m}^i(k)$ should be sampled during the connection time. Hence, nodes record $d_{j,m}^i(k)$ at random sampling times until a disconnection is detected. On meeting node $j$ after a disconnection, node $i$ selects $t_s$, which is the time to take a sample, based on the duration of the previous encounter (the default value is $t_{hello}$ if this is the first meeting). Hence, if the duration of the previous meeting is $t_d^{prev}$, the sampling time $t_s = U(0, t_d^{prev})$. Taking a sample consists of updating $(d_{j,m}^i(k+1))^P$, with the current $d_{j,m}^i(k)$ at $t_s$, as follows:

$$(d_{j,m}^i(k+1))^P = (1-\alpha) \times (d_{j,m}^i(k))^P + \alpha \times d_{j,m}^i(k) \tag{5}$$

After each sampling, the node resets its counters for sent, promise and ack messages and selects a new sampling time. This time, $t_s$ is chosen based on the length of current connection, $t_d^{current}$. Specifically, the new sampling time is $t_s = U(0, t_d^{current})$. This continues until node $i$ detects a disconnection, when it takes the last sample for this connection.

These estimations based on contact history allow a node to understand the quality of its contacts. In the next section, we present an opportunistic forwarding mechanism that adjusts forwarding decisions based on this contact quality.

## 5 Adaptive Forwarding

The main goal of our opportunistic forwarding mechanism is to use contact histories to make adaptive forwarding decisions. This approach is motivated by the recent user

mobility traces that show a bias among different contacts [26]. Our goal is to utilize such bias for more reliable and resource-efficient communication. In the remainder of this section, we first explain the forwarding mechanisms and next, discuss how nodes take forwarding decisions. Finally, we conclude how these decisions can be modified based on current conditions.

## 5.1 Forwarding mechanism

In Seeker, forwarding is opportunistic as it makes use of nodes in the current neighborhood in addition to exploiting predictions based on contact history. Hence, while estimations based on the contact history suggest the best time to send the message, at the send time, the message is locally broadcast so that all neighbors receive it. As the message does not identify a next hop node, the responsibility of forwarding the data further is left to the receivers of the message. In Seeker, depending on the parameter setting, each message is sent up to $x$ times to improve reliability. After a node processes a message, it adds the sequence number of the message to a table to gain immunity to further infections.

## 5.2 Forwarding decisions

In Seeker, on each message receipt, each node makes a "forward" or "not forward" decision. If the decision is "not forward", the message is still buffered for a short time to allow reevaluating decisions and saving messages. In the remainder of this section, we explain forwarding with and without contact history, and reevaluating forwarding decisions in more detail.

### 5.2.1 Forwarding without contact history

Initially nodes may not have enough history to make estimations, therefore, messages are broadcast with a fixed probability $p$. In this "no prediction" state, as a node receives hello, promise, ack, and data messages, it populates its contact history. When the node is able to make predictions, it switches to forwarding using contact history. We assume that once a node moves out of this "no prediction" state, it does not return to this state. However, if a node is disconnected, it might not be able to make accurate predictions. Nevertheless, the duration of bad decisions due to erroneous estimations is limited to the time interval of hello messages. A problem might arise only in the case where the neighbors of a node can hear the node but the node cannot hear anybody (i.e., all the links are asymmetric and outward), which we expect to be highly unlikely.

### 5.2.2 Forwarding with contact history

As a node $i$ populates its contact table, it bases its forwarding decisions on the (1) number of contacts and (2) the quality of these contacts (i.e., $q^i_{j,m}$, which is explained in detail in Section 4.1). Hence, the messages are no longer forwarded with a fixed probability $p$ but with a higher or lower probability depending on the state of contact history. In Seeker, the quality of contacts are used to determine this forwarding probability, $p^i$ as follows:

$$p^i = \max\{p^i_{max}, {p^i_{avg}}^{\frac{1}{n}}\}, \tag{6}$$

where for all $j$, $p^i_{max} = \max q^i_{j,m}$, $p^i_{avg} = \sum_j q^i_{j,m}/n$ and $n$ is the number of contacts. The goal of Eq. 6 is to emphasize both quality and the number of contacts, when deciding $p^i$. For instance, if a node has the following list of $q^i_{j,m}$ values, $Q = \{0.2, 0.3, 0.4, 0.5, 0.4\}$, $p^i = 0.76$ due to a large number of contacts. This probability

is $0.95$ if the node has $Q = \{0.8, 0.9, 0.9, 0.9, 0.8\}$ (i.e., the same number of neighbors but with better $q^i_{j,m}$). On the other hand, if $Q = \{1, 0.1\}$, $p^i = 1$ since the node has one very good contact.

Based on $p^i$, if a node decides to forward a message, it places the message in its *send buffer*. Otherwise, the message is put in its *quarantine buffer*. (Buffer management is explained in detail in Section 6.) After putting a message in send buffer, the node sets a timer, the value of which is the $t^i_{j,wait}$ of the contact with the best $q^i_{j,m}$. If the best contact is already connected (i.e., $t^i_{j,wait} = 0$), the message is still buffered for a short period of time, denoted as $t_{jitter}$, to monitor the neighborhood and drop the message if there is much redundancy (see next section for further detail). Note that the value of the timer might change before it expires if a node hears new hello, promise or ack messages from good contacts. This actually drives the opportunistic operation of forwarding mechanism and allows taking advantage of new meetings without being too tied to the prediction mechanism.

When the send timer times out, the node sends the oldest buffered message for the given destination as a local broadcast. The send timer is rescheduled to the earliest time the next message can be sent based on the current conditions. This continues until all buffered data is sent.

#### 5.2.3 Reevaluating forwarding decisions

Seeker allows rewinding forwarding decisions based on the current observations to improve performance and cost. For instance, since multiple nodes can rebroadcast a message, multiple copies of the same message might be received by a destination. To avoid this, nodes reevaluate their forwarding decisions as follows: (1)The nodes that are in the vicinity of the destination and hear the destination send an ack, cancel the transmission of the message and (2) if a node hears a threshold number of *promise* messages for a data message, it again cancels the transmission. To realize this, nodes delay sending *promise* messages by $t_{jitter}$. If the number of overheard messages reaches a threshold, $overhear$, before a *promise* message is sent, that transmission is canceled. Based on [28], if $overhear > 4$, the expected additional coverage from sending a broadcast is less than 5%. Hence, we calculate the threshold based on the number of neighbors, $n_{nr}$ as follows:

$$overhear = \min\{4, n_{nr} * f_r\}, \tag{7}$$

where $f_r$ represents the desired redundancy and $0 \leq f_r \leq 1$. On canceling a transmission, if a node needs to send $x$ copies of a message, it reduces $x$ by one. Since nodes that overhear each other's messages eventually drop these messages, the redundant copies are transmitted by nodes that do not hear each other. Hence, copies of the same message have a higher probability to be carried by independently moving nodes, increasing the chance to meet the destination. Therefore, reevaluating forwarding decisions in this fashion will allow reaching the destination through fewer nodes.

While rewinding a forward decision improves cost, rewinding a *not forward* decision might improve performance. A node changes its *not forward* decision if it observes no response for a message (i.e., a *promise* or an ack). Specifically, if a node does not hear any response during a threshold time $t_{qack}$ and its forwarding probability for this message is greater than 0, then it moves the message to its send buffer.

These corrective measures are similar to the ones described in [8], which are shown to be effective to improve broadcast in wireless networks. A counter-based scheme which drops packets after overhearing a threshold number of the same packet is also

proposed in [28] to reduce broadcast overhead. Additionally, some of the inspiration comes from directed diffusion [16], where higher quality paths are reinforced via interest dissemination. However, in directed diffusion, such reinforcement is explicitly triggered by the sender, whereas, in our case, it is enabled in an implicit manner, through opportunistic forwarding. Next, we explain buffer management of Seeker, which enables this type of reinforcements.

# 6 Buffer Management

Buffer management in Seeker becomes important as it allows storing data in the expectation of meeting a good contact. Furthermore, it allows rewinding some forwarding decisions. Hence, Seeker uses two different buffers:

- *Send buffer:* Stores messages that are waiting to be sent. If the node chooses to meet a better contact, the message is buffered until the meeting time of that contact.
- *Quarantine buffer:* If a node decides not to forward a message, it is stored for a short time in quarantine buffer.

In *send buffer*, the entry of each message contains the time the message is inserted into the buffer, the time the message should leave the buffer, the number of times it has been sent, the number of promise messages overheard, and whether a promise message has been sent. Based on these fields, the node does the bookkeeping of (1) if all $x$ copies are sent, (2) if $overhear$ is reached and (3) if the message can be deleted from the buffer.

The *quarantine buffer* has the same structure except the fields that record (1) the number of times the message has been sent and (2) whether a promise message has been sent. When the node overhears an ack or hears an $overhear$ number of *promise* messages, it can safely drop the message.

When deleting messages from *send buffer*, we use *source prioritized drophead* policy [30], where a node drops the oldest relay messages first followed by the oldest source messages. It is shown that with this policy, epidemic routing achieves almost the same performance as with an infinite buffer. As this complexity is not necessary in quarantine buffer, here, oldest messages are deleted first. While we chose these policies for their simplicity and good performance, Seeker can also work with more complex buffer management policies, which choose the messages to be discarded optimally [18].

# 7 Performance Evaluation

Seeker is designed to take into account both the past behavior and current conditions to guide forwarding decisions. The goal of our evaluation is to show how such a design enables flexibility in significantly different environments in terms of mobility. Essentially, Seeker is expected to behave like the best performing choice and even achieve performance gains when node buffers and bandwidth are stressed, at each environment. We study four environments: (1) static network, (2) small network with a single mobile relay that guarantees connectivity and two networks, where the connectivity patterns

Table 1: Simulation Parameters

| Contact history and prediction parameters | | |
|---|---|---|
| K | Contact history size | 10 |
| $\Delta$ | Consecutive loss threshold | 3 |
| $\alpha$ | EWMA constant (Eq. 4) | 0.9 |
| $\beta$ | Sink priority (Eq. 3) | 2 |
| Opportunistic forwarding parameters | | |
| $p$ | Initial forward prob. | 0.7 |
| SB | Send buffer size | 64 |
| QB | Quarantine buffer size | 50 |
| $t_{hello}$ | Hello interval | 5 s |
| $t_{qack}$ | Quarantine timeout | 1 s |
| $x$ | Max. # of gossips | 1-6 |
| $f_r$ | Desired redundancy | 0.5 |
| $t_{jitter}$ | Observation jitter | 0.01 s |

of nodes follow the traces collected for (3) the Reality Mining project at MIT [10] and (4) Haggle project at Cambridge [24].

We compared Seeker with the following protocols.

- *Epidemic Routing (Epidemic)*: When two nodes meet, they exchange their packet summaries and based on this, exchange request packets that are not in their buffers [29].
- *Delegation Forwarding - Frequency (D-F)* [11]: Each node $i$ records the total number of nodes it meets as $\tau_i$. A node $i$ forwards a message to a node $j$, if $j$ has a larger total number of contacts (i.e., $\tau_i < \tau_j$). Node $i$ adopts $\tau_j$ as its new quality threshold.
- *Delegation Forwarding - Destination Last Contact (D-DLC)* [11]: Each node $i$ records the time it met a destination $m$ as $\tau_{im}$. A node $i$ forwards a message to node $j$, if node $j$ met the destination $m$ more recently than node $i$ (i.e., $\tau_{im} < \tau_{jm}$). Node $i$ sets $\tau_{jm}$ as its new quality threshold.

Additionally, for the static network study, we compare Seeker with DSDV [22] (the best performing choice of this scenario). All protocols were implemented using ns2. To evaluate reliability and cost, we use five metrics: (1) delivery ratio, which is the ratio of packets delivered to the destination, (2) control overhead, which is the number of hello, ack, packet summary messages per data message, (3) message overhead, which is the buffering and forwarding count in the network per data message, (4) hop count defined as the average number of hops to destination, (5) delay defined as the time between a message is generated and delivered (including buffering delays). The protocol parameters used in simulations are summarized in Table 1. The network settings are presented in their respective sections. The transmission range of nodes is 250 m. Each source generates 256 B packets CBR. Different than previous studies, our evaluations neither assume infinite buffers nor infinite bandwidth.

## 7.1 Case 1: Static network

We first evaluate Seeker in a static environment, where 39 nodes are distributed uniformly random in $500 \times 1500\ m^2$. The average node degree is $\approx 7.1$. Given a high number of neighbors and a connected network, our goal is to show that Seeker is able to find and use stable routes.

Fig. 1 depicts the delivery ratio when the percentage of senders in the network is 25-100% of all nodes. Each sender generates messages at 2 Kb/s to a single destination for

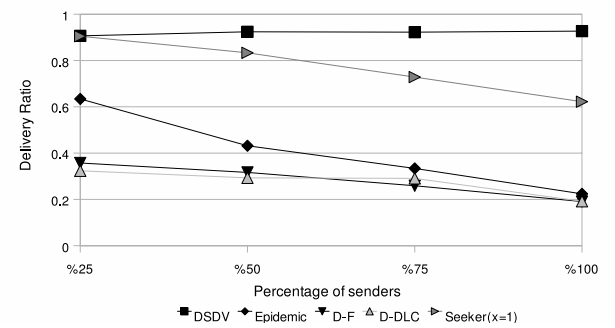


Figure 1: Static network: Delivery ratio.

300 s. Despite its opportunistic operation, Seeker performs similar to DSDV when the number of senders is low. However, as the number of senders increases, the delivery ratio degrades from 0.9 to 0.62. This is due to Seeker sending an ack for each data message, which increases contention. As future work, we plan to try sending a single ack for a window of messages. This will be similar to Epidemic, which has a lower control overhead (in terms of packets, not bytes) as one summary message is sent for all buffered messages. However, the overhead of Epidemic is still high as each message is forwarded to each contact. Even for 25% senders, $\approx$ 75% of nodes forward a message in Epidemic, compared to $\approx$ 48% in Seeker. This high overhead also results in high contention and hence, low delivery ratio. In contrast, D-F and D-DLC suffer from being too conservative. In D-F, each node waits for a node that has higher number of contacts than seen so far, which might not ensure a path to the destination. In D-DLC, each node waits for a node that has met the destination later than any node it has seen so far. However, this information might propagate slowly in a static network. Hence both D-F and D-DLC forwards fewer messages, which results in poor delivery ratio.

We also calculate the route quality of Seeker as the percentage of messages that are sent over longer routes compared to the ideal case, i.e., the shortest-path for each flow is computed offline. Our results for 25% senders case show that 73-87% of the messages are delivered with at most 1-hop increase. Furthermore, given the number of messages received by the destination, $r_k$, and the number of unique messages received by the destination, $u_k$, for a flow $k$, the redundancy factor is $(r_k - u_k)/r_k$ and in Seeker, range between 17-38%. This shows that Seeker is able to find and use good routes effectively.

## 7.2 Case 2: Single mobile relay

In DTNs, it is essential to exploit transitive contacts to enable message delivery. To show Seeker is able to learn and to adapt accordingly, we use the following example, where the source $s$ and destination $d$ are separated by 600 m and a relay node $i$ travels this distance enabling connectivity between them. There is also a second static node $j$, which is in the range of $s$ and sometimes $i$, when $i$ gets close enough to $s$. The only way to ensure delivery is to forward the data to node $i$. The scenario is simple but significantly challenging since there is a highly available neighbor, $j$, which seems attractive for forwarding but useless to reach the destination.

In our simulations, $s$ generates a message every $\approx$ 5 s until 200 messages (i.e., at a rate 0.2 Kb/s). The simulation run is 4555 s. Fig. 2 depicts the control and message

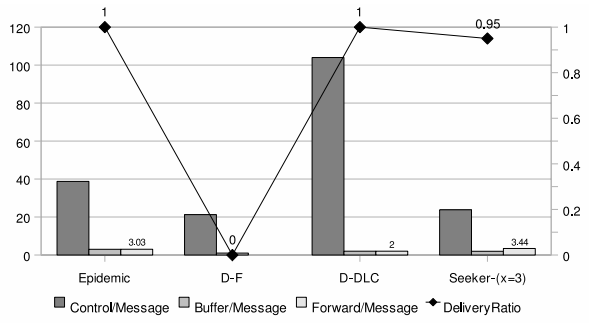


Figure 2: Single mobile relay: Control and message overhead (left y axis) and delivery ratio (right y axis).

overhead, and the delivery ratio. Except D-F, all protocols have high delivery ratio. The delivery ratio of D-F is 0 since $i$ and $s$ have the same number of contacts. D-DLC has 100% delivery ratio but also the highest control overhead since for each message, the destination contact times are compared. Furthermore, with D-DLC the first message is forwarded at time $\approx 198$ (i.e., after $i$ met $d$ and returned to $s$) compared to $\approx 104$ both in the case of Seeker and Epidemic. Note that Epidemic incurs 39% more control overhead than Seeker as it sends a summary message to both its neighbors. We also see the same effect when we calculate the ratio of messages that are sent when $i$ is not in $s$'s neighborhood (S-R miss ratio) and when $d$ is not in $i$'s neighborhood (R-D miss ratio). When calculating these ratios, a batch of messages is counted as one message if the time difference between transmissions is $\approx 0$. Hence, with Epidemic S-R miss ratio is $\approx 80\%$ (since $s$ forwards each message to $j$ as it generates them, while it sends a batch of messages when $i$ returns). In comparison, Seeker has 28% S-R miss ratio, which shows that $s$ was able to learn about the better relay $i$. However, the R-D miss ratio of Seeker is 18% due to wrong predictions, while with Epidemic, $i$ always forwards messages to $d$ timely.

## 7.3 Case 3: Mobility based on traces

In this section, we present how Seeker behaves when the nodes follow a mobility pattern based on MIT and Cambridge traces. For each run, 16 source nodes are selected uniformly random to send 10 messages to a single destination. Next, we filter the flows which do not have a path or do not stay connected long enough to the destination by running Epidemic with each source separately. The results with multiple destinations are not presented for the sake of brevity as they show similar trends with lower delivery ratios.

**MIT Reality Mining traces** Due to the long length of MIT traces, we chose 3 different days with the highest number of connections. The results depict an average of 30 scenarios with 2-10 randomly generated flows.

The characteristic of MIT traces is that connectivity opportunities are scarce: $\approx 50\%$ of the nodes meet only once and $\approx 94\%$ of the nodes meet at most three times. Therefore, Epidemic achieves the best delivery ratio as it forwards messages to all neighbors (see Fig. 3). In contrast, D-F and D-DLC, which are more conservative,

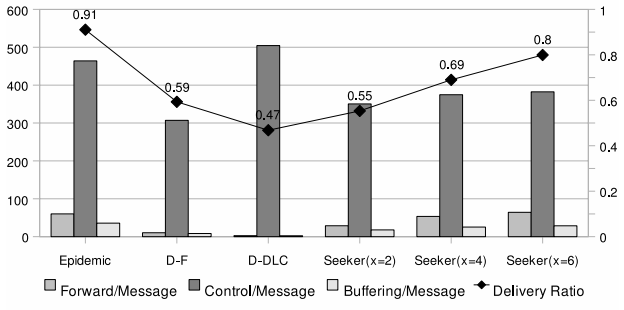


Figure 3: MIT Trace: Control and message overhead (left y axis) and delivery ratio (right y axis).

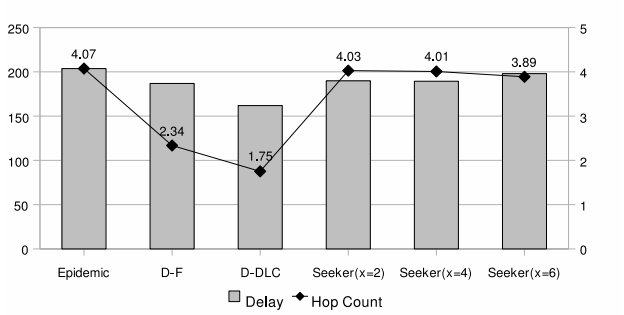


Figure 4: MIT Trace: Delay (left y axis) vs. hop count (right y axis)

achieve low delivery ratios, $59\%$ and $47\%$, respectively. Additionally, D-DLC incurs the highest control overhead. We evaluated Seeker's performance when nodes forward a message up to $x = 2 - 6$ times. Essentially, increasing $x$ improves the delivery ratio without much effect on the overhead. Note that such tuning is not possible with D-F or D-DLC. However, as this does not mean $x$ distinct neighbors receive the message (since Seeker does not identify the next hop in the message), the additional copies might be wasted by being received by the same nodes. Therefore, while Seeker does not reach $91\%$ delivery ratio, the delivery ratio improves to $\approx 80\%$ with $x = 6$. However, note that this is the best case scenario for Epidemic, as its control, forwarding, and buffering overhead could be amortized by its high delivery ratio. It would incur similar overhead even though it cannot deliver any messages, whereas the message overhead of Seeker is low when the delivery ratio is low (e.g., less messages are acked). Fig 4 depicts the delay and the hop count. While Seeker and Epidemic perform similarly, D-F and D-DLC have slightly lower delays due to shorter hop counts. As expected, in D-F and D-DLC messages do not find a chance to be propagated far and hence the hop counts are short. However, they incur additional delays for waiting for better contacts.

**Cambridge Haggle Traces** The experiments in this section were run similarly to MIT traces. The results depict an average of 10 scenarios, which include 4-8 flows. In contrast to MIT traces, in Cambridge traces, contact opportunities are plenty: only

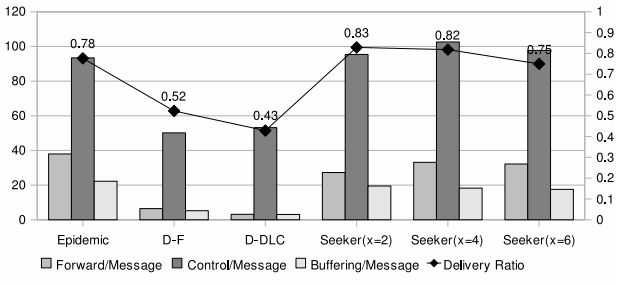


Figure 5: Cambridge Trace: Control and message overhead (left y axis) and delivery ratio (right y axis).

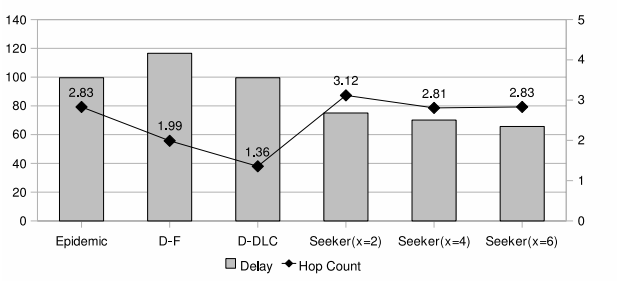


Figure 6: Cambridge Trace: Delay (left y axis) vs. hop count (right y axis)

$\approx 20\%$ of the nodes meet less than 8 times. Hence, Seeker has more chance to learn recurring patterns.

The results also verify our expectations, as Seeker is able to achieve better delivery ratio than Epidemic due to its lower overhead (see Fig. 5). Note that, although Seeker has comparable control overhead compared to Epidemic (due to sending an ack for each message as discussed in section 7.1) it is able to reduce the number of times each message is forwarded and buffered. Additionally, since there are more contact opportunities, increasing $x$ does not necessarily yield better performance; however, it does not hurt the performance either. Also note that, D-F and D-DLC both perform similarly to MIT traces and hence, are not able to take advantage of the higher number of contacts. Finally, Fig. 6 shows that Seeker achieves better delay compared to all protocols. These results are an outcome of Seeker's ability to choose better relays and hence, to reduce message forwarding overhead. In comparison, D-F and D-DLC incur highest delays as both protocols wait too long to pass the message to their neighbors.

In summary, our performance evaluation shows that blindly trying to reduce forwarding and buffering overhead might degrade performance in terms of both delivery ratio and delay. Learning from the past and adapting to the current conditions allows Seeker to perform with high performance and low overhead in environments with significantly different mobility.

# 8 Related Work

In DTNs, communication experiences frequent and long disconnections. Therefore, current research focuses on opportunistic forwarding. In [14], messages are forwarded to the first relay met, which in turn forwards the data to the destination. Since the goal was to show that mobility can improve the capacity of wireless networks, the relays were chosen purely based on a first contact opportunity. However, the expected delay of two-hop forwarding may grow to infinity for some mobility models. This is the case under some power-law distributions, which characterizes the *aggregate* inter-contact time in some DTNs [5]. Therefore, when selecting contacts, the underlying connectivity patterns should be considered. For instance, bounded delay is possible, if the message is forwarded to multiple nodes instead of one. However, the results on empirical studies vary as a more optimistic result is given in [6], where *pairwise* inter-contact times are found to be exponentially or log-normally distributed. Empirical evidence also shows that contacts with friends are more valuable than strangers [20,26]. Nevertheless, these studies show that there is a bias among different contacts. The knowledge of contact patterns is thus, expected to improve forwarding performance.

In an ideal world, if each node knew the contact probability, waiting times [13,17], buffer limits, and traffic demands [17], the delivery probability and delay of different strategies could be computed. However, as these are not typically available, one approach is to disseminate similar information to the entire network. PREP [23] exchanges *link availability* to drop packets that have worse chance of delivery. In [27], a link-state protocol records the expected delay to each node in the network and a message is forwarded to a node only if the message would experience less delay. However, the performance is worse than epidemic routing due to the significant warm-up time of link-state tables, conflicting decisions due to state changes, and the lack of replication. While these approaches propagate information to the entire network, more local algorithms also exist [25,9,2].

Similar to Seeker, several approaches also estimate a *delivery likelihood* [3,19], which is based on the frequency of meeting with contacts. However, although nodes exchange their entire information with all contacts, [3] shows that the majority of savings come from prioritizing messages based on the current hop count and disseminating acks to delete delivered messages from buffers. More suited to social DTNs, BUBBLE [15] and SimBet [7] use information about social community structures to choose good relays. Essentially, using such quality information improves routing cost, although it does not always improve reliability [11,25].

Most importantly, the majority of the approaches assume infinite buffers and bandwidth, while exceptions are [1,18,3]. In [1], at each forward opportunity, nodes evaluate if the gain from replicating a message justifies the resources used. However, such an evaluation relies on information about the replicas of a message in the entire network. This problem is addressed in [18], where the total number of replicas are estimated locally to decide which messages to replicate or drop to achieve either minimum delay or maximum delivery ratio. Nevertheless, nodes are still required to maintain a history of each message forwarded by each node they meet.

In Seeker, we use similar information with existing approaches to make simple predictions about future meetings. However, our goal is to learn from contact *and* communication patterns to gracefully adapt to different mobility conditions. This consequently improves both reliability and cost by maintaining and exchanging as less information as possible.

# 9 Conclusion

Routing in DTNs requires careful selection of relays, scheduling transmissions, and managing buffers to guarantee reliable, timely and low-cost delivery. To this end, many approaches rely on a quality metric in an attempt to achieve a trade-off between forwarding cost and performance. In this paper, we show that this may be too conservative and hence, degrade performance in the presence of different levels of mobility. The main contribution of this paper is a new adaptive forwarding protocol, Seeker, which is indeed able to match different mobility patterns. Seeker does not consider infinite buffers or bandwidth and allows nodes to modify their forwarding strategy based on their perception of the past and current conditions. Hence, Seeker achieves high flexibility, which is invaluable for DTNs, with a simple prediction mechanism. Nevertheless, for future work, we plan to investigate the benefit of more advanced prediction mechanisms on performance.

# References

[1] A. Balasubramanian, B. N. Levine, and A. Venkaratamani. DTN routing as a resource allocation problem. In *ACM SIGCOMM*, Aug. 2007.

[2] F. Benbadis, M. D. de Amorim, and S. Fdida. Elip: Embedded location information protocol. In *IFIP Networking*, June 2005.

[3] J. Burgess, B. Gallaghern, D. Jensen, and B. N. Levine. MaxProp: routing for vehicle-based disruption-tolerant networks. In *IEEE INFOCOM*, Apr. 2006.

[4] H. Cai and D. Y. Eun. Toward stochastic anatomy of inter-meeting time distribution under general mobility models. In *ACM MobiHoc*, May 2008.

[5] A. Chaintreau, P. Hui, J. Crowcroft, C. Diot, R. Gass, and J. Scott. Impact of human mobility on the design of opportunistic forwarding algorithms. 6(6):600–620, June 2007.

[6] V. Conan, J. Leguay, and T. Friedman. Characterizing pairwise inter-contact patterns in delay tolerant networks. In *Autonomics*, Oct. 2007.

[7] E. Daly and M. Haahr. Social network analysis for routing in disconnected delay-tolerant manets. In *ACM MobiHoc*, Sept. 2007.

[8] V. Drabkin, R. Friedman, G. Kliot, and M. Segal. RAPID: reliable probabilistic distribution in wireless ad-hoc networks. In *26th IEEE Intl. Symposium on Reliable Distributed Systems*, Oct. 2007.

[9] H. Dubois-Ferriere, M. Grossglauser, and M. Vetterli. Age matters: Efficient route discovery in mobile ad hoc networks using encounter ages. In *ACM MobiHoc*, June 2003.

[10] Nathan Eagle and Alex (Sandy) Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). `http://crawdad.cs.dartmouth.edu/mit/reality`, July 2005.

[11] V. Erramilli, M. Crovella, A. Chaintreau, and C. Diot. Delegation forwarding. In *ACM MobiHoc*, May 2008.

[12] K. Fall. A delay-tolerant network architecture for challenged internets. In *ACM SIGCOMM*, Aug. 2004.

[13] J.-M. Francois and G. Leduc. Predictable disruption tolerant networks and delivery guarantees. Technical Report cs/0612034, ArXiv Computer Science e-prints, Dec. 2006.

[14] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc wireless networks. 10(4):477–486, Aug. 2002.

[15] P. Hui, J. Crowcroft, and E. Yoneki. BUBBLE Rap: Social-based forwarding in delay tolerant networks. In *ACM MobiHoc*, May 2008.

[16] C. Intanagonwinat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *ACM MobiCom*, Aug. 2000.

[17] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. In *ACM SIGCOMM*, Aug.-Sept. 2004.

[18] A. Krifa, C. Barakat, and T. Spyropoulos. Optimal buffer management policies for delay tolerant networks. In *ACM SECON*, June 2008.

[19] A. Lindgren, A. Doria, and O. SchelÃl'n. Probabilistic routing in intermittently connected networks. 7(3):19–20, July 2003.

[20] A. G. Miklas, K. K. Gollu, K. K. W. Chan, S. Saroiu, K. P. Gummadi, and E. de Lara. Exploiting social interactions in mobile systems. In *UbiComp*, Sept. 2007.

[21] V. Paxson and M. Allman. Computing TCP's retransmission timer. IETF RFC 2988, Nov. 2000.

[22] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM*, Aug.-Sept. 1994.

[23] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan. Prioritized epidemic routing for opportunistic networks. In *1st Intl. Workshop on Mobile Opportunistic Networking (MobiOpp)*, June 2007.

[24] James Scott, Richard Gass, Jon Crowcroft, Pan Hui, Christophe Diot, and Augustin Chaintreau. CRAWDAD data set cambridge/haggle (v. 2006-09-15). `http://crawdad.cs.dartmouth.edu/cambridge/haggle`, Sep. 2006.

[25] T. Spyropoulos, K. Psounis, and C. Raghavendra. Efficient routing in intermittently connected mobile networks: The single-copy case. 16(1):63–76, Feb. 2008.

[26] J. Su, A. Chin, A. Popivanova A. Goel, and E. de Lara. User mobility for opportunistic ad-hoc networking. In *IEEE WMCSA*, Dec. 2004.

[27] J. Su, A. Goel, and E. de Lara. An empirical evaluation of the student-net delay tolerant network. In *Mobiquitous*, July 2006.

[28] Y.-C. Tseng, S.-Y. Ni, and E.-Y. Shih. Adaptive approaches to relieving broadcast storms in a wireless multihop mobile ad hoc network. 52(5):545–557, May 2003.

[29] A. Vahdat and D. Becker. Epidemic routing for partially-connected ad hoc networks. Technical report, Duke University, 2000.

[30] E. Zhang, G. Neglia, J. Kurose, and D. Towsley. Performance modeling of epidemic routing. Technical Report 2005-44, University of Massachusetts, Amherst, 2005.

# Contents

**1 Introduction** **3**

**2 Emphasizing flexibility** **4**

**3 Maintaining Contact History** **5**
3.1 Detecting connection . . . . . 5
3.2 Detecting disconnection . . . . . 6

**4 Estimating Contact Quality** **7**
4.1 Calculating remaining time and service quality . . . . . 7
4.2 Estimating remaining time and service quality . . . . . 8

**5 Adaptive Forwarding** **8**
5.1 Forwarding mechanism . . . . . 9
5.2 Forwarding decisions . . . . . 9
5.2.1 Forwarding without contact history . . . . . 9
5.2.2 Forwarding with contact history . . . . . 9
5.2.3 Reevaluating forwarding decisions . . . . . 10

**6 Buffer Management** **11**

**7 Performance Evaluation** **11**
7.1 Case 1: Static network . . . . . 12
7.2 Case 2: Single mobile relay . . . . . 13
7.3 Case 3: Mobility based on traces . . . . . 14

**8 Related Work** **17**

**9 Conclusion** **18**

Centre de recherche INRIA Saclay – Île-de-France
Parc Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 Orsay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399