Disconnection Prediction in Mobile P2P Networks using publish/subscribe

Massimiliano de Leoni, Massimo Mecella
Dipartimento di Informatica e Sistemistica - DIS
Sapienza Universita' di Roma
Via Ariosto, 25 - Roma (Italy)
{deleoni,mecella}dis.uniroma1.it
Paolo Manfre', Junio Valerio Franchi, Daniele Graziano
Faculty of Computer Engineering
Sapienza Universita' di Roma

***Abstract*—In many pervasive scenarios (e.g., emergency management or health care), operators need to exchange data and information and collaborate in order to carry on a collaborative job, but communication features can be lacking on the spot. Therefore, Mobile Ad-hoc Networks are valuable solutions to let them coordinate. While executing some activities, nodes can move in the area and, hence, disconnect from the others. In order operators to coordinate with each other, devices need to be continuously connected, although that is not guaranteed, due to the mobile nature of the network. In this paper we present a bayesian approach to predict device disconnections in Mobile Ad-hoc Networks, and validating experimental results that show the viability of the approach. This prediction feature is useful for an upper coordination layer, which can arrange appropriate actions to prevent disconnections from occurring or, at least, to mitigate the consequences.**

***Keywords*-Disconnection prediction, Mobile P2P networks, Coordination, Disaster management, Global Positioning System (GPS), Publish/subscribe Middleware.**

I. Introduction

In complex mobile scenarios, such as military or civil protection, several team members for various emergency-response organizations need to collaborate and coordinate. Members are equipped with smart devices, such as PDAs, and execute some activities. Collaboration relies on the fact that members should be able to communicate. In many mobile scenarios the communication infrastructure may be missing or unavailable (e.g., in a rural area or, even, in a city after a certain emergency).

Therefore devices need to form a Mobile Overlay Network to enable the communication. In these scenarios the MANET use is gaining momentum. A Mobile Ad hoc NETwork (MANET) 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. Non-neighboring nodes can however communicate by using other intermediate nodes as relays, which forward incoming packets on a path to destinations [1]. 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 or other communication features are not guaranteed.

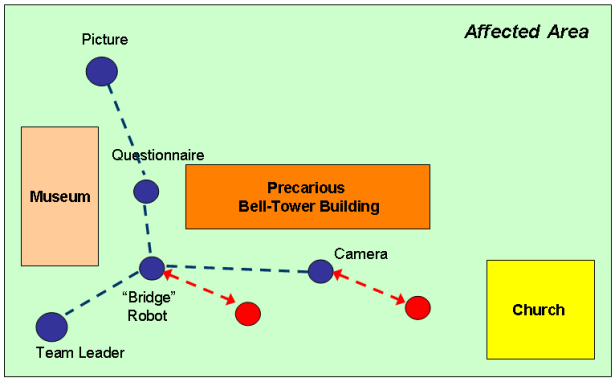


Figure 1. Critical situation and adaptive management

Being very pervasive scenarios, while executing activities, operators and their devices are continuously moving in the area, thus leading to disconnect from the other MANET devices (i.e., there is no other device in radio range to which the former is connected). If that happens, the disconnected node is no longer able to coordinate with the others, nor to execute actions. Therefore, some remedial actions should be enacted: a certain coordinate layer, residing on a special device or distributed over all nodes, ought to be informed, so as to prevent disconnections or mitigate the consequences.

Let's consider Figure 1: after a disastrous event (e.g., an earthquake) a team is sent to the disaster area. Let us suppose now operator `Camera` moves closer to the Church to analyze damages and take some photos. This movement could cause the disconnection of the operator device and, hence, the pictures taken cannot be stored into an appropriate storage (in Figure storage is provided by node *Picture*). At the same time, operator `Camera` cannot be assigned to further activities. In this example, when this disconnection is about to occur, we suppose a coordination

layer to enact recovery actions, thus keeping node *Camera* connected. Of course, there may be several further ways to handle disconnections: the coordination layer may decide nodes to cache some data to perform some activities in disconnection fashion.

Therefore the basic problem to be addressed for providing any kind of coordination on top of MANETs is to be able to predict disconnections of devices, allowing a coordination of nodes to enact for coping with them. Predicting in advance disconnections is worthwhile, as, indeed, recovery after the actual occurrence may be ineffective for two reasons: ***(i)*** some activities need to be rolled back when the executor disconnects and ***(ii)*** some remedial actions can be enacted only when nodes are still connected.

In this paper we propose a novel and effective technique based on Bayesian filtering, which has been validated by both laboratory experiments and an on-the-field drills.

The paper is structured as follows: Section II illustrates the approach, sketching the basic concepts of the Bayesian filters. Section III describes the general architecture of the concrete implementation, whereas Section IV details the experiments carried out and the resulting outcomes. Finally, Section V gives some concluding remarks.

II. Bayesian Filtering for Disconnection Prediction

Our predictive technique is based on few assumptions:

1) Each device is equipped with GPS hardware that allows it to know its space position. Recently, Hadaller et al. [2] proposes some techniques to mitigate the error when computing node position through GPS. Indeed, some experiments have been conducted 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. Kusy et al. [3] presents an alternative and valuable way to track the position of multiple wireless nodes simultaneously. It relies on measuring the position of tracked mobile nodes through radio interferometry and, experiments have shown to reduce significantly the error with respect to GPS.
2) At start-up, all devices are connected: 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 MANET routing protocols [4], e.g., OLSR, AODV, etc.).
3) A specific device, 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. This is not limitative as in many scenarios devices are connected when the respective operators start their activities. For instance, in disaster management rescue operators arrive to the area by jeeps and begin working when they are still connected.
4) A publish/subscribe middleware is deployed on every device. That, from the one side, allows devices to publish periodically their positions and, from the other side, allows the *coordinator* to predict the new positions.

The predictive technique is essentially as follows: nodes periodically publish their positions with period T using a pub/sub middleware. The coordinator subscribes to the middleware so as to receive them when new publications are made. On this basis, at every timestamp t the coordinator builds the probable *connection graph* which contains the MANET device positions at timestamp $t + T$ and the links among them. On the basis of these predictions, an higher-level coordination layer will be in charge of taking appropriated actions when some nodes are predicted as going out of range. Indeed, this approach has been plugged into ROME4EU [5], [6], an Adaptive Process-aware Information System specifically designed for emergency management and highly dynamic and pervasive scenarios. Indeed, ROME4EU is equipped with a special handler that can evaluate disconnection predictions and enact appropriate actions to prevent them from occurring or, at least, to mitigate the consequences.

A first version of the prediction techniques has been previously published in [7]. Here we propose several improvements in different directions. Firstly, we have distributed the information dissemination by making use of a pub/sub middleware, which guarantees an higher efficiency and more reliability in delivering information. Secondly, the initial proof-of-concept implementation has become a real fully-fledged component that has been plugged into ROME4EU. Thirdly, we have performed a more thorough analysis to evaluate the accuracy and goodness of the technique proposed.

A. Bayesian Filtering

Bayes filters [8] 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

timer: *a timer expiring with frequency* $1/T$.
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 PUBLISHING BY NODE I OF ITS POSITION(p)
 1  for each node j ≠ i
 2  do iBuffer[i, j] ← COMPUTEDISTANCE(i, j, p)

UPON EXPIRING OF TIMER()
 1  localBuffer ← iBuffer
 2  /*empty intermediate buffer*/
 3  for each (i, j) in ibuffer
 4  do ibuffer[i, j] ← −1
 5
 6  for each (i, j) in localBuffer
 7  do if localBuffer[i, j] = −1
 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.

sensor data available at time t:

$$Bel_t(\theta) = p(\theta|z_1, z_2, ...z_t) \tag{1}$$

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

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:

$$Bel_t(\theta) = Beta(\alpha_t, \beta_t, \theta) \tag{2}$$

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

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

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]*.

B. 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. The coordinator stores a bi-dimensional matrix *bayesianBuffer*, of which each cell (i, j) contains, in the n-th cycle, three data: $\alpha_n^{(i,j)}$ and $\beta_n^{(i,j)}$, and the last observed distance $d_{n-1}^{(i,j)}$. In addition, the coordinator stores also another bi-direction matrix *iBuffer*, which contains for each cell (i, j) the latest observed distance computed on the basis of the GPS positions of nodes i and j returned by the pub/sub middleware.

Let us assume a node k joins at 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 possible value for distance $d_{m+1}^{(k,j)}$ is equally probable.

With frequency $1/T$ each and every node i publishes its position into the pub/sub middleware. When i publishes this information, the coordinator receives it because of its subscription, and invokes procedure UPON PUBLISHING BY NODE I OF ITS POSITION. For each node $j \neq i$, this procedure computes the distance of i from j and updates matrix *iBuffer* accordingly. The computation is done by sub procedure COMPUTEDISTANCE on the basis of the GPS position of i and j. The procedure, abstracted out for the sake of space, updates the new i's position thus being available to compute future node distances. It is worthy saying that the frequency of publishing GPS positions is the same (i.e. $1/T$) for all nodes, but that happens in moments which are in general not synchronized.

When the timer, whose period is T, expires, the coordinator updates predictions by invoking procedure UPON EXPIRING OF TIMER as described in Figure 2.

Let n be the current clock cycle. At the n-th clock cycle, the procedure copies the tuples $(i, j, d_n^{(i,j)})$ from *iBuffer* to a local buffer (line 1) and, later, fills *iBuffer* with the special constant -1, which means absence of information.

After that, for all collected tuples $(i, j, d_n^{(i,j)})$, the coordinator updates the parameters using a Bayesian filter:

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

where $o_n^{(i,j)}$ is an observation and u is a constant in interval $[0, 1]$. This part is code in lines $11 - 13$.

Constant u is meant to age old observations. Indeed, as new observations arrive, the previous become less and less relevant. Indeed, old observations do not capture the updated status of MANET connectivity and the motion.

The observations 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. In that case, we assume the observation $o_n^{(i,j)}$ to be 1, i.e. i and j are no more neighbors. The distance between i and j could miss because i and j are actually not in radio-range. But that may also be caused by packet losses. In this second case, it is worthy noting that, due to the nature of Bayesian filters, nodes are not immediately assumed as disconnected; a second "disconnected" observation should be notified. As a matter of fact, the first lack could be caused by a simple packet loss.

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]

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

In sum, our Bayesian approach estimates the variation of the future distance between two nodes, normalized in the $[0, 1]$ range. Values greater than 0.5 mean nodes are drifting apart and smaller they are moving closer.

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:

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

C. 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. As an example, in IEEE 802.11 where meters, γ equal to 0.7 means that if the distance predicted between two given nodes is greater than 70 meters, then their link is considered as falling down. 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

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

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:

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

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 can be easily identified: nodes that belong to a different component than the coordinator's one are about to disconnect. Disconnections are finally notified either to the coordination layer, either centralized or distributed over all devices.

The last step is to compute the connected components. 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. The matrix is of course symmetric since always there holds $m_{ij} = m_{ji}$. 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.

[1] That corresponds to lines 7-10 where $localBuffer$ and $bayesianBuffer[i, j].distances$ are respectively $d_n^{(i,j)}$ and $d_{n-1}^{(i,j)}$

```
FUNCTION COMPUTE_COMPONENTS()
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
```

Figure 3. Pseudo-Code of the algorithm for computing the connected components.

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 Algorithm for computing the connected component is described in Figure 3: function COMPUTE_COMPONENTS returns the array of which the integer value at the i-th element denotes the connected component which node i belongs to. 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:

$$\begin{pmatrix} 0 & 0 & \dots & 1 & \dots & 2 & \dots & k-1 \end{pmatrix} \tag{10}$$

Thus, node i is considered as being about to disconnect whether the connected component to which i is predicted to belong is different from the coordinator's one.

D. On errors of communication distance evaluation

GPS and other positioning systems are prone to errors when computing node positions and/or communication distance evaluation. That does not affect significantly our model. Let us assume that for every $d_n^{(i,j)}$ there is an average error $\triangle D$ introduced by the real measure. Thus, by observing that our model is linear, it follows that the $\triangle D$ is spread all over the measures but doesn't depend on the clock cycle. So $d_n^{(i,j)}$ is actually in truth roughly $d_n^{(i,j)} \pm \triangle D$. Indeed, the exact value of $\triangle D$ depends on the technique used for distances evaluation. As it is typically small compared to $d_n^{(i,j)}$, then our average error on the prediction model is only partially affected by this error.

III. Implementation

The technique described in this paper has been implemented in MS Visual C# .NET in form of two different modules developed both for laptops and PDAS. A first module needs to be installed on every device and is intended for retrieving periodically the device GPS position and publishing it to the pub/sub middleware. A second module is deployed only on the coordinator: it subscribes for getting the positions of every MANET device and predicts possible disconnections according to the data gathered. The coordinator device is also one of the MANET nodes and, hence, is configured with both of modules.

For the current implementation, the above modules are interfaced with COSINE [9], a pub/sub middleware specifically conceived and developed specifically for PDAs communicating through MANET. The module specific of coordinator is provided with an event-based interface; through that interface, external modules can subscribed, thus being notifies when devices are predicted to be about to disconnect. Therefore, for every predicted node disconnection, the disconnection predictor module informs every subscribed module at the beginning of clock cycles. In order to simply the job of subscribers, if the same disconnection is predicted in subsequent cycles, the disconnection predictor informs just the first time.

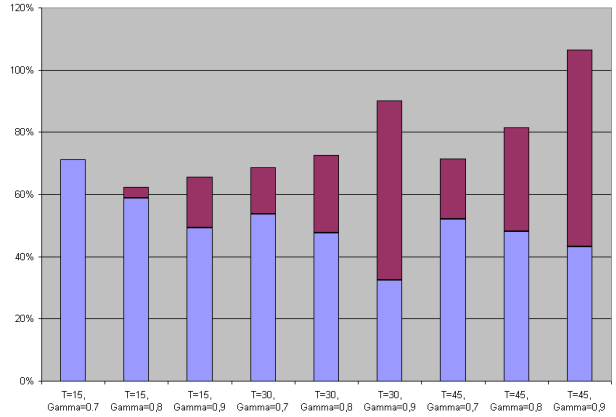
IV. Experiments

Testing on the spot may be expensive both in terms of resources and time. Indeed, several persons are needed and prepared for the experiments. Furthermore, field testing does not provide a controlled environment, nor experiments are repeatable. Therefore, it should be used just as final validation of the system. The best way for making some repeating and inexpensive experiments is to use emulation; during the emulation the software should require quite few changes, relying on an underlay communication stack that is somehow unreal. Following these points, we have developed OCTOPUS [10], and we have used it for testing the actual implementation of the disconnection prediction plug-in module.

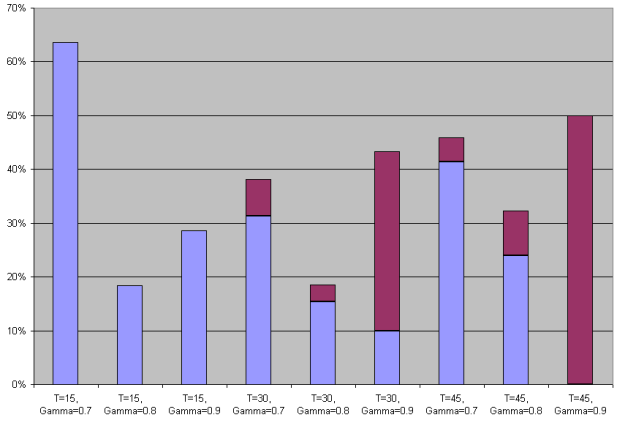
OCTOPUS handles a virtual map that contains virtual nodes/devices, obstacles and other typical objects present in a real open area. Virtual devices are associated to real devices. The machine hosting OCTOPUS and real devices are physically deployed in the same LAN. When a real device wants to send packets to another, such packets are in fact captured by OCTOPUS which acts as gateway. OCTOPUS analyzes whether sender and destination devices are connected according to the virtual map. If so, OCTOPUS forwards the incoming packet to the actual destination, arranging the headers thus hiding its presence. Therefore, OCTOPUS does not require software modules to be "adapted' for interworking with OCTOPUS. The software module is unaware.

Concerning the specific testing of the disconnection prediction technique, the test bed consisted of ten PDAs, which were virtually connected by OCTOPUS to form a MANET, but physically connected through an Wi-Fi access point.

At the beginning, nodes were located into the virtual map in a random fashion, but in any case thus forming one connected component. During experimenting, each S



(a) The mean value of false positives and negatives averaged on values for u between 0.1 e 0.9



(b) The minimum value of false positives and negatives for various u's values between 0.1 e 0.9

Figure 4. Tuning of the best values for Gamma for different values of clock period T. The blue (darker) bars show the percentage of false positives, whereas the red (brighter) bars display the percentage of false negatives.

seconds, every node chose a point (X,Y) in the virtual map and began heading towards at a speed of V m/s. Both S and V are Gaussian random variables: the mean and variance are, 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, the devices used during the experiments were not moving. In our emulated environment nodes moved only in the virtual map. For this purpose, OCTOPUS instructs the virtual motion of given nodes into the virtual map according to the schedule mentioned above.

The purpose of the first experiment set was to find the best values for γ according to different values of clock period T. Specifically we have tested for $\gamma \in \{0.7, 0.8, 0.9\}$, $T \in \{15s, 30s, 45s\}$ and u ranging from 0.1 to 0.9.

Figure 4 shows the cumulative sum of the percentages of false positive and negative cases measured. False positive cases comprise the disconnections that have not occurred but, wrongly, were predicted. False negatives include the disconnections occurred but never predicted.

Specifically, Figure 4(a) shows the mean value averaged on values u, where Figure 4(b) shows the smallest value over u for the cumulative sum of false positives and negatives.

From the analysis of Figure 4(a) and (b), it results that, fixing T, the number of false positives and negatives does, respectively, decrease and increase with increasing γ. Moreover increasing T causes a bigger number of false negatives and positives to occur. In fact, the information is gathered less frequently and the predictions need to look further on, losing some accuracy. Concerning the tuning, it seems that, generally speaking, $\gamma = 0.8$ gives the best results in term of minimizing the cumulative sum. In saying that, we have also judged false-positive cases to be less dangerous than false negatives. Indeed, a false negative (i.e., an unpredicted disconnection) causes unavailability of nodes, possibly compromising the successful activity executions; conversely a false positive would cause the coordination layer to perform unnecessary recovery actions, which reduces the effectiveness but does not prejudice the successfully execution of activities.

In the second testing phase, we aimed at understanding the influence of u on the results. Analyzing the results for different values of T and γ, we have learnt values for u ranging between 0.4 and 0.6 guarantee lowest percentage of false negatives and positives.

Space limitations prevents us again from displaying different graphics showing the percentage of false positive/negative cases for several values of T and γ. Figure 5 depicts the testing results with u ranging between 0.1 and 0.9 for configuration $T = 30$ and $\gamma = 0.8$. Indeed, this configuration balances very well against the configurations with $T = 15$ where a great deal of publication messages floods MANET and $T = 45$ where the false cases are too frequent. Moreover, too small T's values is such that predictions are not really such; for instance, predicting every 3 seconds (i.e., $T = 3$) is like predicting the current state and is not like looking forward in time.

V. CONCLUSIONS

In this paper we have proposed and experimented a novel techniques for predicting disconnections in MANETs. This is a basic problem when building coordination middleware for mobile scenarios, as we argue that any kind of remedial actions need to be enforced in advance and, hence, disconnections needs to be predicted before the actual occurrence.

As from our previous paper [7], some approaches already exist to predict disconnection, but they fail when

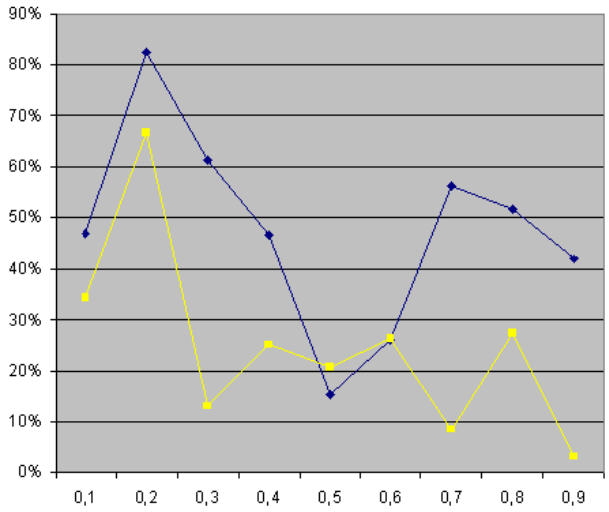


Figure 5. The results for $T = 30, \gamma = 0.8$ where u ranges from 0.1 to 0.9. The blue (darker) line shows the percentage of false positives, and the red (brighter) the percentage of false negatives.

applied to MANET. Indeed, the main differences with our approach are related to different scenarios: MANET versus mobile phone networks. Indeed, peculiarities of MANETs consist in that higher mobility, compared with phone networks. In MANETs, links between couples of devices appear and disappear very frequently. That does not happen in phone cells, which are very wide: leaving a cell and entering into a new is rare with respect to how often MANET links fall down. Due to the different nature of MANETs and mobile phone networks, the actual outcomes of the two classes of approaches are not comparable with each other.

We have developed a concrete implementation of a disconnection predictor, which has been used in concert with our Process-aware Information System named ROME4EU. The goodness of the approach proposed has been firstly tested with laboratory experiments, as detailed in Section IV. Secondly, we have plugged the module into ROME4EU [5], a Process-aware Information System (coordination layer) specifically conceived for pervasive environments and smart devices, such as PDAs.

Indeed, ROME4EU and the disconnection predictor have been used in the context of the EU project WORKPAD [11], which concerns devising a software infrastructure to support rescue teams. In June 2009 we simulated an emergency occurrence, specifically an earthquake, and provided rescue operators with PDAs to coordinate the management of the aftermath. PDAs were communicating with each other by MANETs and moving inside the affected area while executing the assigned activities. During such a drill, we have made some tests of the practical feasibility of the disconnection prediction technique. Unfortunately, the results are only qualitative data: a quantitative tracking is very hard to compute during on-the-field drills.

We have anyway noticed that nodes, while moving in the area, were prone to several disconnections, which ROME4EU was able to manage since it was working in concert with the disconnection prediction module. The lack of such predictor would have caused many problems to enact actions to recovery from disconnections of nodes, with obvious influence in the effectiveness of executing some activities. Indeed, when disconnected, nodes are out of control and, hence, many possible recovery actions cannot be enacted any longer, if involving activities/actions for disconnected nodes.

Acknowledgement.: This work has been partly supported by the European Commission through the FP6-2005-IST-5-034749 project *WORKPAD*.

REFERENCES

[1] D. P. Agrawal and Q. A. Zeng, *Introduction to Wireless and Mobile Systems*. Thomson Brooks/Cole, 2003.

[2] D. Hadaller, S. Keshav, T. Brecht, and S. Agarwal, "Vehicular opportunistic communication under the microscope," in *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*. ACM Press, 2007, pp. 206–219.

[3] B. Kusy, J. Sallai, G. Balogh, A. Ledeczi, V. Protopopescu, J. Tolliver, F. DeNap, and M. Parang, "Radio interferometric tracking of mobile wireless nodes," in *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*. ACM Press, 2007, pp. 139–151.

[4] S. Papanastasiou, L. M. Mackenzie, M. Ould-Khaoua, and V. Charissis, "On the interaction of TCP and Routing Protocols in MANETs," in *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*. Washington, DC, USA: IEEE Computer Society, 2006, p. 62.

[5] D. Battista, M. de Leoni, A. Gaetanis, M. Mecella, A. Pezzullo, A. Russo, and C. Saponaro, "ROME4EU: A Web Service-Based Process-Aware System for Smart Devices," in *ICSOC '08: Proceedings of the 6th International Conference on Service-Oriented Computing*. Springer-Verlag, 2008, pp. 726–727.

[6] D. Battista, D. Graziano, J. V. Franchi, A. Russo, M. de Leoni, and M. Mecella, "A Web Service-based Process-aware Information System for Smart Devices," Dipartimento di Informatica e Sistemistica - SAPIENZA University of Rome, Tech. Rep. 5, 2009, `http://padis2.uniroma1.it:81/ojs/index.php/DIS_TechnicalReports/issue/view/198`.

[7] M. de Leoni, M. Mecella, and R. Russo, "A Bayesian Approach for Disconnection Management in Mobile Ad Hoc Networks," in *WETICE '07: Proceedings of the 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 62–67.

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

[9] L. Juszczyk, H. Psaier, A. Manzoor, and S. Dustdar, "Adaptive Query Routing on Distributed Context - The COSINE Framework," in *MDM 2009: Proceedings of the 10th International Conference on Mobile Data Management*. IEEE Computer Society, 2009, pp. 588–593.

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

[11] T. Catarci, M. de Leoni, A. Marrella, M. Mecella, G. Vetere, B. Salvatore, S. Dustdar, L. Juszczyk, A. Manzoor, and H.-L. Truong, "Pervasive Software Environments for Supporting Disaster Responses," *IEEE Internet Computing*, vol. 12, no. 1, pp. 26–37, 2008.