Mobile Ad hoc Networks for Collaborative and Mission-critical Mobile Scenarios: a Practical Study

Gianluca Bertelli, Massimiliano de Leoni, Massimo Mecella
Dipartimento di Informatica e Sistemistica
SAPIENZA - Università di Roma
Via Ariosto 25, 00185 Roma, ITALY
{bertelli,deleoni,mecella}dis.uniroma1.it

Justin Dean
Information Technology Division
US Naval Research Lab
Washington, DC, 20375-5000, USA
jdean@itd.nrl.navy.mil

Abstract

*Nowadays Mobile Ad hoc NETworks (*MANET*s) are proposed in many collaborative and mobile scenarios. Despite the effort of the research community on algorithms and implementations mainly targeted to laptops, very few implementations exist for commercial PDAs (i.e., equipped with Windows Mobile). In this work, a practical study on one of the few available and running implementation has been conducted, with the aim of providing a toolkit for researchers, engineers and practitioners willing to use* MANET*s in their scenarios/systems/applications.*

1. Introduction

A Mobile Ad hoc NETwork (MANET) is a peer-to-peer 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-neighboring nodes can equally communicate by using other intermediate nodes as relays which forward packets toward 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 is not guaranteed. Examples are military applications, and more recently, cooperative systems for emergency management [3] or pervasive healthcare, as well as many other scenarios, which require highly dynamic node mobility.

As an example, let us consider a scenario of emergency management: each team member (e.g., a fire brigade operator, a Civil Protection operator, etc.) is equipped with handheld devices (PDAs) and communication technologies (e.g., WiFI for constituting a MANET), and, through the interplay with the software running on the device, can execute specific actions. The team member and his device offers a *service* towards the other members, and a Process Management System (PMS) coordinates the actions of all the services, according to a certain *process* [5]. In this scenario as well as in any scenario of distributed and mobile collaboration on MANETs, we believe that the communication layer is very important. According to the classification proposed in [7], coordination can be viewed as being divided into three layers, each depending on those below: communication, collaboration and 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. Therefore in order to design and deploy a really-working coordination system, we need to provide at low level a MANET communication layer that is robust and reliable. Moreover, we need to know what Quality of Service (QoS, e.g, the throughput) such MANET communication layer can really furnish. Indeed, if we did not consider actual provided QoS but we base on only simulated/theoretically-calculated values, we would build up coordination systems that are not really working.

In most of MANET scenarios (such as emergency management ones), nodes may be both (ultra-mobile) laptop and PDAs. Since we believe that de-facto PDA standard is Windows Mobile (due to the widespread availability of this operating system on commercial devices), we argue that it would be worthy to perform QoS evaluation of possible MANET communication layers on such an operating system. But currently many of available MANET implementations for PDAs are running only on Linux-based hacked PDAs and not at all on Windows Mobile. Among the few implementations working on both on Windows-based laptops and PDAs, which are described in Section 2, we tested in this work the US Naval Research Lab one.

We did not test through simulations, as many of other approaches do, since it would not return actual results. Conversely we performed emulation, by letting PDAs really exchange packets. Clearly on-field tests would be the better solution, but they require many people moving around in large areas and repeatability of the experiments would be compromised. Therefore emulation is considered an acceptable trade-off, in which the mobility is "synthetic" but the devices (and whatever running onto) are real (to be compared vs. simulation, in which both mobility and devices are synthetic). An important concern is that since all nodes are in the same laboratory room, the interference among nodes might be quite enough. Nevertheless, we discovered and proofed a relationship between laboratory results and on-the-spot ones, thus being able to derive on-the-spot performance levels from those got in the laboratory.

The aim of this work is to present the conducted experiments on MANETs of Windows-based PDAs, based on the US Naval Research Lab implementation, in order to derive lessons learned and guidelines useful to researchers, engineers and practitioners when developing mobile collaborative applications on ad hoc networks. Section 2 provides some background material and compares with relevant work. Section 3 describes some technical aspects of the specific implementation, whereas Section 4 shows the test bed, as well as the obtained results. Section 5 concludes the paper, analyzing results and providing some guidelines.

2. Background

2.1. MANET Routing Protocols and Implementations

Routing protocols, especially in a MANET where a node has a mobility property, can be divided in *(i)* topology-based or *(ii)* position-based. A position-based routing needs information about the current physical position of a node, that can be acquired through a "localization service" (e.g., a GPS). Only very recently, such localization services are becoming easily available on PDAs, although the research community has already defined some protocols of this type since long time (e.g., [8, 2]). Topology based protocols use information about the existent link beetween each node in the network. These protocols can be classificated by the "time of route calculation": *(i)* proactive, *(ii)* reactive and *(iii)* hybrid.

A proactive approach to MANET routing seeks to maintain a constantly updated topology understanding. The whole network should, in theory, be known to all nodes. This results in a constant overhead of routing traffic, but no initial delay in communication. Example protocols are OLSR and DSDV [11].

Reactive protocols seek to set up routes on-demand. If a node wants to initiate a communication with a node to which it has no route, the routing protocol will try to establish such a route. DSR [6], AODV [1] and DYMO [2] are all reactive protocols. Finally hybrid protocols use both proactive and reactive approaches, as ZRP [3].

Out of these routing protocols, some implementations exist, mainly for laptops, and only a few of them works on PDAs. Protocols that require special equipment on board of devices or on the field, such as position-based protocols, were discarded in our study because we aim at using off-the-shelf devices and at operating with no existing infrastructures (e.g., in emergency management). Moreover, we notice that reactive protocols in general have worse performance than proactive ones in term of reactiveness to changes in the topology, conversely proactive protocols require more bandwidth [10].

A working implementation of AODV is WINAODV [12]; DYMO is the most recent project and hence it is still in the standardization stage; an implementation for PDAs does not exist yet. Three OLSR working implementations are available. The OLSRD [4] has a strong development community and it can be extended through plug-ins. The "OLSRD for Windows 2000 and PocketPc" implementation [5] is the porting of the laptop OLSR version to mobile devices. But these two projects seem not to be working properly on the latest Windows CE version (Windows Mobile 6). The NRL (US Naval Research Lab) implementation [6] offers QoS functionalities, appears as a mature project, works on Unix/Windows/WinCE and also on some network simulators (e.g., ns-2).

2.2. Emulation vs. Simulation

Both simulation and emulation provide a controllable environment which enables several experiments in a cheaper fashion with respect to field tests. Simulator and emulator do not exclude each other. Simulation can be used at earlier stage: it enables to test algorithms and evaluate their performance before starting actually implementing on real hardware or devices.

Simulations are not intended to "replicate" all features of real hardware or software components, although every reproduced aspect has to keep the same performance levels of the one that is simulated. Even if application code written on top of the simulators can be quickly written and performances easily evaluated, it must be thrown out and rewritten when developers want to migrate on real architectures.

The emulators' approach is quite different: during emulation, some software or hardware pieces are not real whereas others are exactly the ones on actual systems. All emulators share the same idea: software systems are not

[1]`http://www.faqs.org/rfcs/rfc3561.html`
[2]`http://tools.ietf.org/html/draft-ietf-manet-dymo-02`
[3]`http://www.tools.ietf.org/id/draft-ietf-manet-zone-zrp-04.txt`
[4]`http://www.olsr.org`
[5]`http://www.grc.upv.es/calafate/olsr/olsr.htm`
[6]`http://cs.itd.nrl.navy.mil/work/olsr/index.php`

aware about working on an emulated layer (at all or partially); the software running on emulators can be deployed on actual systems with very few or no changes. On the other hand, performance levels may be worse.

In our study, we used an emulator, namely OCTOPUS [4], which keeps a virtual map with the position of virtual nodes. Each virtual node in OCTOPUS is bound to a real device. During the experiments, all real devices were deployed in a real MANET. The same network comprised also a dual-core workstation where OCTOPUS was running. During emulations, when a device builds a route to a certain node, it broadcasts an appropriate packet in the network; all broadcasted packets are captured by OCTOPUS, which forwards it to all virtual neighbors, according to the maintained virtual map. In turn the virtual neighbor that is on the best path to get to the destination replies. Every node is unaware that neighbors are virtual: the MANET routing algorithm and implementation are not changed to work in the emulated environment. Afterwards, communication takes place directly without passing any longer through OCTOPUS (till the next phase in which a new path should be calculated, in which again broadcasting is done through OCTOPUS).

The first difference with real scenarios is that route requests are passing through OCTOPUS. But since route request packets are sensibly less than data packets, the performance decrease is really slightly.

Finally OCTOPUS enable clients to interactively influence changes in topology, upon firing of events which were not defined before the start of the emulation. Other emulators defines in batch mode, i.e., when emulation is not yet started, which and when events fire. A deeper comparison of different emulators for MANETs can be found in [4].

3. NRL Implementation of OLSR Protocol

NRLOLSR is a research oriented OLSR implementation, evolved from OLSR draft version 3. It is written in C++ according to an object oriented paradigm, built on top of the NRL protolib library [7] for system portability.

Protolib works with Linux, Windows, WinCE, OpenZaurus, ns-2, Opnet; it can works also with IPv6. It provides a system independent interface, so NRLOLSR does not make any direct system calls to the device operating system. Timers, socket calls, route table management, address handling are all managed through protolib calls. To work with WinCE, protolib uses the RawEther component to handle raw message and get access to the network interface card.

The core OLSR code is used for all supported systems. Porting NRLOLSR to a new system only requires re-defining existing protolib function calls.

NRLOLSR has non-standard command line options for research purposes, such as "shortest path first route calculations", fuzzy and slowdown options, etc. Moreover, it uses a link-local multicast address instead of broadcast by default.

[7] http://cs.itd.nrl.navy.mil/work/protolib/

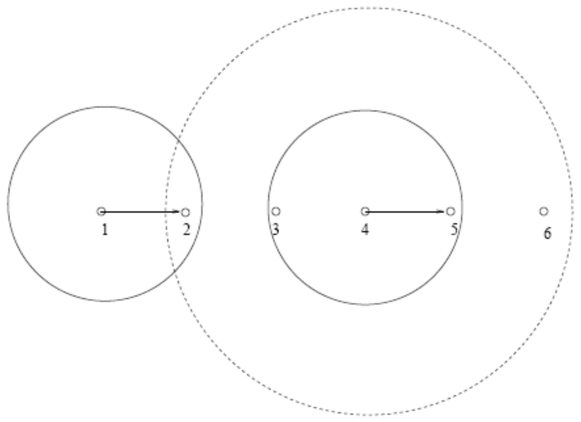

Figure 1. MAC interference among a chain of nodes. The solid-line circle denotes a nodes valid transmission range. The dotted-line circle denotes a nodes interference range. Node 4's transmission will corrupt node 1's transmissions to node 2

4. Testing MANETs

In our tests all the devices are in the same room, which means that they are in a medium sharing context. An 802.11 compliant device can not receive and/or transmit simultaneously. If multiple devices are in the same transmission range, only one at a time will be able to transmit data. Therefore the whole bandwidth is shared amongst all the devices in the same transmission range. In a simple chain topology in which all devices are far away from each other at the maximum transmission range, (Figure 1) packets travel along a chain of intermediate nodes towards the destinations. The successive packets of a single greedy connection interfere with each other as they move down the chain, forcing contention in the MAC protocol. This kind of topology is the basic block of many other configurations and it is affected by the ad hoc problems (hidden and exposed terminal).

4.1. From Laboratory Tests to On-field Results

Let $Q_{field}(n)$ be the throughput in a real field for a chain of n links (i.e., $n+1$ nodes). We want to define a method in order to compute it starting from $Q_{lab}(n)$, which is computed from the corresponding virtual chain; it is virtual, since topology is kept by OCTOPUS, where all the communication happens inside the laboratory. Hence, as said before, there is an higher interference.

Here, we aim at finding a function $Conv(n)$, such that:

$$Q_{field}(n) = Conv(n) \cdot Q_{lab}(n) \qquad (1)$$

in order to derive on-field performance. We rely on the following assumptions:

1. Every node is placed at a maximum coverage distance from the previous and the next node in the chain, such as in Figure 1. From other studies (e.g., [9]) we know that every node is able to communicate only with the previous and the next, whereas it can interfere also with any other node located at a distance less or equal to the double of the maximum coverage distance.

2. The first node in the chain wishes to communicate with the last one (e.g, by sending a file). The message is split into several packets, which pass one by one through all intermediate nodes in the chain.

3. Time is divided in slots. In the beginning of each slot all nodes, but the last one, try to send to the following in the chain a packet that they are in charge of.

4. Communications happen on the TCP/IP stack. Hence, every node that has not delivered a packet has to transmit it again.

As reported in the Appendix, the following statement is valid, where symbol $\lfloor\rfloor$ denotes the truncation to the closest lower integer:

Statement. *Let us consider a chain formed by* $(n+1)$ *nodes connected through* n *links. On the basis of assumptions above, function* $Conv(n) = \left(\lfloor\frac{n}{3}\rfloor + 1\right)^{\frac{\beta}{2}}$*, for some* β.

4.2. The Test-bed

The used devices are all off-the-shelf, certified for the 802.11b standard. In particular, iPAQ 5550 (running PocketPC 2003 - WinCE 4.2) and ASUS P527 (running Windows Mobile 6.0 - WinCE 5.0) are used. The environment is completed with some laptop running Microsoft Windows XP (SP2), some of them in the Tablet edition. In such laptops, the Microsoft PDA emulator is executed, as we aim at considering only MANETs of PDAs.

We build the ad hoc network with 802.11b, and we connect all the devices without any encryption and RTS/CTS ability turned off. One more workstation (equipped with a wireless card) is is running the OCTOPUS emulator and is the default gateway for the sub-net mask 192.168.0.128 on each device, so that every node sends its packets to the OCTOPUS machine. Each device is running the NRLOLSR protocol implementation specific for its operating system (WinCE or Windows XP).

We investigate on three kinds of tests: the performance of chain topology; some tuning related to the protocol; some tests with moving devices.

Performance of the chain topology. The aim of this test is to get the maximum transfer rate on a line of nodes. To obtain the measurements an application for Windows CE

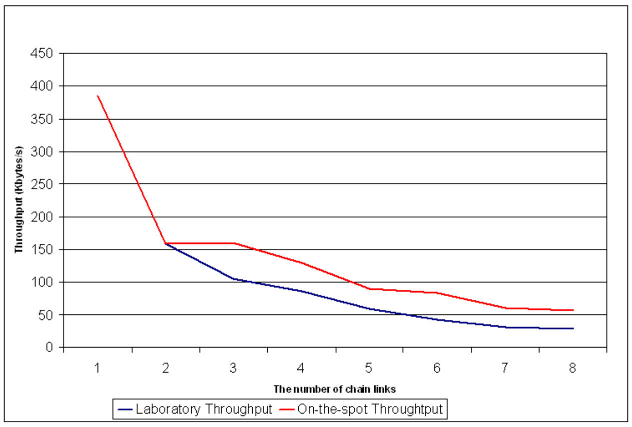

Figure 2. Test results for a chain-MANET, in the laboratory and estimated on-the-spot results

was built (using the .NET Compact Framework 2.0), which according to a client/server paradigm transfers a file from the head to the tail node (on top of TCP/IP) and reports the elapsed time.

All the devices use the routing protocol with the default settings and HELLO_INTERVAL set to 0.5 seconds; the broadcast address is set to 192.168.0.254. OCTOPUS emulates the chain topology and grabs all the packets sent to 192.168.0.254. Each host knows only the two neighboring nodes, as correctly in the OCTOPUS configuration, therefore it sends periodic HELLO messages to the broadcast address declaring only these two hosts as neighbors. When a node wants to communicate to another node it sends packets directly to it if this is in his neighborhood, otherwise it sends packets following the routing path. The test was ran five times for each configuration. Both real and emulated devices were used; each reported value is the mean value of the five attempts.

Figure 2 represents the results of the tests. The trend is compliant with what we are arguing in the statement of Section 4.1; specifically, through interpolation we discover that $\alpha = 385$ and $\beta = 1.21$. The red curve shows the predicted on-field performance obtained with the equation 10. It is clear that in the real field, where the devices are not all on the same transmission range, the throughput falls in the chart region between the two curves.

Tuning of the protocol. There are a lot of parameters of NRLOLSR that can be changed but only few of them have a strong impact on the protocol effectiveness. We focus on the HELLO_INTERVAL that is the most important value because it influences the mobility reactivity and the overhead of the protocol. We test how increasing or decreasing this parameter could affect the topology as seen by a node, and hence the reactivity of the network. As every mobility pattern can be stepwise considered as a crossing of chain of

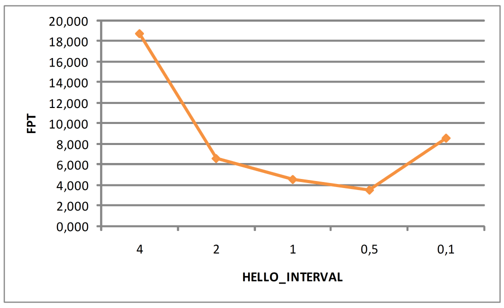

Figure 3. Time elapsed to establish a direct communication in a chain of five nodes

nodes, we investigate a single chain, by considering it as a "building block".

The scenario is as shown in Figure 1: the nodes in the chain are fixed and stable, each node knows only two neighbors; at time *t* node 1 enters in the range of node 2; we take the time elapsed between *t* and the first application message from 6 received by 1. To do this a client/server application that continuously sends UDP messages from the head node to the tail node was built; this indeed introduce a small delay that can be ignored.

This interval is referred to as FPT (First Packet Time) and it can be broken as follows:

$$FPT = 2 \cdot chain\_time + build\_route\_time \quad (2)$$

where *chain_time* is the time used by the packet to travel along all the chain and to come back, and *build_route_time* is the fraction of time that is necessary to the head node to build the new routing table and choose the correct path for the packet. To catch the exact time, in this test, the head node and the entering node are laptop instead of PDAs, so it easy to use a network sniffer software (that is not available on PDAs). Again the mobility emulation is provided by the OCTOPUS machine.

Figure 3 shows the trend of FPT vs. decreasing the HELLO_INTERVAL. Each reported value is the mean value of eight runs. The curve decreases linearly except on the last point, where the interval is set to 0.1 second. For interval leq 0.1 second the FPT increases. This result (a minimum around 0.5s) is due to the inability of the devices to follow the network load. The value of the minimum depends upon the CPU, the RAM, in general upon the hardware configuration of the PDA, so with more powerful devices the FPT can be faster.

All these values have to be considered for one single traffic flow, so in a real scenarios where the traffic is very high and there are multiple flows, it is important to choose an interval value that allows fast topology reactivity (minor FPTs) and that does not overload too much the devices.

Tests with moving devices. This kind of test is necessary to determine whether or not the NRLOLSR implementation is suitable for a real environment. In a real field it is important not to break the communication among the movements of the nodes. If a team member is transmitting information to another team member, and one of them runs away, without exiting from the MANET, all the data must be delivered successfully.

To replicate such a scenario, we investigate three topologies, as shown in Figure 4, where the dashed line shows the path of a running device. The topologies are designed in order to have *(i)* the moving node always connected at least another node, and *(ii)* each node is connected in some ways to at least another one, i.e., there are not disconnected node (no partitions in the MANET).

A WinCE application is used that continually sends TCP/IP packets of 1000 bytes between the node S and the node D, storing on both sides the sent/received bytes. To reach a higher number of nodes, five real PDAs and five emulated PDAs are used to perform the test. The test was ran five times for each topology and every run was 300 seconds long.

The results are good enough, for each topology and for each run the communication never breaks down, and there is no packet loss. This can be explained on the basis of the TCP/IP protocol, the movement's pattern and speed.

It is clear that there are multiple time slots where the moving node has not a route to D because is leaving or entering a node range, and it is computing the new routing table. During this time the TCP connection is in a waiting state, retransmission occurs, and a countdown starts. If the node computes the route before the TCP timeout goes off, the connection is woken up and so the communication can go on. It is important to note that with UDP it is not the same, because the connection does not go into a waiting state (no retransmission) but it continues to send datagram that go all lost until the new route is built.

In order not to incur in the TCP timeout, the speed of the movement is crucial: a too fast node cannot establish and maintain communication. On the topology that we have investigated the maximum reachable speed in order to perform a correct communication (without any losses) is more or less 18 m/s.

5. Guidelines and Conclusions

The results presented in this paper should be used as a toolkit for researchers, engineers and practitioners willing to use MANETs in their scenarios/systems/applications.

In particular, Figure 2 allows to carefully take into account the throughput that a MANET of real devices can nowadays support. Surely, on the basis of the previous discussions, whichever configuration of a MANET will present a performance that lies in the area between the two lines, being one the possible worst case and the other the possible best case. The result is very important when designing applications and middleware on top of MANETs of PDAs, as for example it stems that for more than 5 devices we have a throughput of about 50 Kbytes/s. Depending on your ap-

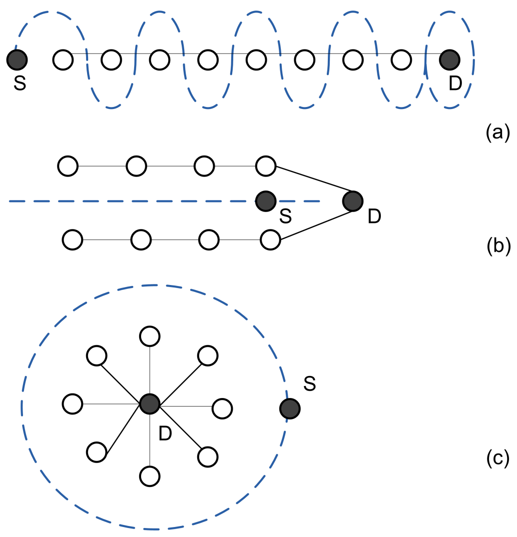

Figure 4. Dynamic topologies for testing TCP/IP disconnections

plication (e.g., exchanging an image – a map in a scenario of emergency management – of 2 Mbytes takes approx. 40 seconds) this could be fine, or conversely an optimized use of the bandwidth should be carefully designed in the application since the early stage, in order not to have unsatisfactory results.

Analogously, the results of Figure 3 help in designing applications in very dynamic MANETs (a lot of entering and exiting of devices), which require very frequent exchanges of messages, thus overloading the bandwidth, vs. applications that may adopt greater values of HELLO_INTERVAL being less reactive to changes.

Finally, the results of the tests of moving nodes, allow us to affirm that it is possible to build up reliable MANETs with nodes moving around, but if the nodes are held by humans walking and/or running (18 m/s are approx. 65 km/h). But if the nodes are moving on top of vehicles (such as cars on an highway), these values put serious doubts on the viability of the connection.

Just to make a final consideration (again with the aim of showing how the results of this paper are an useful toolkit), we can argue that current implementations of MANET routing protocols running on WinCE (and therefore on most of commercial embedded devices) are not appropriate for VANETs (Vehicular Ad hoc NETworks) and for the so-called Automotive Cooperative Systems. Indeed cars in an highway are in general faster and the topology changes more frequently than what we have shown, therefore future research in engineered solutions for this specific scenarios are needed.

In conclusion we can argue that more research and development are needed on engineered algorithms for MANETs of real mobile devices (e.g., PDAs). The implementations and experimentations conducted so far have missed an important category of devices (the one of which MANETs are really interesting and useful), which require specific studies and developments due to the performance constraints presented by such devices.

Acknowledgements. This work has been partly supported by the European Commission through the project FP6-2005-IST-5-034749 WORKPAD. The authors would like to thank Thomas F. Divine and PCAUSA for providing RAWETHER for WinCE to the group of SAPIENZA – Università di Roma.

References

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

[2] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proc. 4th ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98)*, pages 76–84. ACM, 1998.

[3] 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*, 12(1):26–37, 2008.

[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 ACM Workshop on System Evaluation for Mobile Platform: Metrics, Methods, Tools and Platforms (MobiEval) at Mobisys 2007*, 2007.

[5] M. de Leoni, M. Mecella, and G. De Giacomo. Highly Dynamic Adaptation in Process Management Systems Through Execution Monitoring. In *Proc. 5th International Conference on Business Process Management (BPM 2007)*, pages 182–197, 2007.

[6] D. B. Johnson, D. A. Maltz, and J. Broch. *Ad Hoc Networking*, chapter DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks, pages 139–172. Addison-Wesley, 2001.

[7] M. Klein. Coordination Science: Challenges and Directions. In *Proc. Workshop on Coordination Technology for Collaborative Applications - Organizations, Processes, and Agents (ASIAN)*, 1996.

[8] Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad hoc Networks. *Wireless Networks*, 6:307–321, 2000.

[9] J. Li, C. Blake, D. S. J. De Couto, H. I. Lee, and R. Morris. Capacity of Ad Hoc Wireless Networks. In *Proc. 7th International Conference on Mobile Computing and Networking (MOBICOM 2001)*, pages 61–69, 2001.

[10] S. Papanastasiou, L. Mackenzie, M. Ould-Khaoua, and V. Charissis. On the Interaction of TCP and Routing Protocols in MANETs. In *Proc. International Conference on Internet and Web Applications and Services/Advanced International Conference on (AICT-ICIW '06)*, page 62, 2006.

[11] C. E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proc. SIGCOMM 94*, 1994.

[12] D. West. An Implementation and Evaluation of the Ad-Hoc On-Demand Distance Vector Routing Protocol for Windows CE. M.sc. thesis in computer science, University of Dublin, September 2003.

Appendix – Proof of the Statement in Section 4.1

From the first assumption, we can say that, if the i-th node successes in transmitting, then $(i-1)$-th, $(i-2)$-th, $(i+1)$-th and $(i+2)$-th cannot.

Let us name the following events: *(i)* D_n be the event of delivering a packet in a chain of n links and *(ii)* S_n^i be the event of delivering at the i-th attempt.

Let us name $T_{i,n}$ as the probabilistic event of delivering a packet in a network of n links (i.e., $n+1$ nodes) after i retransmissions [8].

For all n the probability of delivering after one attempt is the same as the probability of deliver a packet: $P(T_{1,n}) = P(D_n)$. Conversely, probability $P(T_{2,n})$ is equal to the probability of not delivering at the first $P(\neg S_n^1)$ and of delivering at the second attempt $P(S_n^2)$:

$$P(T_{2,n}) = P(S_n^2 \cap \neg S_n^1) = P(S_n^2) \cdot P(\neg S_n^1 | S_n^2) \tag{3}$$

Since, for all i, events S_n^i are independent and $P(S_n^i) = P(D_n)$, Equation 3 becomes:

$$P(T_{2,n}) = P(S_n^2) \cdot P(\neg S_n^1) = P(D_n) \cdot (1 - P(D_n))$$

In general, the probability of delivering a packet to the destination node after i retransmissions is:

$$\begin{aligned} P(T_{i,n}) = P(S_n^i) \cdot P(\neg S_n^{(i-1)}) \cdot \ldots \cdot P(\neg S_n^1) = \\ = P(D_n) \cdot (1 - P(D_n))^{i-1} \end{aligned} \tag{4}$$

We can compute the average number of retransmissions, according to Equation 4 as follows:

$$\begin{aligned} T_n = \textstyle\sum_{i=1}^{\infty} P(T_{i,n}) = \\ = \textstyle\sum_{i=1}^{\infty} P(D_n) \cdot (1 - P(D_n))^{i-1} = \frac{1}{P(D_n)} \end{aligned} \tag{5}$$

In a laboratory, all nodes are in the same radio range. Therefore, independently on the nodes number,

$$P(D_n^{lab}) = 1/n \tag{6}$$

. On the field, we have to distinguish on the basis of the number of links. Up to 2 links (i.e., 3 nodes), all nodes interfere and, hence, just one node out of 2 or 3 can deliver a packet in a time slot. So, $P(D_1^{field}) = 1$ and $P(D_2^{field}) = 1/2$. For links $n = 3, 4, 5$, two nodes success: $P(D_n^{field}) = 2/n$. For links $n = 6, 7, 8$, there are 3 nodes delivering: $P(D_n^{field}) = 3/n$. Hence, in general we can state:

$$P(D_n^{field}) = \frac{\lfloor \frac{n}{3} \rfloor + 1}{n}. \tag{7}$$

By applying Equations 6 and 7 to Equation 5, we derive the number of retransmission needed for delivering a packet :

$$\begin{aligned} T^{field}(n) = \tfrac{n}{\lfloor \frac{n}{3} \rfloor + 1} \\ T^{lab}(n) = n. \end{aligned} \tag{8}$$

[8]Please note this is different with respect to S_n^i, since $T_{i,n}$ implies deliver did not success up to the $i-1$-th attempt

Fixing the number of packets to be delivered, we can define a function f that expresses the throughput in function of the number of sent packets. If we have a chain of n links and we want to deliver a single packet from the first to the last node in the chain, then we have altogether to send the number n of links times the expected value for each link T_n. Therefore:

$$\begin{aligned} Q_{lab}(n) = f(T^{lab}(n) \cdot n) = f(n^2) \\ Q_{field}(n) = f(T^{field}(n) \cdot n) = f(\tfrac{n^2}{\lfloor \frac{n}{3} \rfloor + 1}) \end{aligned} \tag{9}$$

From our laboratory experiments described in Section 4.2, as well as from other theoretical results [9]), we can state $f(n^2) = \frac{\alpha}{n^\beta}$. By considering it and Equations 9, the following holds:

$$\frac{Q_{lab}(n)}{f(n^2)} = \frac{Q_{field}(n)}{f(\frac{n^2}{\lfloor \frac{n}{3} \rfloor + 1})} \Rightarrow Q_{field}(n) = Q_{lab}(n) \cdot \left(\lfloor \frac{n}{3} \rfloor + 1\right)^{\frac{\beta}{2}}. \tag{10}$$