Performance Analysis of Delay Tolerant Networks with Model Checking Techniques

Michele Garetto, Marco Gribaudo
Dipartimento di Informatica, Università di Torino
Corso Svizzera 185, 10149 Torino, Italy
{marcog, garetto}@di.unito.it

Abstract

Delay Tolerant Networks are a class of wireless networks that has recently attracted a lot of attention from the networking community. They are characterized by frequent network partitioning, intermittent connectivity, long and variable delays, high error rates. Such performance-challenging conditions are usually found in environments populated by a sparse set of fixed or mobile nodes with limited communication capabilities. The store-carry-and-forward *principle, according to which messages can be stored at mobile nodes moving around the network area before being forwarded to the destination, has emerged as the new communication paradigm for delay tolerant networks, demanding novel performance evaluation tools and methodologies with respect to those developed in traditional* store-and-forward *packet networks. In this paper, we describe an analytical framework to study Delay Tolerant Networks, based on model checking techniques. In particular we employ the logic* asCSL *as a powerful formalism to define specific performance metrics suitable to analyze the behavior of these systems.*

1. Introduction

In recent years Delay Tolerant Networking (DTN) has emerged as a new area of research with many promising applications [1, 5]. Delay tolerant networks are characterized by intermittent connectivity, network partitioning, long and variable delays, high error rates. Such performance-challenging conditions can be found in many different environments such as vehicular networks (VANET), sparse sensor/actuator networks, in-the-field military or disaster-relief networks, satellite and deep-space interplanetary networks [2], terrestrial networks serving remote or rural areas. Interesting real-life experiments comprise bus-based DTN networks [13], sensor networks for wildlife tracking [10], Internet access to remote villages in underdeveloped countries [11, 12].

A typical DTN scenario consists of a sparse network of fixed or mobile wireless devices, where most of the time there does not exist a complete path from a source to a destination, or such a path is highly unstable and may soon break. Indeed, two nodes can communicate (i.e., establish a link between them) only when they happen to be in communication range of each other [1]. Communication occurs in a multi-hop fashion, which allow a node to reach far away destinations using intermediate nodes as relays. Over time, different links come up and down due to node mobility. This implies that a message could be sent over an existing link and be buffered at the receiving node for quite a long time before being forwarded on the next available link toward the destination. While being stored at a mobile node, a message is physically carried to a different location in the network. This new communication concept is usually referred to as *store-carry-and-forward* or *mobility-assisted routing*.

The Delay Tolerant Network Research Group (DTNRG) [1] has proposed an architecture [3] to support messaging in such a context. The architecture consists mainly of the addition of an overlay, called the bundle layer, above a network's transport layer [4]. A *bundle* is a large block of user application data which is moved as a whole from a storage place on one node to a storage place on another node along a path that eventually reaches the destination. Intermediate nodes can be requested to act as *custodians* of the bundle, i.e., take responsibility of eventually transferring the message one more hop toward the destination. Actually, data communication in DTN closely resembles the operation of the traditional postal service. Traffic is typically non-interactive and one-way.

While DTN applications are expected to tolerate much longer delays than usually found in an always-connected network, this does not mean that message transfer delay is not important. Indeed, data sent into the network typically has a limited lifespan, i.e., it is no longer useful after a given amount of time, thus can be discarded when the lifespan is reached. Moreover, the storage capacity of the nodes is finite, so message latency in the network must be kept under control to avoid buffer overflow.

In this paper, we propose a modeling framework to characterize delay in a generic DTN network comprising a mixture of fixed and mobile nodes. We use asCSL [22], an ex-

1 the communication range is the maximum distance at which a transmitted signal can be correctly decoded by the receiver. Typical values in ad-hoc wireless networks span from a few tens to a few hundreds meters

tended stochastic logic, as a tool to analyze the proposed model. In particular we describe a general technique to obtain interesting performance indices by means of asCSL programs and formulas. We consider quite general mobility patterns, with different degrees of randomness. However, for analytical tractability, we assume that mobiles nodes move independently of each other. Our model is able to predict the entire distribution of end-to-end delay, i.e., the amount of time that elapses since a bundle is injected into the network, til it reaches the destination. To the best of our knowledge, our approach is the only one proposed so far to derive detailed delay distributions in networks comprising a mixture of fixed and mobiles nodes following general mobility patterns. Existing techniques either consider deterministic meeting times among nodes, or random (memoryless) motion.

The paper is organized as follows. In Section 2 we overview the existing literature about DTN. In Section 3 we introduce our model of a delay-tolerant network. A brief background on the asCSL logic is provided in Section 4. In Section 5 we describe how to analyze the model using asCSL logic. Section 6 presents application examples of our performance evaluation methodology in several different case-studies. We conclude in Section 7.

2. Related Work

In the literature related to DTN, routing is the main issue that has been addressed so far. Indeed, in a delay-tolerant network traditional routing protocols fail, due to lack of end-to-end connectivity. Current proposals to route messages in a DTN strongly depends on the mobility of nodes and the nature of contacts among them. If meeting times of nodes are predictable, intelligent routing and forwarding decisions can be made, although optimal schemes that take into account also the volume of data that can be exchanged during a contact lead to complex linear programming formulations [6]. In [9], the authors assume that nodes' movement are not only known in advance, but can be controlled to further improve data collection and delivery across the network.

When nodes' mobility is random, nodes have to communicate during opportunistic, unscheduled contacts. Epidemic routing [14] is one of the first proposals to enable message delivery in intermittently connected networks with random mobility. Each node maintains a list of messages, whose delivery is pending. Whenever it encounters another node, the two nodes exchange all messages that they do not have in common. This way, all messages are eventually disseminated to all nodes, including their destination. Although optimal in terms of minimizing delay, epidemic routing is very wasteful of network resources.

To avoid flooding the entire network, messages can be duplicated in a limited number of copies [16, 17], or probabilistically [15], possibly exploiting history of encounters for a better selection of next-hops. Other approaches make use of erasure coding techniques [7] or network coding [18, 19] to cope with partial data loss and to reduce routing overhead.

Performance evaluation of DTN under random mobility has been mainly carried out by simulation. Existing analytical techniques mainly focus on variations of the random walk. In [8] authors consider mobile entities (called MULEs) moving according to a two-directional random walk over a square grid, and carrying data between a set of fixed sensor nodes and a set of access points. A discrete time Markov model is developed to compute performance metrics such as buffer requirements and data success rate. In [21] the authors study popular mobility models such as random direction and random way-point, and derive the mean meeting time averaged over all possible pairs of nodes uniformly distributed in the area.

3. Delay Tolerant Network Model

In this section we present our model of a delay-tolerant network. Nodes are partitioned into two classes: fixed or mobile. A wireless link is established between two nodes if they are within communication range of each other. The communication range thus defines a connectivity graph in which (bi-directional) edges represent wireless links. If we consider the connectivity subgraph formed by fixed nodes alone, we can identify one or more connected components that we call *islands*. Any pair of nodes belonging to the same island can communicate directly using traditional routing protocols adopting the *store-and-forward* scheme.

Nodes belonging to different islands, instead, can communicate only using mobile nodes as relays. This can happen in two different ways:

- By providing temporary connectivity between two islands a_i and a_j. This occurs, for example, when a mobile node happens to be simultaneously in communication range with a fixed node belonging to island a_i and a fixed node belonging to island a_j, allowing the traditional *store-and-forward* scheme.
- By receiving a message from a_i, storing it in its buffer, travelling around the network, and delivering the message to island a_j as soon as it gets in communication range with a fixed node belonging to a_j. This is the novel *store-carry-and-forward* scheme.

In most DTN applications, direct communication takes negligible time with respect to the delay caused by the *store-carry-and-forward* delivering mechanism (i.e., a few milliseconds compared to several seconds or even hours). For this reason, we assume that communication among directly connected nodes incurs zero delay. Notice that we also assume that the time taken to discover direct paths coming up between two islands is negligible. Indeed, this can be accomplished in marginal time with respect to the time-scale of node movements, employing well-known route discovery mechanisms developed for wireless ad-hoc networks.

As a consequence of the above assumption, we focus on the total end-to-end delay between pairs of nodes belong-

ing to different islands, and we neglect the internal details of the islands considering them as single entities. Since islands do not change over time, it is reasonable to assume that sources know in advance the sequence of islands to be traversed to reach the destination. Of course, there can be more than one sequence of islands to reach a destination. In this case, we assume that routing occurs opportunistically among a (restricted) set of alternative paths: the first path that becomes available thanks to nodes' mobility, is immediately used. Alternative paths can be pre-computed for each source-destination island pair using, for example, geographical forwarding techniques [20].

Based on our assumptions, we formally define a DTN by a tuple:

$$\mathcal{D} = \langle \mathcal{I}, \mathcal{P}, f_P(\cdot), \mathcal{M} \rangle$$

$\mathcal{I}$ represents the set of islands. The portion of the network area reached by mobile nodes is approximated by a finite discrete set of positions $\mathcal{P}$. At each time instant, a mobile node is located at one of the possible positions. When a mobile node is at a given position $p_i \in \mathcal{P}$, it can be in the communication range of nodes belonging to one or more islands. The function $f_P : \mathcal{P} \rightarrow 2^{\mathcal{I}}$, where $2^{\mathcal{I}}$ represents the set of all possible subsets of $\mathcal{I}$ (including the empty set $\emptyset$), specifies the set of islands in contact with a given position. In particular, island a_j is in the communication range of a mobile node in position p_i, if $a_j \in f_P(p_i)$.

The motion of mobile nodes is described using a set of state-labelled finite Markov chains $\mathcal{M}$. Each element $m_i \in \mathcal{M}$ represents the motion of a particular mobile node. The cardinality $|\mathcal{M}|$ of the set is thus equal to the number N of mobile nodes. For the sake of analytical tractability, we assume that mobile nodes move independently of each other. Each labelled Markov chain $m_i = \langle \mathcal{S}_i, \boldsymbol{Q}_i, l_i(\cdot) \rangle$ is defined by its state space $\mathcal{S}_i$, its infinitesimal generator $\boldsymbol{Q}_i$, and its labelling function $l_i(\cdot) : \mathcal{S}_i \rightarrow \mathcal{P}$. The labelling function associates each state of the Markov chain with the position currently occupied by the node: mobile m_i is located at position $p_k = l_i(s_{i;j})$ if the corresponding Markov chain is in state $s_{i;j} \in \mathcal{S}_i$.

We remark that our description of the motion of a mobile node through a state-labelled Markov chain is very general and flexible, and it allows to represent both random-like movements, and deterministic or pseudo-deterministic routes. For example, we can define complex paths in which a mobile node visits the same position several times before moving to a different part of the network area, and we can model the sojourn time of a node at a given position using a phase-type approximation of a general distribution.

As an example, Figure 1 depicts a simple scenario with two mobile nodes, six islands, and six positions. In this scenario, the first mobile cycles around positions P_0, P_1, P_2 and P_1 again. At position P_0 it connects together island A with island C, whereas at position P_2 it connects B with D. It does not connect any island while it is at position P_1. Here, position P_1 has been introduced to better specify the route of the mobile node. We assume, for simplicity, that the sojourn time of the mobile node in each position is exponentially distributed with mean $1/\lambda$. The second mobile cycles around positions P_3, P_4, P_5 and P_3, with a mean sojourn time of $1/\mu$ at each position. It can reach island C at position P_3, island E at position P_4, and both island D and F at position P_5.

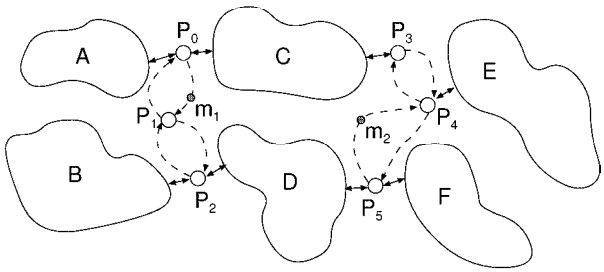

Figure 1. A network scenario

This simple scenario could represent, for example, the motion of two buses around different parts of a city. While the routes of the buses are fixed, the travelling times from one position to another are random. According to our modeling framework, the considered DTN scenario is represented by the tuple:

$$\begin{aligned} \mathcal{I} &= \{A, B, C, D, E, F\} \\ \mathcal{P} &= \{P_0, P_1, P_2, P_3, P_4, P_5\} \\ \mathcal{M} &= \{\langle \mathcal{S}_1, \boldsymbol{Q}_1, l_1(\cdot) \rangle, \langle \mathcal{S}_2, \boldsymbol{Q}_2, l_2(\cdot) \rangle\} \\ \mathcal{S}_1 &= \{s_{1;0}, s_{1;1}, s_{1;2}, s_{1;3}\} \\ \mathcal{S}_2 &= \{s_{2;0}, s_{2;1}, s_{2;2}, s_{2;3}\} \end{aligned}$$

where we have:

$$\begin{aligned} &f_P(P_0) = \{A, C\},\ f_P(P_1) = \emptyset,\ f_P(P_2) = \{B, D\}, \\ &f_P(P_3) = \{C\},\ f_P(P_4) = \{E\},\ f_P(P_5) = \{D, F\}, \\ &l_1(s_{1;0}) = P_0,\ l_1(s_{1;1}) = P_1,\ l_1(s_{1;2}) = P_2,\ l_1(s_{1;3}) = P_1, \\ &l_1(s_{2;0}) = P_3,\ l_1(s_{2;1}) = P_4,\ l_1(s_{2;2}) = P_5,\ l_1(s_{2;3}) = P_4, \end{aligned}$$

$$\boldsymbol{Q}_1 = \begin{vmatrix} -\lambda & \lambda & 0 & 0 \\ 0 & -\lambda & \lambda & 0 \\ 0 & 0 & -\lambda & \lambda \\ \lambda & 0 & 0 & -\lambda \end{vmatrix} \qquad \boldsymbol{Q}_2 = \begin{vmatrix} -\mu & \mu & 0 & 0 \\ 0 & -\mu & \mu & 0 \\ 0 & 0 & -\mu & \mu \\ \mu & 0 & 0 & -\mu \end{vmatrix}$$

4. Background on asCSL logic

The logic called asCSL has been defined in [22] as an extension of the CSL logic introduced in [23] and [24], where path are characterized by regular expressions, called *programs*. Models that can be analyzed with asCSL are defined using *Action and State-labelled Markov Chains* (ASMC). An ASMC is a tuple $\mathcal{M} = (\mathsf{S}, \mathsf{Act}, \mathsf{AP}, L, \mathbf{R})$, where S is a finite set of states, Act is the set of action labels, AP is the set of atomic propositions, $L : \mathsf{S} \rightarrow 2^{\mathsf{AP}}$ is the labelling function that associates atomic propositions to states, and $\mathbf{R} : \mathsf{S} \times \mathsf{Act} \times \mathsf{S} \rightarrow \mathbb{R}$ is the transition rate matrix, whose element $r(s_i, a, s_j)$ specifies the transition rate from a state s_i to a state s_j, associated to a given action a. For a deeper description of ASMC, please refer to [22].

The asCSL logic allows the verification of logical formulas $\phi \in \Phi$ defined by the following grammar:

$$\phi ::= q \mid \neg\phi \mid \phi \vee \phi \mid \mathcal{S}_{\bowtie p}(\phi) \mid \mathcal{P}_{\bowtie p}(\alpha^I)$$

In the previous definition, $q \in \mathsf{AP}$ is an atomic proposition, $p \in [0,1]$ is a probability value, $\bowtie \in \{<, \leq, \geq, >\}$ is a comparison operator, $I = [t, t']$ is a time interval and α is a program (defined below). The operator $\mathcal{S}_{\bowtie p}(\phi)$ is the steady state operator, meaning that the probability of being in a state where ϕ is verified on the long run is within the interval $\bowtie p$. The operator $\mathcal{P}_{\bowtie p}(\alpha^I)$ asserts that the probability measure of all infinite paths that satisfy a program α within a given time bound specified by interval I, is within the interval $\bowtie p$. A program is defined by a regular expression over the alphabet:

$$\sigma = \Phi \times (\mathsf{Act} \cup \surd) = \{\phi b \mid \phi \in \Phi \wedge b \in (\mathsf{Act} \cup \surd)\}$$

Regular expressions representing programs are defined by grammar:

$$\alpha ::= \epsilon \mid \phi b \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^*$$

Here $\surd$ represents a pseudo-action, i.e., an action (not belonging to Act) which is immediately executable and does not change the current state of the ASMC; ϕb is used to describe a state s of the ASMC where formula ϕ holds, and either there is an outgoing b-transition (if $b \in \mathsf{Act}$) or there is no statement about outgoing transitions (if $b = \surd$); ϵ represents the empty word, the operator ; denotes sequential compositions, $\cup$ represents choice, and * defines the n-fold sequential composition (Kleene star). For further details on asCSL and its semantics, please refer to [22].

5. Analysis of the DTN model

In this section we explain how the asCSL logic can be applied to analyze the DTN model proposed in Section 3. In particular, in Section 5.1 we explain how to build an ASMC from the DTN definition. In Section 5.2 we will see how to write asCSL programs to analyze routing in the network. The main performance metrics of interest are defined in Section 5.3, while some details on the model solution technique and its complexity are provided in Section 5.4.

5.1. Deriving the Action and State-labelled Markov Chain for the DTN model

The state space. The state space of the ASMC is defined as the cartesian product of the state spaces of the Markov chains m_i describing the mobile nodes:

$$\mathcal{S} = \mathcal{S}_1 \times \mathcal{S}_2 \times \cdots \times \mathcal{S}_N \tag{1}$$

Action synchronization. For our particular application, we do not need the action synchronization capabilities of asCSL. Therefore, the set of action labels reduces to a single element α used to label all of the transitions:

$$\mathsf{Act} = \{\alpha\} \tag{2}$$

The transition matrix. The infinitesimal generator of the Markov chain of the asCSL model, is defined as the Kronecker sum of the infinitesimal generators of the Markov chains describing the nodes' motion:

$$\boldsymbol{Q} = \bigoplus_{i=1}^{N} \boldsymbol{Q}_i \tag{3}$$

Since there is a single generic action α, matrix $\boldsymbol{Q}$ completely specifies the rate matrix $\boldsymbol{R}$ of the ASMC:

$$\boldsymbol{R}(s_i, \alpha, s_j) = \boldsymbol{Q}(s_i, s_j) \tag{4}$$

The atomic propositions. The set of atomic propositions is a subset of the cartesian product of the set of mobile nodes $\mathcal{M}$ and the set of islands $\mathcal{I}$:

$$\mathsf{AP} \subseteq \mathcal{M} \times \mathcal{I} \tag{5}$$

Each element of AP represents the fact that a mobile node m_i occupies a position which is in the communication range of an island a_j. In order to facilitate the reader, we write an atomic proposition as: $[m_i @ a_j]$. We also extend the notation to sets, defining $[m_i @ \{a_1, \ldots, a_k\}] \equiv \{[m_i @ a_1], \ldots, [m_i @ a_k]\}$, to express the fact that a mobile node is concurrently in the communication range of all islands $a_1, \ldots, a_k$.

The labelling function. The labelling function provides for each state of the ASMC the set of atomic propositions that are currently associated to the mobile nodes. In particular, it associates to each state the propositions that define which islands are in communication range with the mobile nodes. Formally,

$$L(s_1, \ldots, s_n) = \bigcup_i [m_i @ f_P(l_i(s_i))]. \tag{6}$$

In the example of Figure 1, we have: $L(s_{1:0}, s_{2:0}) = \{[m_1 @ \{A, C\}], [m_2 @ C]\}$

5.2. Writing asCSL programs to analyze routing in the DTN model

The main purpose of our model is to compute the total end-to-end delay to transfer a data bundle from a node belonging to an arbitrary island a_i to a node belonging to an arbitrary island a_j, where $a_j \neq a_j$. In this section we examine how to write asCSL programs to take into account various routing strategies. We first consider the simple case in which the data bundle to be transferred from a_i to a_j is constrained to follow a unique, predetermined path (e.g., source routing); moreover, we initially assume that direct communication between two islands occurs through a single mobile node. Then we extend the analysis to multiple routes and to direct communication through multiple mobile nodes.

5.2.1. Basic routing

We use the notation $(a_i \rightarrow_{m_k} a_j)$ to identify a communication hop taking place from island a_i to island a_j, using mobile node m_k as relay. The communication hop is feasible only if both islands a_i and a_j comes in contact with node m_k, i.e., if $\exists s_{k:u}, s_{k:v} \in \mathcal{S}_k : a_i \in f_P(l_i(s_{k:u})) \wedge a_k \in f_P(l_i(s_{k:v}))$.

We first define $\mathsf{Hop}(a_i \rightarrow_{m_k} a_j)$ as the asCSL program that verifies a communication hop between islands a_i and a_j, using mobile node m_k as relay:

$$\mathsf{Hop}(a_i \rightarrow_{m_k} a_j) = (\neg[m_k@a_i]\mathsf{Act})^*\,;[m_k@a_i]\surd;\ (\neg[m_k@a_j]\mathsf{Act})^*\,;[m_k@a_j]\surd \quad (7)$$

The expression in (7) can be interpreted as follows: first, the source node in island a_i waits until mobile node m_k arrives within its transmission range ($(\neg[m_k@a_i]\mathsf{Act})^*$). Then, it immediately forwards the bundle to the mobile node ($[m_k@a_i]\surd$). If necessary, the mobile node m_k travels around the network area ($(\neg[m_k@a_j]\mathsf{Act})^*$), until it reaches a position in the communication range of the destination island, where it finally delivers the bundle ($[m_k@a_i]\surd$).

Note that, thanks to the pseudo-action $\surd$ of the asCSL logic (immediately executable), the formula in (7) also applies to the case of direct communication between islands a_i and a_k, using the classical *store-and-forward* approach, with mobile m_k acting as relay.

The same idea can be extended to a route. A route $(a_{i_1} \rightarrow_{m_{k_1}} a_{i_2} \rightarrow_{m_{k_2}} \cdots \rightarrow_{m_{k_{h-1}}} a_{i_h})$ is just a sequence of communication hops. It is feasible only if the single hops that compose it $(a_{i_1} \rightarrow_{m_{k_1}} a_{i_2}, a_{i_2} \rightarrow_{m_{k_2}} a_{i_3}, \ldots)$ are feasible. The program $\mathsf{Route}(\cdot)$ that verifies a route is simply given by the sequence of programs verifying each intermediate hop:

$$\mathsf{Route}(a_{i_1} \rightarrow_{m_{k_1}} a_{i_2} \rightarrow_{m_{k_2}} \cdots \rightarrow_{m_{k_{h-1}}} a_{i_h}) = \mathsf{Hop}(a_{i_1} \rightarrow_{m_{k_1}} a_{i_2});\cdots\mathsf{Hop}(a_{k_{h-1}} \rightarrow_{m_{k_{h-1}}} a_{i_h}) \quad (8)$$

As an example, in the scenario of Figure 1, the program that verifies the communication hop between island A and island D, using mobile node m_1, can be written as:

$$\mathsf{Hop}(A \rightarrow_{m_1} D) = (\neg[m_1@A]\mathsf{Act})^*\,;[m_1@A]\surd;\ (\neg[m_1@D]\mathsf{Act})^*\,;[m_1@D]\surd$$

The route $B \rightarrow_{m_1} C \rightarrow_{m_2} E$ can be verified as:

$$\mathsf{Route}(B \rightarrow_{m_1} C \rightarrow_{m_2} E) = (\neg[m_1@B]\mathsf{Act})^*\,;[m_1@B]\surd;(\neg[m_1@C]\mathsf{Act})^*\,;[m_1@C]\surd;\ (\neg[m_2@C]\mathsf{Act})^*\,;[m_2@C]\surd;(\neg[m_2@E]\mathsf{Act})^*\,;[m_2@E]\surd$$

5.2.2. Opportunistic routing

As already mentioned, a data bundle can be delivered to the destination using different mobile nodes as relays. For example, in Figure 1, consider a node in island C, wishing to communicate to a node in island D. Feasible alternatives are $C \rightarrow_{m_1} D$ and $C \rightarrow_{m_2} D$. According to the basic scheme, a node has to choose one of the two alternative routes at the beginning, and use that route regardless of the network state. For example, if mobile m_1 comes in contact with island C before mobile m_2, but the node has chosen the path $C \rightarrow_{m_2} D$ to deliver its message, it will have to wait for node m_2 anyway.

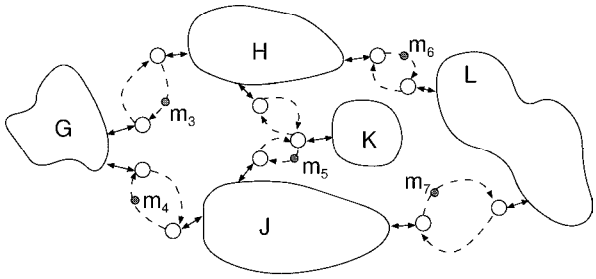

Figure 2. A more complex network scenario

With opportunistic routing, the node in C may choose to forward the packet to the first mobile node it comes in contact with, and rely on it to eventually deliver the packet to the destination. We express this possibility with the following notation:

$$C(\rightarrow_{m_1} \bigvee \rightarrow_{m_2})D \quad (9)$$

Notice that, in the above example, the sequence of islands traversed by the message is the same, i.e., (C, D). In general, the sequence of intermediate islands traversed by a bundle can vary, and the choice of a given mobile node as relay can affect the choice of mobiles nodes to be used in successive hops. For example, in figure 2, a node in G can communicate with a node in L either using the route $(G \rightarrow_{m_3} H \rightarrow_{m_6} L)$ or the route $(G \rightarrow_{m_4} J \rightarrow_{m_7} L)$. In this case, opportunistic routing selects mobile m_3 or mobile m_4, whichever arrives first, and this choice determines the next mobile node to use. We represent this case with the notation:

$$G(\rightarrow_{m_3} H \rightarrow_{m_6} \bigvee \rightarrow_{m_4} J \rightarrow_{m_7})L$$

As a last variation of opportunistic routing, consider in Figure 2 a node in island K , wishing to communicate with a node in island G. Since island K is only reached by mobile node m_5, it can only forward the bundle to m_5. However, m_5 may either dispatch the bundle to island H or to island J, whichever it reaches first. We denote this case with the notation:

$$K \rightarrow_{m_5} (H \rightarrow_{m_3} \bigvee J \rightarrow_{m_4})G$$

The path choice operator is directly mapped on the asCSL choice operator ($\cup$). For example:

$$\mathsf{Route}\left(C(\rightarrow_{m_1} \bigvee \rightarrow_{m_2})D\right) = (\neg([m_1@C] \vee [m_2@C])\mathsf{Act})^*\,; \left([m_1@C]\surd;(\neg[m_1@D]\mathsf{Act})^*\,;[m_1@D]\surd \ \cup\ [m_2@C]\surd;(\neg[m_2@D])\mathsf{Act})^*\,;[m_2@D]\surd\right)$$

The previous asCSL program can be explained as follows: the node in island C waits for either mobile m_1 or mobile m_2 to come, and sends the bundle to the first node that arrives. The next propositions to be checked depend on which mobile node has been used. Expressions for the other cases of opportunistic routing considered above can be derived in a similar way.

5.2.3. Direct Connection using multiple mobiles

Two islands can communicate directly using a sequence of mobiles nodes, instead of a single mobile node. Our analysis can be extended to account also for this case. To do so, we introduce *virtual* islands in between two successive mobile nodes on the direct path connecting two islands. Indeed, *virtual* islands are just an artifact to reduce ourselves to the basic case considered in Section 5.2.1. A bundle can traverse a *virtual* island, but cannot stop on it waiting for a mobile node to pick it up. As an example, in Figure 1, imagine that island D is virtual. This means that island B can communicate directly with island F through the sequence of mobile node (m_1, m_2) (as if island D would not exist). This communication can take place only at a time instant in which position P_2 is occupied by m_1, and at the same time P_5 is occupied by mobile m_2. We represent the direct hop between B and F with the notation

$$B \rightarrow_{m_1} \rightarrow_{m_2} F$$

Having introduced the virtual island D, the program that verifies the direct hop above can be written as follows:

$$\begin{array}{l}\mathsf{Hop}(B \rightarrow_{m_1} \rightarrow_{m_2} F) = \\ (\neg([m_1@B] \wedge [m_1@D] \wedge [m_2@D] \wedge [m_2@F]\mathsf{Act})^* ; \\ \quad ([m_1@B] \wedge [m_1@D] \wedge [m_2@D] \wedge [m_2@F])\surd\end{array}$$

Note that, although this procedure can be applied in general, and combined with the opportunistic routing introduced in Section 5.2.2, it requires the enumeration of all possible paths connecting two islands, which can be unfeasible in the presence of a large number of positions or mobile nodes through which two islands can communicate. However, many applications of DTN usually comprise only a small number of mobile nodes.

5.3. Performance metrics

Various performance indices can be computed by the proposed model. In this section we will show how to compute four interesting metrics.

5.3.1. Immediate path availability. In some cases, a communication between two adjacent islands can take place immediately (i.e. without having to wait for a mobile), if there exist a direct path available. For example, island B can directly communicate to island D in Figure 1, when mobile m_1 is at position P_2. Island B can also directly communicate with island F if mobile m_1 is at position P_2, and mobile m_2 is jointly at position P_5. The availability of a direct path can be computed using the steady state operator $\mathcal{S}_{\bowtie p}(\phi)$. In particular, a route $(a_{i_1} \rightarrow_{m_{k_1}} a_{i_2} \rightarrow_{m_{k_2}} \cdots \rightarrow_{m_{k_{h-1}}} a_{i_h})$ can be traversed without delay, with probability $\geq p$, if the following formula is verified:

$$\mathcal{S}_{\geq p}\left(\bigwedge_{j=1}^{h-1}\left([m_{k_j}@a_{i_j}] \wedge [m_{k_j}@a_{i_{j+1}}]\right)\right) \qquad (10)$$

5.3.2. End-to-end delay distribution. The end-to-end delay distribution can be derived by testing if a message can reach its destination before a given deadline with a certain probability. This can be done using the $\mathcal{P}_{\bowtie p(\alpha^I)}$ operator. In particular, a message can be delivered through route $(a_{i_1} \rightarrow_{m_{k_1}} a_{i_2} \rightarrow_{m_{k_2}} \cdots \rightarrow_{m_{k_{h-1}}} a_{i_h})$ before a given deadline t with a probability $\geq p$, if the following formula is verified:

$$\mathcal{P}_{\geq p}\left(\mathsf{Route}(a_{i_1} \rightarrow_{m_{k_1}} a_{i_2} \rightarrow_{m_{k_2}} \cdots \rightarrow_{m_{k_{h-1}}} a_{i_h})^{[0,t]}\right) \qquad (11)$$

5.3.3. Distribution of the time to catch a mobile. Another interesting performance index is the distribution of the time spent by a message at an island, waiting to be picked up by a mobile node. This is an important performance measure, since it can be used to size the buffers of the fixed nodes. In particular, the probability that a message at island a_i waits for mobile m_k to arrive for less than t time units, is $\geq p$ if:

$$\mathcal{P}_{\geq p}\left((\neg[m_k@a_i]\mathsf{Act})^* ; [m_k@a_i]\surd\right)^{[0,t]}\Big) \qquad (12)$$

5.3.4. Distribution of the sojourn time on a mobile. In a similar way, we can compute the distribution of the time spent by a message on a given mobile. This performance measure can be used to properly dimension the buffer size of the mobile nodes. In particular, the probability that a message starting from island a_i and directed to island a_j, spends less than t time units on a mobile m_k, is $\geq p$ if:

$$\mathcal{P}_{\geq p}\left(([m_k@a_i]\surd ; (\neg[m_k@a_j]\mathsf{Act})^* ; [m_k@a_j]\surd)^{[0,t]}\right) \qquad (13)$$

5.4. State space reduction

The proposed model can lead to ASMC with very large state space. In particular, the size of the ASMC state space is the product of the sizes of the state spaces describing the paths of the mobile nodes. Following the procedure defined in [22], the product Markov chain must be multiplied by the nondeterministic program automata (*NPA*) that accepts the regular expression corresponding to the asCSL program. This is shown in Figure 3(a), for the case of route $(A \rightarrow_{m_1} C \rightarrow_{m_2} E)$ in Figure 1. However, in the basic routing case, in each state of the NPA formulas that label the outgoing arcs refer to a single mobile only, as emphasized in Figure 3(a) by the dashed boxes. We can exploit this fact and the independence of nodes' movement,

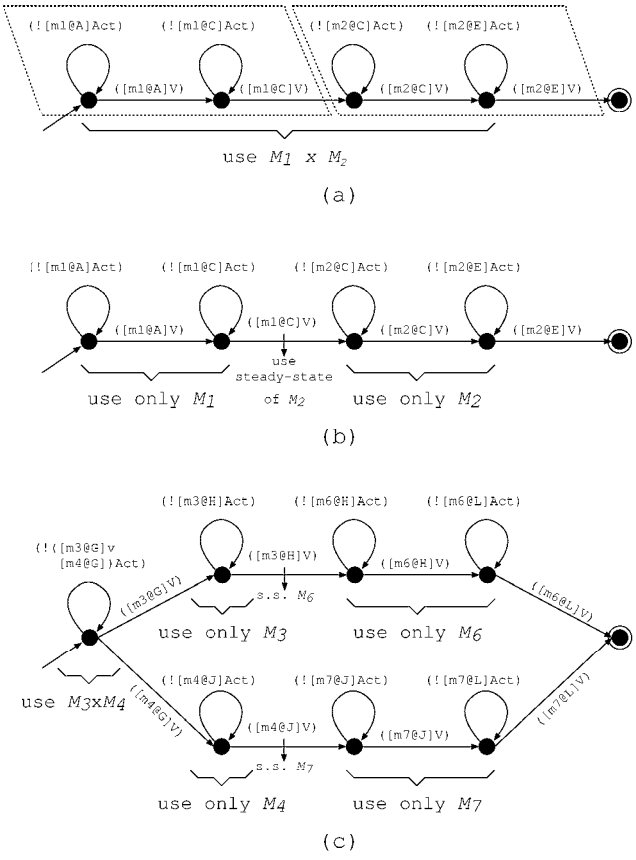

Figure 3. Reducing the state space

to reduce the state space. In particular we can consider in each state of the NPA, only the Markov chain corresponding to the mobile node appearing in the labels of the outgoing arcs. This is shown in Figure 3(b). When we transit from a state where formulas are defined for mobile m_k to a state where formulas are defined for mobile m_l, we use the steady state distribution of the Markov chain corresponding to mobile m_l, to correctly define the outgoing transitions of the product of the ASMC and the NPA. With this reduction, the complexity of the proposed approach is linear in the number of hops. This allows the application of the proposed technique to large networks comprising many independent mobile nodes.

With similar reasoning, the reduction technique can be extended to opportunistic routing, as shown in Figure 3(c) for the route $G(\rightarrow_{m_3} H \rightarrow_{m_6} \bigvee \rightarrow_{m_4} J \rightarrow_{m_7})L$. In this case, in the states where the NPA has to choose the route, we must consider the product of all Markov chains defining the path of the alternate mobile nodes that can be used. However, usually the number of alternatives is small, and in all of the other NPA states, only a single Markov chain is sufficient to define the NPA x ASMC process.

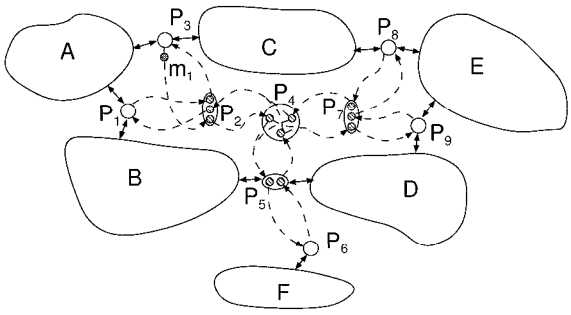

Figure 4. Network scenario in case of single mobile with a complex path

previous loop \ next loop	L_1	L_2	L_3
L_1	–	0.8	0.2
L_2	0.5	–	0.5
L_3	0.5	0.5	–

Figure 5. Transition matrix between the loops of mobile m_1 in Figure 4

6. Case Studies

In this section we present a number of case-studies to illustrate the potentialities of the proposed model.

6.1. Single Mobile with a complex path

We start considering the case of a single mobile node following a complex path in which it gets in contact with several disconnected islands. The topology, depicted in Figure 4, comprises 6 islands $(A, \ldots, F)$ and 9 possible positions $(1, \ldots, 9)$ occupied by the mobile node m_1.

The route of the mobile node includes 3 loops departing from central node P_4:

- loop L_1: $P_4 - P_2 - P_1 - P_2 - P_3 - P_2 - P_4$
- loop L_2: $P_4 - P_7 - P_8 - P_7 - P_9 - P_7 - P_4$
- loop L_3: $P_4 - P_5 - P_6 - P_5 - P_4$

Upon completion of a loop, the mobile node goes back to position P_4 and enters a new loop according to the loop transition matrix reported in Figure 5. The ASMC describing this pseudo-random motion requires 16 states. The sojourn time at all positions is set equal to 1 time unit.

Figure 6 reports all end-to-end delay distributions of a message originated at island A and destined to one of

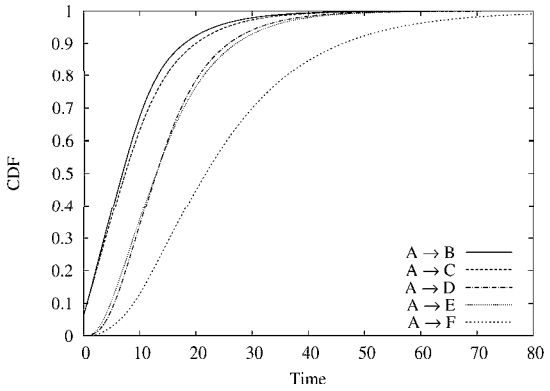

Figure 6. End-to-end delay distribution from island A to all other islands

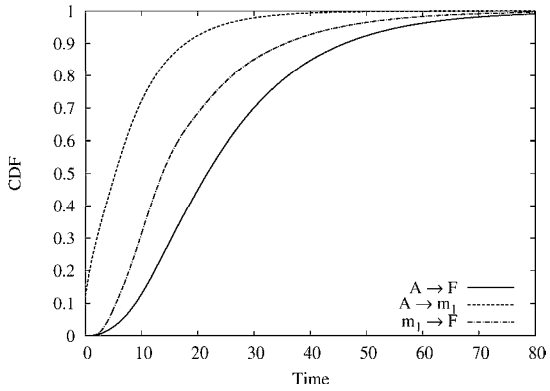

Figure 7. Contributions to the overall delay of route $A \rightarrow_{m_1} F$: time to catch the mobile (labelled $A \rightarrow_{m_1}$) and time spent on mobile (labelled $m_1 \rightarrow_F$).

other islands. We observe that the curves referring to routes $A \rightarrow_{m_1} B$ and $A \rightarrow_{m_1} C$ departs from a non-zero value. This is due to the fact that a direct path with zero delay can be immediately available between island A and islands B or C, in case the mobile node is already at positions P_1 or P_3, respectively. Indeed, for these two routes the *immediate path availability* as computed in Section 5.3.1 is just equal to the stationary probability of finding the mobile at P_1 or P_3, which is 0.061. Note that, at any time t, the probability to have reached island B from island A is larger than the probability to have reached island C. This because a message from A can be delivered to B using *store-carry-and-forward* by moving on the mobile node at P_3, and arriving at B from P_5, before the establishment of the direct link through P_1. Instead, A can reach C only upon establishment of the direct link through P_3, thus *store-carry-and-forward* does not help in this case. We also observe that route $A \rightarrow_{m_1} F$ has significant larger delay than other routes, due to the fact that loop L_3 is selected with lower probability than loop L_2 by a mobile finishing loop L_1. For route $A \rightarrow_{m_1} F$, we have computed also the two contributions to the overall delay related to the time spent at island A waiting for the mobile to come (Section 5.3.3), and to the time spent on mobile m_1 (Section 5.3.4). Results are reported on Figure 7. We observe that the time spent by the message while being carried on mobile m_1 accounts for the greater part of the overall delay.

6.2. Multiple mobiles and the impact of opportunistic routing

We now consider the scenario illustrated in Figure 1, and study the impact of multiple mobile nodes and multiple routes selected according to opportunistic routing. In this scenario, mobile m_1 performs the loop $P_0 - P_1 - P_2 - P_1 - P_0$ (this requires 4 states in the underlying ASMC), and the mean sojourn time at each position is 1 time unit. Similarly, m_2 cycles around $P_3 - P_4 - P_5 - P_4 - P_3$ with mean sojourn time of 1 time unit at each position.

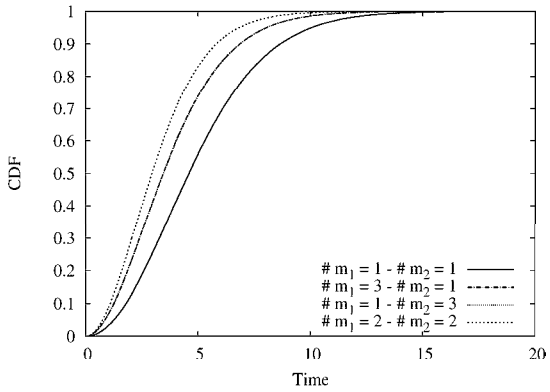

Figure 8. End-to-end delay distribution of route $A \rightarrow_{m_1} D \rightarrow_{m_2} F$ with a variable numbers of nodes of type m_1 and m_2

First, we consider the (unique) route $A \rightarrow_{m_1} D \rightarrow_{m_2} F$. Figure 8 reports the end-to-end delay of this route for various combinations of the number of mobile nodes of type m_1 and of type m_2. This means that there can be multiple (independent) instances of mobile nodes following the same route, and, of course, the message jumps on the first one that arrives.

As expected, the end-to-end delay improves while increasing the number of mobile nodes, because this reduces the time to catch the mobile. For the same total number of mobile nodes, namely 4, the best performance is obtained assigning an equal number of nodes to type m_1 and to type

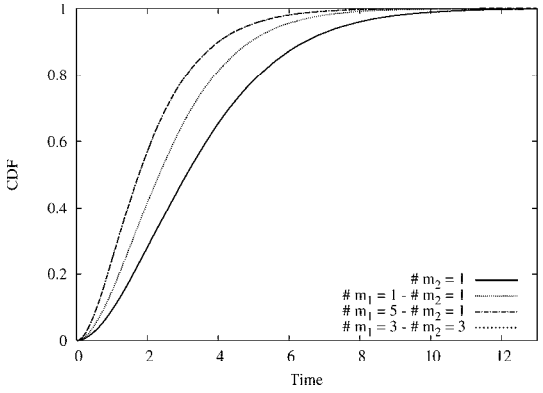

Figure 9. End-to-end delay distribution from island C to island D in four different cases

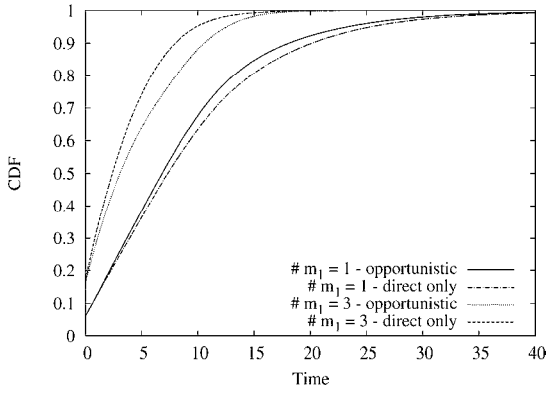

Figure 10. End-to-end delay distribution from island A to island B with of without opportunistic routing, for different number of mobile nodes

m_2 (2 nodes each). The combinations $(3,1)$ and $(1,3)$, instead, perform the same, and result into a higher delay because of the diminishing return obtained putting all nodes on the same route, instead of distributing them on both.

Next we consider the opportunistic routing case expressed by (9), i.e., the case of a message having two alternative routes to go from island C to island D in Figure 1, being carried either by mobile m_1 or by mobile m_2. Figure 9 reports several curves of the end-to-end delay from C to D in this scenario. Curve labelled "$\#m_2 = 1$" corresponds to the case in which a single route is available, using m_2. The curve labelled "$\#m_1 = 1$ - $\#m_2 = 1$" corresponds to the case in which both are available, and shows the gain achievable by opportunistic routing. The other two curves refer to opportunistic routing with multiple nodes of type m_1 and m_2, and show that, in this particular case, the way we partition the nodes among classes m_1 and m_2 is not important. Indeed, the combination $(5,1)$ achieves the same performance of $(3,3)$.

It is important to note that the benefit resulting from opportunistic routing strongly depends on the considered network scenario, and it is also affected by the number of mobile nodes. In some cases, opportunistic routing can even be worse than single-path routing. To show this fact, we use again the topology in Figure 4, and consider two different strategies for routing a message from island A to island B. One strategy uses only the direct path made available when a mobile node arrives at position P_1. According to the second strategy, the message is moved on the first mobile that comes in contact with island A, either at position P_1 or at position P_3. Actually, this is the same strategy already considered in Section 6.1, and can be regarded as a form of opportunistic routing. Figure 10 shows the performance achieved by the two strategies above when the number of mobiles nodes is either 1 or 3. We observe that, in case of a single mobile node, the opportunistic strategy outperforms the single-route strategy. In case of 3 mobile nodes, it is the opposite: the single-route is better than the opportunistic strategy. This can be explained by the fact that, with many mobile nodes, it is better to wait til the direct path through P_1 becomes available, instead of moving on the mobile at P_3, and possibly making a long trip around the network before arriving in proximity of island B.

6.3. When is *store-carry-and-forward* the right choice ?

Since the *store-carry-and-forward* paradigm incurs very long delays due to the time waited while being carried on the mobile node(s), it can be argued that, unless this is the only alternative, it may not be the best way to communicate between disconnected network components. Indeed, if there is a sufficiently large number of mobiles nodes, one can simply wait for a direct path to form, and then transfer the message in negligible time over this path. Our model is able to solve this interesting dilemma as the number of mobile nodes vary. Figure 11 reports the average transfer delay (not the distribution) of routes $A \rightarrow F$ and $A \rightarrow D$ in the scenario in Figure 4, for two routing strategies: curves labelled *store-carry-and-forward* refer to the case in which the message jumps on the first node that comes, and waits til the same node arrives in proximity of the destination; curves labelled *store-and-forward* refers to the case the message waits til a direct path forms between source and destination. We observe that, if the number of nodes is small, *store-carry-and-forward* is the only strategy that can actually deliver the message, because there are not enough nodes to form a direct path (the minimum is 4 for $A \rightarrow D$, 5 for $A \rightarrow F$). As we increase the number of nodes, the *store-and-forward* strategy soon becomes optimal, while the performance of the *store-carry-and-forward* strategy is only marginally improved by the larger number of carriers. Notice that, for a given number of mobile nodes, the best alter-

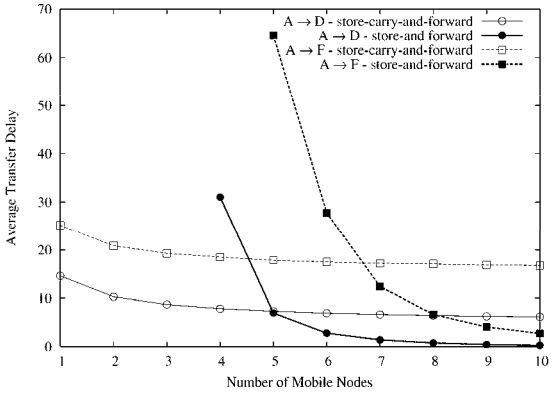

Figure 11. Comparison of average transfer delay between *store-carry-and-forward* and *store-and-forward* for two different routes of the topology in Figure 4

native may depend on the route. For example, in case of 6 mobile nodes, waiting for a direct path is optimal for route $A \rightarrow D$, whereas for route $A \rightarrow F$ the *store-carry-and-forward* strategy is better.

7. Conclusions and Future Work

In this paper we have introduced a general model of a delay-tolerant network comprising a mixture of fixed and mobile nodes. We have represented the motion of mobile nodes through independent state-labelled Markov chains, with the possibility of considering different degrees of randomness in the mobility pattern of nodes. We have explained how to define and verify by asCSL specific performance indices of interest to delay tolerant networking, such as the end-to-end delay distribution between any pair of nodes. Our analysis allows to evaluate the impact of opportunistic forwarding strategies, as well as to predict whether the *store-carry-and-forward* scheme is indeed preferable with respect to traditional routing, as we vary the number of mobile nodes. Our delay analysis has many interesting networking applications. For example, given a traffic matrix, it can be used to predict the storage requirement at intermediate nodes that prevents data loss due to buffer overflow. Conversely, for a given amount of storage in the network, it can be used to predict the allowable sending rate of the sources, and thus evaluate the network capacity in terms of throughput.

References

[1] Delay Tolerant Network Research Group (DTNRG). Website: `www.dtnrg.org`

[2] S. Burleigh et. al.,"Delay-Tolerant Networking: An Approach to Interplanetary Internet", *IEEE Communications Magazine*, pp. 128–136, June 2003.

[3] V. Cerf et al., "Delay tolerant network architecture", `draft-irtf-dtnrg-arch-04.txt`, December 2005.

[4] K. Scott, S. Burleigh, "Bundle Protocol Specification", `draft-irtf-dtnrg-bundle-spec-04.txt`, December 2005

[5] K. Fall, "A Delay-Tolerant Network Architecture for Challenged Internets," *In Proc. ACM SIGCOMM*, 2003.

[6] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network", *In Proc. ACM SIGCOMM*, 2004.

[7] S. Jain, M. Demmer, R. Patra, K. Fall, "Using redundancy to cope with failures in a delay tolerant network", *In Proc. ACM SIGCOMM*, 2005.

[8] R. C. Shah, S. Roy, S. Jain and W. Brunette, "Data MULEs: Modeling and analysis of a three-tier architecture for sparse sensor networks", *Elsevier Ad Hoc Networks Journal*, Vol. 1, No. 2-3, pp. 215–233, Sept. 2003.

[9] W. Zhao, M. Ammar, and E. Zegura, "A message ferrying approach for data delivery in sparse mobile ad hoc networks", *In Proc. MobiHoc04*, 2004.

[10] P. Juang, H. Oki, Y. Wang, M. Martonosi, L.–S. Peh, D. Rubenstein, "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet", *in Proc. ASPLOS-X*, Oct. 2002, San Jose, CA.

[11] Wizzy Project. `http://www.wizzy.org.za`

[12] A. Pentland, R. Fletcher, and A. Hasson, "Daknet: rethinking connectivity in developing nations", *IEEE Computer*, 37:78–83, 2004.

[13] J. Burgess, B. Gallagher, D. Jensen, B. N. Levine, "MaxProp: Routing for Vehicle-Based Disruption-Tolerant Networking", *in Proc. IEEE INFOCOM 2006*, Barcelona, Spain, April 2006.

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

[15] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks", *Mobile Computing and Communications Review*, 7(3):19–20, 2003.

[16] T. Small and Z. J. Haas, "Resource and performance tradeoffs in delay-tolerant wireless networks,". *in Proc. ACM workshop on Delay Tolerant Networking*, August 2005.

[17] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks", *In Proc. ACM workshop on Delay-tolerant networking*, August 2005.

[18] X. Zhang, G. Neglia, J. Kurose, D. Towsley, "On the Benefits of Random Linear Coding for Unicast Applications in Disruption Tolerant Networks", *IEEE NETCOD Workshop*, April 2006.

[19] J. Widmer, J.-Y. Le Boudec, "Network Coding for Efficient Communication in Extreme Networks", *In Proc. ACM workshop on Delay-tolerant networking*, August 2005.

[20] Y. Ko, N. H. Vaidya, "Location-Aided Routing (LAR) Mobile Ad Hoc Networks", *In Proc. ACM/IEEE MobiCom '98*, Dallas, TX, October 1998.

[21] A. Spyropoulos, K. Psounis, C. Raghavendra, "Performance Analysis of Mobility-assisted Routing," to appear at *ACM MobiHoc '06*, Florence, Italy, May 2006.

[22] C. Baier, L. Clotj, B. Haverkort, M. Kuntz, M. Siegle, "Model Checking Action- and State-Labelled Markov Chains", *In Proc. of International Conference on Dependable Systems and Networks DSN '04*, June 2004

[23] A. Aziz, K. Samwal, V. Singhal, R. Brayton, "Verifying continuous time Markov chains", *In CAV'96, LNCS 1102*, 1996.

[24] C. Baier, J.-P. Katoen, H. Hermanns, "Approximate symbolic model checking of continuous-time Markov chains", *In CONCUR '99, LNCS 1664*, 1999.