A Bayesian Approach for Disconnection Management in Mobile Ad-hoc Networks

Massimiliano de Leoni, Shah Rukh Humayoun, Massimo Mecella, Ruggero Russo
Dipartimento di Informatica e Sistemistica
SAPIENZA – Università di Roma
Via Ariosto, 25 - Roma (Italy)
{deleoni,humayoun,mecella,russo}@dis.uniroma1.it

Abstract

Mobile Ad-hoc Networks are used in many scenarios (e.g., emergency management) for supporting collaborative work of operators. But this requires either ***(i)*** continuous connections, or at least ***(ii)*** the possibility to foresee that a device is going out and disconnecting. Therefore a basic problem is how to predict the possible disconnections of devices, in order to let the upper layers appropriately address connection anomalies (e.g., either taking global remedial actions to maintain the network connected, or local ones to let the disconnecting device to go on for some time with its own work, e.g., cache important data needed for the following activities). In this paper we present a bayesian approach to predict disconnections in MANETs, and validating experimental results that show the viability of the approach.

Keywords: disconnection prediction, mobile ad hoc networks, cooperative work.

1 Introduction

A Mobile Ad hoc NETwork (MANETs) is a peer-to-peer (P2P) network of mobile nodes capable to communicate with each other without an underlying infrastructure. Nodes can communicate with their own neighbors (i.e., nodes in radio-range) directly by wireless links. Anyway, non-neighbor nodes can anyway communicate by using other intermediate nodes as relays, which forward packets toward destinations. The lack of a fixed infrastructure makes this kind of network suitable in all scenarios where it is needed to deploy quickly a network but the presence of access points is not guaranteed. Examples are military applications, and more recently, pervasive systems for process management [13].

In order to guarantee the coordination and data exchange the nodes should be aware of the moment when they are going to disconnect from their MANETs (i.e., there is not any neighbor – node in the radio range – to which node is connected). Indeed, if nodes are alerted about probable disconnections, some remedial actions can be enacted:

local disconnection management – That is the node can take autonomous – not requiring coordination with other nodes – remedial actions, e.g., to cache some data that are needed for other tasks to be carried out while disconnected, or to disseminate to other nodes the critical information it is storing that is crucial for the other nodes (so to let them working);

global disconnection management – where the nodes together coordinate, e.g., through an appropriate Process Management System [6] to be hosted on a "leader device". Such a device acts as a central server and rearranges nodes in a new configuration in which none is going to disconnect.

This paper introduces a new technique that permits to predict when MANET nodes are going to disconnect from all of others and to become isolated. According to the classification proposed in [17, 18], coordination can be viewed as being divided into three layers, each depending on those below: ***(i)*** communication, ***(ii)*** collaboration and ***(iii)*** coordination. The lowest layer allows information sharing; collaboration permits participants to collectively establish the shared goals; the latter ensures to enact collaborative actions to achieve shared goals as efficiently as possible.

According to this classification, predicting disconnections can be categorized as part of the communication layer. Indeed, without predicting and handling disconnections, MANET communication would not work and, consequently, collaboration and coordination would be impossible

The rest of the paper is structured as follows. This section ends by illustrating an example about global

disconnection management where disconnection prediction may make sense.

Section 2 illustrates our approach: after sketching the assumptions and some basis of Bayesian filtering, it continues explaining how to predict when links fall down. Finally, it ends by describing the algorithm for building an estimated next *connection graph*. Such a graph illustrates estimated MANET topology in the close future: nodes are MANET peers and arcs represent links that are estimated as active.

Section 3 describes OCTOPUS, the Virtual Environment that has been used as emulation platform during testing. Section 4 shows some technical details of the actual implementation where Section 5 describes the experiments and following results. The paper concludes with Section 7 where we report final comments in the light of results and discuss future progression of this work.

1.1 A Motivating Example

We are currently working on an European-funded project, namely WORKPAD [3], that is in charge of designing and implementing a 2-level software infrastructure to support cooperative works in disaster management scenarios. In complex emergency scenarios, different teams, belonging to different organizations, need to collaborate. So, there exist the need of inter-team and intra-team coordination. The front-end level consists of several teams, whereas the back-end is formed by the headquarters. A single front-end community is constituted by the operators of a team, equipped with mobile devices, connected in an ad-hoc and peer-to-peer fashion, that carry on a process, in which the adaptiveness to connection/task anomalies is fundamental. Operators may be humans or robots; therefore, teams may be mixed.

Every human team member is equipped with hand-held devices (PDAs) and communication technologies, and is in charge of specific tasks. Team includes also special robots, e.g., UAVs - Unmanned Aerial Vehicles, that are intended only to ensure connectivity among nodes, working as relays. In such a way we can see the whole team as carrying on a process, and the different teams (of the different organizations) collaborate through the exchange and integration of data. Different data sets are stored in the headquarters of organizations which are responsible for them.

Front-end operators access to the corresponding back-end server located at headquarters through private defined protocols. Here they can get or set information which is relevant to the situation they are dealing with. At the same time, operators can query information provided by other organizations. This feature is provided by an underlying P2P network. Indeed, each and every headquarter exports data and their schema. Then, all these data sources are integrated in order to create a single "virtual" data source. It does not make difference to clients which servers actually provide those data.

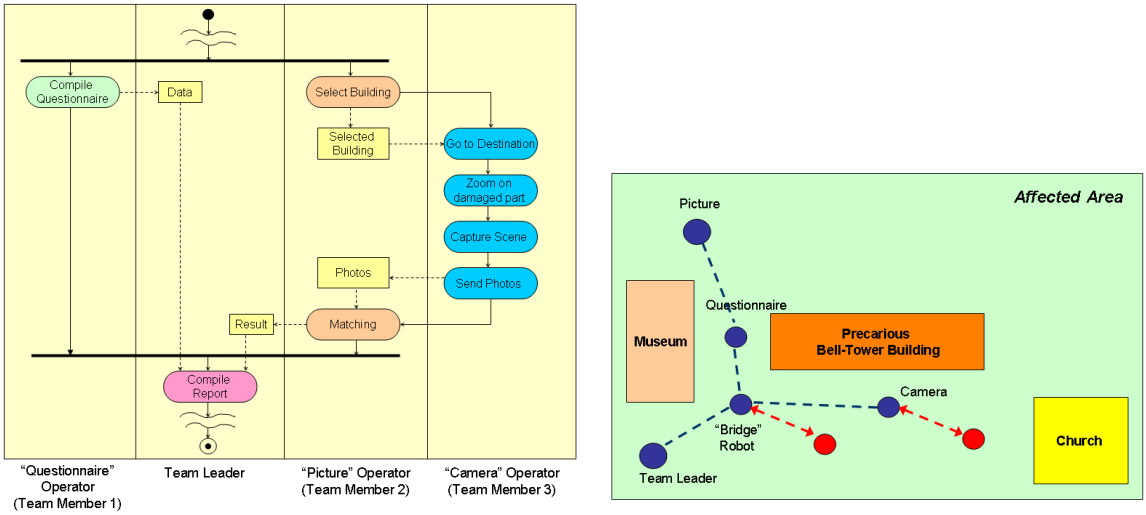
As a more concrete example, consider a scenario of archeological disaster/recovery: after an earthquake, a team is sent to the hit area to evaluate the state of archeological sites and of precarious buildings; the goal is to draw a situation map to schedule restructuring jobs. A typical cooperative process to be enacted by the team would be as shown in Figure 1(a) (depicted as a UML Activity Diagram).

The team leader has got installed specific hardware/software to connect to the back end to retrieve all previously stored data details, including a map of the site, the list of the most sensible objects located in the site, and precedent reports/materials, as well as to update the information about the emergency situation.

The team is considered as an overall MANET, in which the team leader's device (requiring the most computational power, therefore usually a laptop) coordinates the other team members devices, by providing suitable information (e.g., maps, sensible objects, etc.) and assigning activities.

Team members are equipped with hand-held devices (PDAs), which allow them to execute some operations but do not have too much computational power. Such operations, possibly provided through the support of particular hardware (e.g., digital cameras, computational power for image processing, main storage, clients for Geographic Information System, etc.), are offered as software services to be coordinated. Team Member 1 (using its device) could compile some specific questionnaires (after a visual analysis of a building). This will be analyzed by the team leader to schedule next activities, supported by specific softwares. Team Member 3 could take some pictures of the precarious buildings. Finally Team Member 2 is in charge of specific image processing tasks on previous and recent pictures (e.g., for first identification of architectural anomalies).

In this scenario, Team Member 2 is in charge at a given point of matching the photos taken by Team Member 3 and the previous ones stored at back-end. PDAs are not so powerful that they can retrieve all photos and temporally store. So, Team Member 3 must be connected to the team leader during the whole task execution. Indeed, Team Leader provides a special hardware proxy allowing any member to connect to the back end. In theory, every member should not need to be connected, possibly through multi-hop paths, to any other member at all times. But applications running on member devices have to get ensured that a connection always



(a) Process

(b) Critical situation and adaptive management

Figure 1: Disconnection prediction for adaptive coordination in MANETs

exists between every couple of devices. They do not have to be worried about disconnections: the Coordination Layer will take care of them.

But in a particular scenario, as the one depicted in Figure 1(b), it could happen that the movement of the operator/device equipped with the camera would result in a disconnection from other devices/operators. The Team Leader device has got deployed a Prediction Layer being in charge of predicting disconnection. A Coordination Layer is arranged above and is continually ready to receive disconnection alarms from the layer below. It could decide to instruct a possible "bridge", a robot, to go after the disconnecting operator (i.e. his device), to maintain the connection and to ensure a path between all devices. Instructing a bridge device is just a way to handle disconnections: the coordination layer could consider more convenient, in some situations, to duplicate required data such that the going-to-disconnect node can be performing assigned task, even if disconnected.

The example described above is a typical *global disconnection management*. Indeed, actions for handling disconnections are centrally managed by the coordination layer of the leader device. The leader device is actually the "global entity" that has the knowledge about the status of all operators/devices, and takes into account idle devices, dependencies and operations that can and cannot be safely delayed, etc.

2 Bayesian Filtering for Disconnection Prediction

Our predictive technique is based on few assumptions:

1. Each device is equipped with specific hardware that allows it to know its *distance* from the surrounding connected (i.e., within radio range) devices. This is not a very strong assumption, as either devices are equipped with GPS or specific techniques and methods (e.g., TDOA - time difference of arrival, SNR - signal noise ratio, the Cricket compass, etc.) are easily available. Paper [16] presents a precise technique to track multiple wireless nodes simultaneously. It relies on measuring the position of tracked mobile nodes through radio interferometry. This is guaranteed to reduce significantly the error with respect to GPS. Nevertheless, the authors of paper [14] have recently devised techniques to mitigate the error when computing node position through GPS. Indeed, they performed experiments where the error has been reduced to 3 meters when nodes are not moving and to 20 meters when nodes are at 80 km/h.

2. At start-up, all devices are connected (i.e., for each device there is a path - possibly multi-hop - to any other device). The reader should note that we are not requiring that each device is within the radio range of (i.e., one hop connection to) any

other device (*tight* connection), but we require only a *loose* connection (guaranteed by appropriate routing protocols, e.g., DSR, AODV, etc.).

3. A specific device in MANET, referred to as *coordinator*, is in charge of centrally predicting disconnections. As all devices can communicate at start-up and the ultimate goal of our work is to maintain such connections through predictions, it is possible to collect centrally all the information from all devices;

The predictive technique is essentially as follows: at a given time instant t_i the coordinator device collects all distance information from other devices (for assumptions *(1)* and *(3)*); on the basis of such information, the coordinator builds a probable *connection graph* that is the probable graph at the next time instant t_{i+1} in which the predicted connected devices are highlighted. On the basis of such prediction, the coordinator layer will take appropriate actions (which are no further considered in the following of this paper).

2.1 Bayesian Filtering

Bayes filters [2] probabilistically estimate/predict the current state of the system from noisy observations. Bayes filters represent the state at time *t* by a random variable Θ_t. At each point in time, a probability distribution $Bel_t(\theta)$ over Θ_t, called *belief*, represents the uncertainty. Bayes filters aim to sequentially estimate such beliefs over the state space conditioned on all information contained in the sensor data. To illustrate, let's assume that the sensor data consists of a sequence of time-indexed sensor observations $z_1, z_2, ..., z_n$. The $Bel_i(\theta)$ is then defined by the posterior density over the random variable Θ_t conditioned on all sensor data available at time t:

$$Bel_t(\theta) = p(\theta|z_1, z_2, ...z_t) \quad (1)$$

Generally speaking, the complexity of computing such posterior density grows exponentially over time because the number of observations increases over time; the following two assumptions are needed for making the computation tractable:

1. The system's dynamic is markovian, i.e., the observations are statistically independent;

2. The devices are the only subjects that are capable to change the environment.

On the basis of the above two assumptions, the equation in a time instant *t* can be expressed as the combination of a prediction factor $Bel_{t-1}(\theta)$ (the equation in the previous time instant) and an update factor that on the basis of the observation in the time instant *t*, realizes the update of the prediction factor.

In our approach, the random variable Θ_t belongs to $[0, 1]$ and we use the Beta(α,β) function as a *belief* distribution to model the behavior of the system, according to the following equation:

$$Bel_t(\theta) = Beta(\alpha_t, \beta_t, \theta) \quad (2)$$

where α and β represent the state of the system and vary according to the following equations:

$$\begin{cases} \alpha_{t+1} = \alpha_t + z_t \\ \beta_{t+1} = \beta_t + z_t \end{cases} \quad (3)$$

In our approach, the observation z_t represents the variation of the relative distance between nodes *(i,j)* normalized with respect to radio range in the time period *[t-1,t]*. It is used to update the two parameters α and β of the Beta function according to Equation 3. The evaluated $Beta(\alpha, \beta)$ function predicts the value of $\theta_{t+1}^{(i,j)}$ estimating the relative distance that will be covered by the nodes *(i,j)* in the next time period *[t,t+1]*.

2.2 Prediction of the Distances

Our approach relies on clock cycles whose periods are T. The pseudo-code for the coordinator is described in Figure 2. We assume the *iBuffer* data structure to be stored only at Team Leader and accessed only by local threads in a synchronized way. For each ordered couple (i, j) of nodes, in the n-th cycle, the monitor stores two float parameters, $\alpha_n^{(i,j)}$ and $\beta_n^{(i,j)}$, and the last observed distance $d_{n-1}^{(i,j)}$.

Let us assume a node k comes in a MANET during the m-th clock cycle. Then, for each MANET node j we initialize $\alpha_m^{(k,j)} = \beta_m^{(k,j)} = 1$. In such a way we get the uniform distribution in $[0, 1]$ and, so, every distance $d_{m+1}^{(k,j)}$ gets the same probability.

For each time period T, each generic node i sends a set of tuples (i, j, d^j) to the coordinator, where j is an unique name of a neighboring node and d^j is the distance to j. The coordinator collects continuously such tuples (i, j, d^j) coming from the nodes in an intermediate buffer. We do no assumptions about clock synchronization. So, every node collects and sends information to Team Leader according to its clock, which is in general shifted with respect to the one of other nodes.

Monitor performs prediction according to the same clock T: at the beginning of the generic n-th clock cycle upon timer expiring, it copies the tuples (i, j, d_n^j) from the intermediate buffer to another one and, then, it empties the former buffer to get ready for updated values. In the clock cycle, for each collected tuple (i, j, d^j) monitor updates the parameters as follow by a bayesian filter:

timer: *a timer expiring each T seconds.*
iBuffer[x,y]: *a bi-dimensional squared matrix storing distance among couples of nodes X and Y.*
bayesianBuffer[x,y]: *a bi-dimensional square matrix storing a triple (α, β, distance) for each couple of nodes X and Y.*

```
UPON DELIVERING BY NODE I OF TUPLE(i, j, dist)
1  iBuffer[i, j] ← dist

UPON EXPIRING OF TIMER()
 1  localBuffer ← iBuffer[i, j]
 2  /*empty intermediate buffer*/
 3  for (i, j) ∈ ibuffer
 4  do ibuffer[i, j] ← RADIO_RANGE
 5
 6  for (i, j) ∈ localBuffer
 7  do if localBuffer[i, j] ← RADIO RANGE
 8        then observation ← 1
 9        else observation ← (localBuffer[i, j] − bayesianBuffer[i, j].distance)/RADIO RANGE
10             observation ← (observation + 1)/2
11     bayesianBuffer[i, j].distance ← localBuffer[i, j]
12     bayesianBuffer[i, j].alpha ← u ∗ bayesianBuffer[i, j].alpha + observation
13     bayesianBuffer[i, j].beta ← u ∗ bayesianBuffer[i, j].beta + (1 − observation)
```

Figure 2: Pseudo-codes of the Bayesian algorithm for predicting node distances.

$$\begin{cases} \alpha_{n+1}^{(i,j)} = u \cdot \alpha_n^{(i,j)} + o_n^{(i,j)} \\ \beta_{n+1}^{(i,j)} = u \cdot \beta_n^{(i,j)} + (1 - o_n^{(i,j)}) \end{cases} \tag{4}$$

where $o_n^{(i,j)}$ is an observation and $u \in [0,1]$ is a constant value. Constant u aims for permitting old observations to age. As new observations arrive, the previous gets less and less relevance. Indeed, old observations do not capture the updated status of MANET connectivity and motion.

The value for observation can be computed from the relative distance variation between i and j, scaled with radio-range:

$$\Delta dr_n^{(i,j)} = \frac{d_n^{(i,j)} - d_{n-1}^{(i,j)}}{radio\_range} \tag{5}$$

where $radio\_range$ is the maximum distance from where two nodes can communicate with each other.

Possibly $d_n^{(i,j)}$ can miss in the cycle n. The distance between i and j could miss because i and j are not in radio-range or packets sent by i to Team Leader are lost or delivered lately.

It is straightforward to prove $\Delta dr_n^{(i,j)}$ to range in [-1, 1] interval. This range is not suitable for Bayesian filter since observations should be between 0 and 1. So we map the value in Equation 5 into the suitable range [0, 1] as follows[1]:

$$o_n^{(i,j)} = \begin{cases} \frac{d_n^{(i,j)} - d_{n-1}^{(i,j)}}{radio\_range} & d_n \; and \; d_{n-1} \; are \; available \\ 1 & if \; d_n \; is \; unavailable \\ \frac{1}{2} & if \; d_n \; is \; available \; but \; d_{n-1} \; is \; not \end{cases} \tag{6}$$

In sum, our Bayesian approach estimates the variation of the future distance between every couple of nodes, normalized in the $[0,1]$ range. Values greater than 0.5 mean nodes to drift apart and smaller values to move closer. If the value is equal to 0.5, node i is estimated not to move with respect to j.

The parameters α and β are the inputs for Beta distribution $Beta(\alpha, \beta)$, where the expectation $\theta_{n+1}^{(i,j)} = \mathbb{E}\big(Beta(\alpha_{n+1}^{(i,j)}, \beta_{n+1}^{(i,j)})\big)$ is the variation of the distance between i and j in radio-range percentage that will be estimated at the beginning of $(n+1)$-th clock cycle.

At this stage we can estimate the distance between nodes i and j at the beginning of $(n+1)$-th clock cycle. That can be done from Equation 6 by replacing the observation term $o_n^{(i,j)}$ with the estimated value $\theta_{n+1}^{(i,j)}$. Hence:

$$\begin{aligned} &\widetilde{d}_{n+1}^{(i,j)} = d_n^{(i,j)} + \widetilde{\Delta d}_n^{(i,j)} = \\ &= d_n^{(i,j)} + (2\theta^{(i,j)} - 1) * radio\_range \end{aligned} \tag{7}$$

It should hold $d_n^{(i,j)} = d_n^{(j,i)}$; so, it should be $\widetilde{d}_{n+1}^{(i,j)} = \widetilde{d}_{n+1}^{(j,i)}$. But we have to consider $\widetilde{d}_{n+1}^{(i,j)} \neq \widetilde{d}_{n+1}^{(j,i)}$. Indeed distance sent by i about distance (i,j) can differ from what is sent by j about the same distance. This is why distances are collected at beginning of clock cycles but these can be shifted.

Therefore, estimated distance $\widetilde{d}_{n+1}^{i,j}$ is computed by considering both $\widetilde{d}_{n+1}^{i,j}$ and $\widetilde{d}_{n+1}^{(i,j)}$, through different weights.

$$\widetilde{d}_{n+1}^{i,j} = rel_{n+1}^{(i,j)} * \widetilde{d}_{n+1}^{(i,j)} + rel_{n+1}^{(j,i)} * \widetilde{d}_{n+1}^{(j,i)}$$

where $rel_{n+1}^{(i,j)}$ is a factor for the estimation reliability and it is inversely proportional to

[1] If a node has entered in this cycle we assume $o_n^{(i,j)}$ = 0.5, i.e., it is not moving.

$\sigma_{n+1}^{(i,j)} = \sqrt{Var(Beta(\alpha_{n+1}^{(i,j)}, \beta_{n+1}^{(i,j)}))}$:

$$rel_{n+1}^{(i,j)} = \frac{\frac{1}{\sigma_{n+1}^{(i,j)}}}{\frac{1}{\sigma_{n+1}^{(i,j)}} + \frac{1}{\sigma_{n+1}^{(j,i)}}} = \frac{\sigma_{n+1}^{(j,i)}}{\sigma_{n+1}^{(i,j)} + \sigma_{n+1}^{(j,i)}}.$$

2.3 Connected Components Computation

Disconnection prediction depends on a parameter γ, which stands for the fraction of the radio-range for which the predictive technique does not signal a disconnection anomaly[2]. Let be $P(disc_{n+1}^{(i,j)}) = P(\widetilde{d}_{n+1}^{(i,j)} \geq \gamma radio\_range)$; two nodes i and j are predicted going to disconnect if and only if

$$rel_{n+1}^{(i,j)} * P(disc_{n+1}^{(i,j)}) + rel_{n+1}^{(j,i)} * P(disc_{n+1}^{(j,i)}) > \frac{1}{2} \quad (8)$$

i.e. two nodes i and j are estimated disconnecting if it is more probable their distance to be greater than $\gamma radio\_range$ rather than distance to be smaller than such a value. We could tune more conservativeness by lowing γ (i.e. the fraction of radio-range in which disconnections are not predicted). If we consider Equation 7, then:

$$\begin{aligned} P(disc_{n+1}^{(i,j)}) &= P(\left| \tfrac{d_n^{(i,j)}}{radio\_range} + (2\theta^{(i,j)} - 1) \right| \geq \gamma) \\ &= P\big(\theta^{(i,j)} \geq \tfrac{1+\gamma}{2} - \tfrac{d_n^{(i,j)}}{2*radio\_range}\big) \end{aligned} \quad (9)$$

where the last term in Equation 9 is directly computable from the estimated beta distribution:

$$P(\theta^{(i,j)} > k) = \int_k^1 Beta\big(\alpha^{(i,j)}, \beta^{(i,j)}\big)$$

Once the algorithm predicts which links exist at the next cycle, we can compute easily the connected components (i.e., sets of nodes that are predicted to be connected). Afterwards, on the basis of the connected components, disconnection anomalies are identified by the monitor. Finally, they are notified either to the nodes (if a local management strategy is adopted) or to a coordination layer (if a global management strategy is adopted). Connected components are computable through "The Mobile Gamblers Ruin Algorithm" below, where an edge between couples of nodes in the connection graph exists if Equation 8 is false.

2.4 The Overall Technique

Our predictive algorithm, called as the "The Mobile Gambler's Ruin" (MGR) algorithm, is taken from the Markov chain model of the well known gambler's ruin problem [10, 11]. Such a study of the device movements and the consequent distance prediction is based on Markov chains, because the success of a prediction depends only on events of previous time frame units. Instead of using a markovian process in time domain, we are going to focalize on spatial domain and we will build a matrix, which is similar to the one presented in the original gambler's ruin model but with other elements.

Let's consider a square matrix of $|E| \times |E|$ elements, where $|E| = m$, with m, with m is the total number of mobile devices in the MANET. We build $\mathbf{M} = (m_{ij})$ as a $m \times m$ symmetric matrix, in which $m_{ij} = 1$ is the Equation 8 is false or, otherwise $m_{ij} = 0$ if the equation is true[3]. Every diagonal element $m_{ii} = 1$ since the $P(disc_{n+1}^{(i,j)}) = P(disc_{n+1}^{(j,i)}) = 0$. That follows for definition: the distance of a mobile device from itself is always equal to 0.

The matrix $\mathbf{M} = (m_{ij})$ can be considered as the Adjacency matrix of an (undirected) graph where the set of nodes are devices and an arc exists between two nodes if they are foreseen as direct neighbors.

The strategy of the MGR algorithm, which is described in Figure 3, is to find the connected components of the graph (using the CCDFSG procedure), and then, by giving two devices e_i and e_j, to verify if they belong to the same connected component (the TEST_CONNECTION function); if it is true then e_i, e_j will still communicate in the next time period; else they will lose their connection within the next time period. Using this strategy, after building the matrix $\mathbf{M} = (m_{ij})$, we can verify which devices are connected, directly (i.e., one hop) or indirectly (i.e., multi hop), and thus let decide when disconnection management techniques should be activated in order to keep the connection between the involved devices. The aim of such techniques should be to have a unique connected component in the graph.

The **MGR** algorithm computes the connected components starting from the matrix that represents the graph. The output of the **MGR** program is the *Comps* array in which for each i-th element there is an integer value corresponding to the connected component it belongs. For example, if we have a set of devices $E = \{e_1, ..., e_m\}$ and they form a graph with k connected components, we will have an output vector of this shape:

$$\big(0 \quad 0 \quad \ldots \quad 1 \quad \ldots \quad 2 \quad \ldots \quad k-1\big) \quad (10)$$

[2] As an example, in IEEE 802.11 with 100 meters of radio-range, γ equal to 0.7 means that for a communication distance of 70 meters the prediction algorithm signals a probable disconnection.

[3] The matrix is of course symmetric since always there holds $m_{ij} = m_{ji}$

```
FUNCTION MGR()
1  numcomps ← 0
2  Comps ← newArray_of_integer[m];
3  for i ← 0 to (m − 1)
4  do if Comps[i] = 0
5        then numcomps ← numcomps + 1
6             Comps[i] ← numcomps
7             CCDFSG(M, i, numcomps, Comps[])
8  return Comps[]

SUB CCDFSG(M, i, numcomps, Comps[])
1  for i ← 0 to (m − 1)
2  do if Comps[j] = 0 and M[i, j] = 1
3        then numcomps ← numcomps + 1
4             CCDFSG(M, j, numcomps, Comps[])
5

FUNCTION TEST_CONNECTION(i, j, Comps[])
1  if Comps[i] = Comps[j]
2    then TEST ← true
3    else TEST ← false
4  return TEST
```

Figure 3: Pseudo-Code of the MGR algorithm.

Thus for two different devices e_i, e_j we have only to test, using the TEST_CONNECTION program, if they have the same value in the vector (10), It will give us a confidence about the probability of being still connected in the next time period.

3 The OCTOPUS Virtual Environment

We implemented and deployed a fully-fledged version of the algorithm, which is running on PDAs, laptops e tablet PCs. The aim is to test a real implementation in order to get realistic results, instead of having information from simulations. On the other hand, on-spot testing may be expensive both in terms of resources and time: several persons are required to be arranged in a wide area. That might need much time (and, thus, high cost) to prepare the whole test-bed. Furthermore, field testing does not provide a controlled environment and it should be used just as the final user validation of the system.

Therefore, we test the implementation through emulation. We used OCTOPUS [4], an emulator, which we have implemented, specifically targeted for MANETs [4].

OCTOPUS keeps a map of virtual areas, which users can design and show by a GUI. Such a GUI enables the users to put in that map the virtual nodes and bind each one to a different real device. Furthermore, users can add possible existing obstacles in a real scenario: ruins, walls and buildings.

OCTOPUS benefits are that real devices are complete unaware of it: so when a node send packets, it believes to send them to the specified destination. But they are captured by OCTOPUS, playing a gateway role. OCTOPUS analyzes the sender and receiver: the distance of the corresponding virtual nodes in the virtual map (basically whether in the radio-range), the probability of losses, obstacles screening direct view[5], motion speed and so on. According to such parameters, it decides whether or not to deliver each packet to the recipient. Since the nodes are basically unaware of OCTOPUS, it is possible to remove it and deploy finally the software with no or very limited changes.

4 Technical Details

We implemented the Bayesian algorithm on actual devices. We coded in MS Visual C# .NET as it enables to write applications once and deploy them on any device for which a .NET framework exists (PCs and PDAs included). In this section, we describe the technical details of packages and classes for implementing the Bayesian algorithm.

We can identify two sides in the implementation as described in Figure 5: the code running on the coordinator device, which realizes the prediction, and the one on the generic peers sending information about neighbors to the coordinator.

The code of generic peers is conceptually easy. It is basically composed of two modules:

it.uniroma1.dis.Octopus. We tested our algorithm by OCTOPUS. OCTOPUS is intended to emulate small

[4] OCTOPUS can be downloaded at `www.dis.uniroma1.it/~deleoni/Octopus`

[5] We assume whenever two nodes are not directly visible, every packet sent by the first node to the second is always dropped.

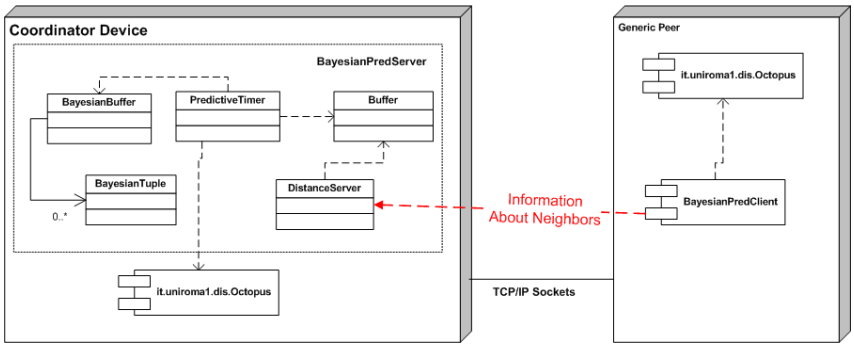


Figure 5: The components of the actual implementation.

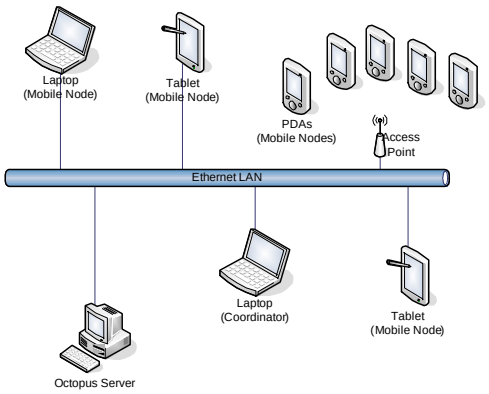


Figure 6: The test bed setting.

MANET and holds a virtual map of the are where nodes are arranged. Every real device is mapped to a virtual node in the map. This module is intended to query OCTOPUS in order to know neighbors and their distance. It allows also to send special commands to OCTOPUS during run-time emulation. In such a way modes can virtually move in the virtual map. That causes topology to change at any time and, therefore, new disconnections are continuously predicted.

BayesianPredClient. This module includes internally two timers. The first timer has a clock T, where T is the same as defined in Figure 2. For each clock period, it gets information about neighbors (who and how far they are) by using the *it.uniroma1.dis.Octopus* module. Then, it arranges such an information in a proper packet, which is sent to coordinator. Upon expiring of the second timer, the client sends a command to OCTOPUS to change the position of the node which this device is mapped to. Of course, this timer uses also the *it.uniroma1.dis.Octopus* module.

The code of the coordinator's implementation has as its core the *BayesianPredServer* module. In order to detail more, we have exploded it to its five classes:

DistanceServer. This module implements a TCP/IP server to retrieve the neighboring information from peers (sent by them through the module *BayesianPredClient*). At the same time, it stores retrieved information in the intermediate buffer, which is implemented by the module *Buffer*. It corresponds to event handler for *upon delivering of a tuple from a peer* as defined in Figure 2.

Buffer. It implements the intermediate buffer module, written by the *DistanceServer* module and read/made empty by *PredictiveTimer*. This module guarantees synchronized accesses.

PredictiveTimer. This is a timer that repeats each T seconds. It implements the event *upon expiring of timer* as defined in Figure 2. Consistently to the pseudo-code, it accesses to the *Buffer* module to get new information from other peers, as well as the *BayesianBuffer* module. The latter module stores the information to compute for each couple of nodes the Equations 4 e 6. This module uses also the *it.dis.uniroma1.Octopus* module. Indeed, Team Leader is a node itself and it can lead to disconnections. Therefore, it has to ask for neighbors to OCTOPUS and predict distances to any other node.

BayesianBuffer, BayesianTuple. The *BayesianBuffer* class handles and stores the triple $\left(\alpha_{(i,j)},\beta_{(i,j)},d_{(i,j)}\right)$, each one represented by a *BayesianTuple* object.

5 Experiments

Figure 6 shows the testbed, which consists of ten machines (PCs and PDAs). One of them hosts OCTOPUS and, hence, it does not represent a real mobile node. Each of the other machines is bound to a different virtual node of OCTOPUS' virtual map.

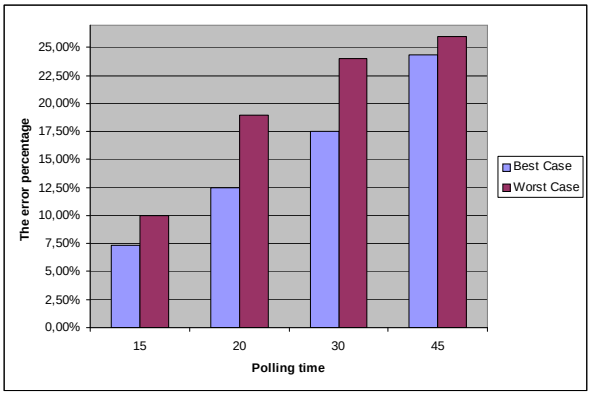
We set the testing virtual map as 400×300 meters wide and communication radio-range as 100 meters. At the beginning, nodes are located into the virtual map in a random fashion in order to form one connected component. Afterwards, each S seconds, every node chooses a point (X,Y) in the map and begin heading towards at a speed of V m/s. Both S and V are Gaussian random variables: the mean and variance are set as, respectively, 450 and 40 seconds for S and 3 and 1.5 m/s for V. The couple (X,Y) is chosen uniformly at random in the virtual map. Of course, devices used in tests do not move actually: nodes move only in the virtual map. For this purpose, devices send particular commands to a specific OCTOPUS socket for instructing node motions.

The first set of experiments has been intended to verify which error in percentage is obtained for different values of clock period T. The error here is defined as the gap between the estimated distances $\widetilde{d}_n$ at $(n-1)$-th clock cycle and the actual measures d_n at n-th clock cycle. The value is scaled with respect to the radio-range:

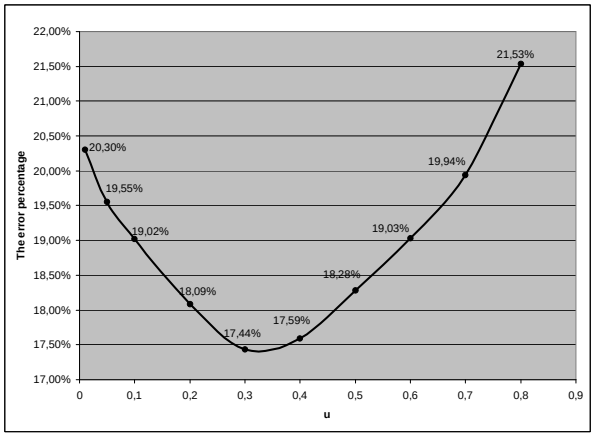
The Figure 7(a) shows the outcome for the clock periods equal to 15, 20, 30, and 45 seconds. We have set the parameter u of Equation 4 to value 0.5 and performed ten tests per clock period. Every test was 30 minutes long. The results show, of course, that the error percentage grows high as clock period increases. Probably the most reasonable value for real scenarios is 30–45 seconds (smaller values are not practically feasible since MANETs would be probably overloaded by "distance" messages). Please consider the greatest clock period we tested: the error ranges between 24.34% and 26.8% (i.e., roughly 25 meters).

Afterwards, in a second tests set, we fixed clock period to 30 seconds, testing for u equal to 0.01, 0.05, 0.1, 0.2, $\ldots$, 0.8. We even tripled the frequency which nodes start moving with. The outcomes are depicted in Figure 7(b), where x-axis corresponds to u values and y-axis to the error percentage. The trend is parabolic: the minimum is obtained for $u = 0.3$ where the error is 17.44% and the maximum is for $u = 0.8$ where the error is 21.54%. Small values for u mean that the past is scarcely considered whereas large values mean the past is strongly taken into account. This matches our expectation: we get the best results for the intermediate values. That is to say that the best tuning is obtained when we consider the past neither too little nor too much.

Concerning coordination, applications can rely on



(a) The smallest and largest measured error in percentage, changing clock periods.



(b) The measured error in percentage, changing the weight of past observations.

Figure 7: Experiment results.

such predictions. Indeed, setting polling time to 30 seconds, we have got errors around 18% for $u = 0.3$. If range is supposed to be 100 meters, the mean error is around 18 meters. Considering this is a preliminary validation, these values are quite good. Indeed, if we set $\gamma = 0.75$ (i.e., disconnections are predicted when nodes are more than 75 far), we would be sure to predict every actual disconnection. That means no disconnection is not handled, although coordination layer (distributed or centralized, local or global) will be alerted about some false negatives, enacting recovery actions to handle unreal disconnections.

We remark that this only a preliminary validation; we are working on more detailed experiments. In new test sets, we will be evaluating directly disconnection predictions: varying γ, how many disconnections are predicted and among them how many are real. We will be analyzing, as well, how many actual disconnections are not predicted on time.

6 Related Work

Much research on mobility prediction has been carried on (and still it is in progress) for cellular phone systems [1, 8]. These approaches are based on Markov models, which predict the mobile user future's location on the basis of its current and past locations. The aim is to predict whether a mobile user is leaving a current cell (crossing the cell boundaries) and the new cell where she is going. Such an information is then used for channel reservation in the new cell. Anticipating reservation should lower the probability of a call to be dropped during handoff [6] due to the absence of a free channel for the call in the new cell.

The main differences with our approach are related to different scenarios: MANETs versus mobile phone networks. Indeed, peculiarities of MANETs consist in the higher mobility compared with phone networks. In MANETs, links between couples of devices disappear very frequently. That does not happen in phone cells, which are very big: leaving a cell and entering into a new is rare with respect to how often MANET links falls down.

We use a centralized approach like in cellular network where a coordinator collects information to allow prediction. The difference is that our approach takes into account the knowledge of all distances among all users. Indeed, we do not have any base station; therefore, we do not have just to predict the distance of any mobile device to it. We are interested in the distance from any device to whichever else.

In the literature, several approaches predict the state of connectivity of MANET nodes. The most common approaches assume that some of nodes are aware of their location through GPS systems in order to study node motions and predict disconnections. In [19] the authors perform positioning in a network using range measurements and angle of arrival measurements. But their method requires a fraction of nodes to disseminate their

[6] In cellular telecommunications, the term *handoff* refers to the process of transferring an ongoing call or data session from one channel connected to a core network or cell to another.

location information such that other nodes can triangulate their position. In [20] the probability that a connection will be continuosly available during a period of time is computed only if at least one node knows its position and its speed through GPS. Our approach is more generic as it does not require any specific location techniques: every hardware allowing to know node distances is fine.

In [9], MANETs are considered as a combination of clusters of nodes and it studies the impact (i.e., the performances) of two well defined mobility prediction schemes on the temporal stability of such clusters; unlike our approach the authors use the pre-existing predictive models while the novelty of our approach consists in the formalization of a new model based on Bayesian filtering techniques. In [21] neighbor prediction in MANETs is enacted through a suitable particle filter and it uses the information inside the routing table of each node. Routing table is continuously updated by the underlying MANET protocol. The first drawback is that it can operates only with those protocols that work by updating routing tables. Since it is based only on routing table updates, it predicts how long couples of nodes are going to be connected on the basis of how long they have been connected in the past. It does not consider whether couples of nodes are moving closer or drifting apart, nor node motion speed. Our approach takes such an information also into account, making prediction more accurate.

[7] addresses the issue of robot location estimation. For each position p_i and each robot r_j, the technique gives the probability for r_j to be in p_i. This approach cannot be easily used to compute when nodes are going to disconnect.

7 Conclusion and Future Works

In this paper, we have proposed, implemented and tested a novel technique for predicting disconnections in MANETs. We think prediction is a basic block for any middleware for coordination in MANET settings.

Collaboration and coordination among MANET nodes could not take place when they cannot communicate. Since MANETs are very dynamic, nodes are continuously moving and that can cause several disconnections. If disconnections were not handled, communication could not take place and, consequently, coordination. Remedial actions (either local or global) need to be enforced in advance (i.e., predicted) against the actual disconnection.

This work is the basis of the development of a coordination system for MANETs in emergency management – cfr. the WORKPAD project (`http://www.workpad-project.eu`) we are currently involved, as our plans are to build a global management approach in which, after predicting disconnections, the coordination middleware instructs devices on how to arrange differently their tasks in order to keep the MANET connected.

We are realizing such a middleware: our first results consist in a general recovery method to detect and cope with any unpredictable event (see [5]) which changes the environment where the process is executed. These event may be such that the process cannot be carried on any longer. Therefore, we plan to apply the described prediction technique to such a middleware, as disconnections represent unforeseeable events.

Future works include the evaluation in very unreliable environments where information pieces could be lost. We plan, as well, to distribute prediction among all nodes which participate actively to the prediction. This should make the prediction layer more reliable as it does not have to rely on any special node (e.g., Team Leaders), which could crash. Moreover, distributing computational load among all devices permits to balance battery consumption. Indeed, every node consumes roughly the same energy amount, instead of having only a central node to consume its battery.

References

[1] I. F. Akyildiz, J. S. M. Ho, and Y. B. Lin. Movement-based Location Update and Selective Paging for PCS Networks. *IEEE/ACM Transactions on Networking*, 4(4):629 - 638, 1996.

[2] J. O. Berger. Statistical Decision Theory and Bayesian Analysis. *Springer*, 1985.

[3] T. Catarci, M. de Leoni, F. De Rosa, M. Mecella, A. Poggi, S. Dustdar, L. Juszczyk, H.L. Truong, G. Vetere. The WORKPAD P2P Service-Oriented Infrastracture for Emergency Management In *Proc. of the 3rd IEEE International Workshop on Collaborative Service-Oriented P2P System (COPS) at WETICE 2007*, 2007.

[4] F. D'Aprano, M. de Leoni, and M. Mecella. Emulating Mobile Ad-hoc Networks of Hand-held Devices. The OCTOPUS Virtual Environment. In *Proc. of the International ACM Workshop on System Evaluation for Mobile Platforms (MobiEval)*, 2007.

[5] M. de Leoni, M. Mecella, G. De Giacomo. Highly Dynamic Adaptation in Process Management System through Execution Monitoring. In *Proc. of*

the 5th Internation Conference on Business Process Management (BPM 2007), 2007.

[6] M. de Leoni, F. De Rosa, M. Mecella. MOBIDIS: A Pervasive Architecture for Emergency Management. In *Proc. of the 4th International Workshop on Distributed and Mobile Collaboration (DMC 2006) (at WETICE 2006)*, 2006.

[7] D. Fox, J. Hightower, L. Lao, D. Schulz, and G. Borriello. Bayesian Filters for Location Estimation. In *IEEE Pervasive Computing*, 2(3):24 - 33, 2003.

[8] B. Liang, and Z. J.Haas. Predictive Distance-based Mobility Management for Multidimensional PCS Networks. *IEEE/ACM Transactions on Networking*, 11(5):718 - 732, 2003.

[9] A. Venkateswaran, V. Sarangan, N. Gautam, and R. Acharya. Impact of Mobility Prediction on the Temporal Stability of MANET Clustering Algorithms. In *Proc. of the 2nd ACM International Workshop on Performance Evaluation of Wireless Ad hoc, Sensor, and Ubiquitous Networks (PEWASUN '05)*, 114 - 151, 2005.

[10] W. Feler. An Introduction to Probability Theory and its Applications (2nd ed.). *Willey*, 1966.

[11] G. Harik, E. Cantu-Paz, D. E. Goldgerg, and B. L. Miller. The Gambler's Ruin Problem, Generic Algorithms, and the Sizing of Populations. In *Proc. of the IEEE International Conference on Evolutionary Computation*, 1997.

[12] A. Jardosh, E. M. BeldingRoyer, K. C. Almeroth, and S. Suri. Towards Realistic Mobility Models for Mobile Ad-hoc Networks. In *Proc. of MobiCom*, 2003.

[13] G. Hackmann, R. Sen, M. Haitjema, G. C. Roman, and C. Gill. MobiWork: Mobile Workflow for MANETs. *Technical Report WUCSE-06-18, Washington University, Department of Computer Science and Engineering, St. Louis, Missouri*, 2006.

[14] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal. Vehicular Opportunistic Communication Under the Microscope. In *Proc. of the 5th International Conference on Mobile systems, Applications and Services (MobiSys)*, 2007.

[15] R. J. Punnoose, P. V. Nikitin, and D. D. Stancil. Efficient Simulation of Ricean Fading Within a Packet Simulator. In *Proc. of the 52th IEEE Vehicular Technology Conference*, 2000.

[16] Branislav Kusy, Jnos Sallai, Gyrgy Balogh, kos Ldeczi, Vladimir Protopopescu, Johnny Tolliver, Frank DeNap, and Morey Parang. Radio interferometric tracking of mobile wireless nodes. In *Proc. of the 5th International Conference on Mobile systems, Applications and Services (MobiSys)*, 2007.

[17] Thomas W. Malone, Kevin Crowston. The Interdisciplinary Study of Coordination. In *ACM Computing Surveys, Vol. 26, No.1*, March 2004.

[18] Mark Klein. Coordination Science: Challenges and Directions. In *Proc. of the workshop on Coordination Technology for Collaborative Applications - Organizations, Processes, and Agents (ASIAN)*, 1996.

[19] Dragos Niculescu, Badri Nath Position and Orientation in ad hoc Networks In *Elsevier Journal of Ad Hoc Networks, Vol. 2, No.2*, April 2004.

[20] Min Qin, Roger Zimmerman, Leslie S. Liu Supporting Multimedia Streaming Between Mobile Peers with Link Availability Prediction In *Proc. the 13th annual ACM International Conference on Multimedia*, 2005.

[21] Ovidiu V. Drugan, Thomas Plagemann, Ellen Munthe-Kaas Non-intrusive Neighbor Prediction in Sparse MANETs In *Proc. of 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON'07)*, 2007.