A novel architecture for a framework to support the storage of network simulation data into distributed storages for remote access

Maurizio Colizza
Center of Excellence DEWS, Italy
colizza@westaquila.com

Luca De Nardis
Sapienza University of Rome, Italy
lucadn@newyork.ing.uniroma1.it

Marco Patrizi
Sapienza University of Rome, Italy
marco.patrizi.lextel@gmail.com

Claudia Rinaldi
University of L'Aquila
claudia.rinaldi@univaq.it

ABSTRACT

Design, implementation and test of a communication network require the use of tools allowing the execution of simulations accurate enough to enable: a) the assessment of the correctness of the algorithms that govern the behavior of the nodes; b) the simulation of actual operating conditions in terms of accuracy of channel and mobility models, as well as of traffic conditions and network usage, in order to verify compliance with the QoS parameters; c) simulation code reusing on target devices. Simulation of Mobile Ad-hoc NETworks (MANETs), designed for civil and or military use is particularly challenging, due to high number of nodes, different mobility and channel models and different types of traffic and QoS requirements, and led to the design of several network simulators. Among them, OMNeT++, OPNET and ns-3 are arguably the most popular ones for scientific and industrial activities. Each of the above simulators is characterized by specific advantages and drawbacks, as well as different available models. Although they share an approach based on discrete event simulation, they are significantly different in the way the user provides inputs, collects outputs and builds his/her own models.
In this paper we propose a methodology to store simulation data in a distributed and scalable storage system that is exploitable in any of the above listed simulators. Furthermore, the paper illustrates a way to apply the proposed methodology to an emulation environment.

1. INTRODUCTION

Simulation experiments are widely used in the domain of the Mobile Ad hoc Networks (MANETs) to evaluate the results of the design activities. These experiments must model the network topology, network traffic, and the routing and other network protocols. In addition, the wireless and mobile nature of MANETs necessitates consideration of node mobility, physical layer issues, including the radio frequency channel, terrain, and antenna properties, and, perhaps, energy and battery characteristics. Node mobility, coupled with physical layer characteristics, determines the status of link connections and, hence, the network's dynamic topology. Link connectivity is an important factor, if not the most important factor, affecting the relative performance of MANET routing protocols. Accurate models are needed in order to realize high fidelity simulations; also, different models must be used at the same time to realize realistic scenarios.
In the following, a brief review of typical simulation settings adopted in the analysis of MANETs is provided. As mobility models are concerned, the random waypoint model is one the most commonly used mobility models for simulations of MANETs [16, 39, 40, 17, 19, 28, 31, 32, 43]. This model is implemented in three popular simulation tools, ns-2 [26], QualNet [13], and OPNET [11]. Computational resources required to compute node positions and determine link connectivity can be relatively high, thus increasing the cost of simulation. In [15] the authors have investigated the influence of three mobility models on the performance of ad hoc network routing protocols (AODV and GPSR). As a benchmark, the first model used is the popular random waypoint mobility model. The second model is based on a vehicular traffic simulator. Finally, as third model, the authors propose a novel mobility model based on vectorized street maps and speed limit information extracted from a geographic information system. The vehicular mobility traces used in [15] are available at ETH Zurich, where was developed a Multi-agent Microscopic Traffic Simulator (MMTS). MMTS models the behavior of people living in a specific area, reproducing their movement (using vehicles) within a period of 24 hours [41], [42]. Around 260.000 vehicles are involved in the simulation with more than 25.000.000 recorded direction/speed changes in an area of around 250 km for 260 km. In [38] the authors investigate the impact of realistic radio propagation settings on the evaluation of VANET-based systems, using the JiST/SWANS simulator [14] and rely on STRAW [20] to model vehicular mobility. All simulations are conducted on a map of the same region of Chicago, bounded on the east side by Lake Michigan and on the south by the Chicago River. The field extends about 1.6 Km to the west and 1.1 Km to the north, giving an area

EMUTools Workshop 2013, March 05-07

DOI 10.4108/icst.simutools.2013.251715

of about 1.76 Km2. The area contained 331 road segments with a total length of 60 Km. The simulation also included 150 nodes on the field at random intersections, and selected 25 of these nodes as transmitters. Each node sent a beacon on average every 5 seconds, and the transmitting nodes has injected a message with average size of 10 KB approximately every 60 seconds. Another aspect relevant to MANET design is the measure of the overhead of traffic signaling due to routing activities. In [46] a mathematical and simulative framework is presented for quantifying the overhead of reactive routing protocols, such as dynamic source routing and ad hoc on-demand distance vector, in wireless variable topology (ad hoc) networks. The authors have considered five networks of size 49, 121, 225, 361, and 529. Every node is placed at the intersection of a square Manhattan grid. Here, nodes have fixed positions without any movement for the entire simulation. Different field sizes was chosen for different network sizes; 1400 m by 1400 m field has 49 nodes; 2200 m by 2200 m for 121 nodes; 3000 m by 3000 m for 225 nodes; 3800 m by 3800 m for 361 nodes; and 4600 m by 4600 m for 529 nodes. The simulator used was ns-2.
The above overview on the complexity of MANET simulations, which is represented from heterogeneity of the models and of data, is useful to understand the need to have a system to store the results derived from a simulation and a methodology conceived to improve the computational resources by means of emulation tools.

In the present work we focus on two problems which concern the simulations of MANETs networks: the storing of the simulation data and how to improve the experiment by means of emulations resource. The paper is organized as follows: Section II presents a methodology and an architecture to store data in a scalable database; in the Section III the proposed methodology is extended to emulation domain; finally, the paper closes with the conclusions and future works discussed in Section IV.

2. PROPOSED METHODOLOGY

In order to improve the analysis of data provided from a MANET simulation the following items should be taken into account:

- All the simulator used to simulate MANET networks are discrete event simulator;
- The data are heterogeneous: a qualitative and non exhaustive list follows:
- statuses of finite state machines;
- routing tables;
- Packet Error Rate;
- Signal to Noise Ratio;
- end-to-end delay
- Each one of data provided from a MANET simulation is generated in asynchronous way;
- The amount of data generated from MANET simulations is large, due to:
- the complexity and heterogeneity of models;
- the simulation time, which for realistic simulations it could be tens or hundreds of hours.

Also, we want to remark the following Critical Success Factors (CSF) [18] for the development of a Software (SW) conceived to store MANETs simulation data:

- The use of the interface in order to exchange data between the simulation and the storage system;
- The use of the scalable storage;
- The use of a methodology to acquire information of measure that is:
  - reusable across multiple network simulators and on different platforms (Windows, Linux, Mac OS X);
  - valid for different types of measurement.

With the aim to take into account all the previous constraints, we have defined a methodology to realize the storage of the simulation data, named MANET Simulation Data Storage Methodology (MSDSM). The guidelines of our methodology are listed below:

- The measurement information must be collected in XML format via a serialization process of the object which includes the information;
- The buffer which includes the XML string must be sent on a IP connections to a System, System for Storage Management, (SSM) which has in charge the management and the archiving of XML data to a scalable storage system;
- Each data XML string must have an header which must be used to classify the data and to select a properly rule for the archiving;
- The SSM system must be able to do the following operations:
  - Classification of the data XML string;
  - Selection of a properly archiving rule;
  - change of the data XML string in a sequence of bytes;
- Storage of the sequence of bytes in a scalable storage system;

Using MSDSM, we have conceived a three tiers architecture [24] for a Software Connector (SC) [37] as shown in Figure 1.
The test environment is made up from the elements listed below:

- a Windows machine which hosts:
  - a main process which provides a XML data string; the string is got by means of a serialization operation applied to a prototype of a class which models a routing table; both "main" and class routing table were implemented using the C++ language;
  - an object, named ClsTst, which implements the SAP, coded using Java language;

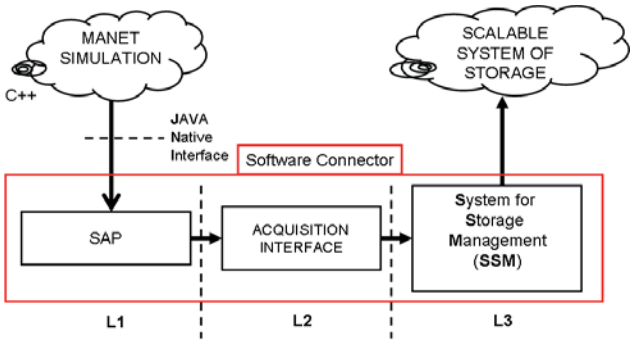

Figure 1: Connector Architecture.

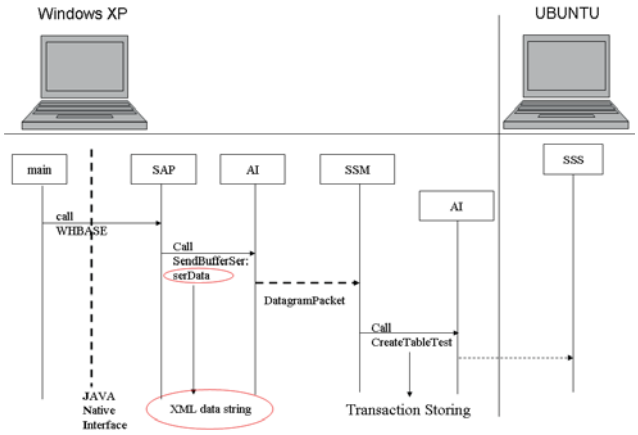

Figure 2: Test Environment.

  - an object, named WrapperHBASE, which implements the AI, coded using Java language;
  - a process, named ClientServer, which implements the SSM, coded using the Java language;

- a Linux machine, which hosts a scalable Database (DB), HBASE, configured in pseudo distributed mode [25], [29].

The "main" process shown in Figure 2 plays the role of a generic process used to simulate the protocols stack of a node of a MANET; in the test environment this was a simulator implemented in OMNeT++ [10]. At the same time, the main process executes the serialization of a routing table class. To do that, it uses library utilities of the Common Language Runtime [2]. After the serialization, the process uses the method "callWHBASE" of the class ClsTst to access the storage service; the called method receives the XML data string and calls the method "callSendBufferSer" of the class WrapperHBASE to send the XML data string via an IP connection to java process ClientServer [12]. When the ClientServer receives the data, it uses the HTABLE utilities to write a new table with the new data in the scalable storage system, HBASE. Following the insertion of simulation data into the HBASE database, such data can be retrieved by means of queries, and shown in any web-browser, or used as an input to any post-processing library for further analysis or generation of plots and tables. In the following section we will extend this methodology in order to propose a more general methodology applicable to a unified domain of simulation/emulation.

3. EXTENSION OF MSDSM TO UNIFIED SIMULATION/EMULATION ENVIRONMENT

Using Tissue Methodology (TM) [21], this section proposes an extension of the MSDSM methodology, in order to use MSDSM in a unified domain of simulation/emulation for MANETs networks. Emulation is a hybrid approach that combines hardware and software where some components are implemented on real hardware (e.g. Motes, [8]) and some are simulated (e.g. links, traffic, etc.). Even if a large number of performance evaluation tools are available for a specialized class of MANETs, the Wireless Sensor Network (WSN), none of them simultaneously supports simulation, emulation and testbed implementation. Recent studies revealed that results obtained from simulation, emulation and testbeds were significantly different for the same scenarios [30]. Few studies have already realized the importance of an integrated tool that supports modeling, simulation, emulation and testbed implementation for algorithm validation, performance evaluation and proof-of-concept implementation in WSNs at different stages ([30]). At the same time experimentation is not yet a mature methodology in the context of ad-hoc networks: results are often non reproducible and hard to explain. In most cases it is nearly impossible to validate the measurements and to isolate external influences from the actual behavior of the investigated algorithm. Furthermore there are no benchmark settings and there exists no "best-practice" for conducting experiments. This makes it very hard to compare the results of experiments from different research groups. A significant effort to solve these problems is necessary to provide credible and comparable results and to encourage researchers to validate their ideas in real-world settings [33]. Some of the critical factors which rise from the merge between simulation domain, emulation domain and test bed, are listed below:

- the usability of the same data structures both for simulations on host PC and for implementation inside target devices;
- the usability of the same communications techniques both for simulations on host PC and for implementation inside target devices;
- the usability of the same architecture both for simulations on host PC and for implementation inside to target devices;
- the management of the processing nodes.

Tissue Methodology (TM) [22] aims to define an architecture that takes into account the previous critical factors. The methodology proposed in [22] emphasizes the following modelling paradigms:

- modular programming [34, 44, 23];
- patterns programming [27];
- events oriented programming [35];

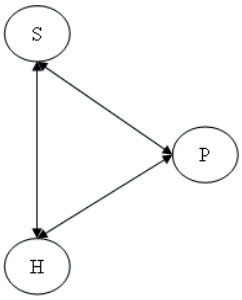

Figure 3: Basic Tissue Pattern.

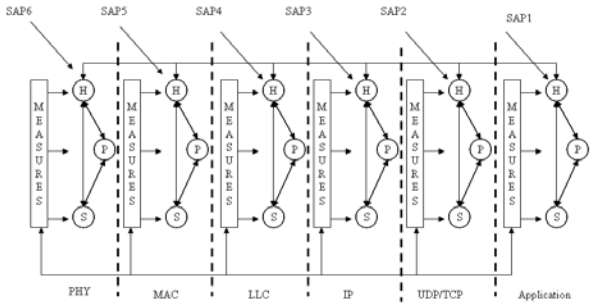

Figure 4: Use of the Tissue Pattern to model a Protocols stack.

- fractal programming [3];

Using TM, the design is run out by means of the design patterns that are named Tissue Patterns (TP), structured as shown in Figure 3. The TP has the aim of enabling the design by using three basic modules able to:

- receive and generate events (H);
- process events (P);
- storing a state space or other information (S);
- increase their "skills" through interaction with other units or reconfiguration.

The growth of tissues is achieved through the repetition of basic units, as well as of fractal structures; the link between H, S and P is represented by functional calls, an access to remote resources, or any communication protocol. The basic units can be used to build macro structures that can be in turn used for growing a tissue. Following this approach, a protocol stack can be rethought as shown in Figure 4 where a model made up of a basic TP is presented; each of the H modules receives events from other layers and, at the same time, it generates events towards other layers. Each event in each layer is processed by the P module. The data exchanged between two layers, or the data needed by a P module, are stored in the S module. It is worthwhile mentioning that each S module represents a system to store data (e.g. a bank of memories, a remote data source). The P module, or the H module, may retrieve a specific data set, through an identification code. Through this code, the module, which

ID	handle
ID1	handle1
IDN	handleN

Figure 5: Data set.

needs to use the data set, receives a handle; this handle enables the use of the data set in read/write mode as exemplified in Figure 5. Using the TM we have defined a hardware (HW)/software (SW) platform (see Fig. 6) that enables to reuse the MSDSM methodology in hybrid domain of simulation/emulation. The platform is formed by the following modules:

- P1, and P2: it is a processing unit able to execute C++ Language program and a Java Virtual Machine, so a Java language program [36];
- S: it is an storage embedded system (e.g. flash memory, SDRAM);
- H: it is an Arbiter Dispatcher; it has the aim to manages the classification of messages and to forward each message at correct addressee; the addressee may be bus arbiter or IO device
- H1, H2, H3, H4: they are bus arbiters;
- H5, H6, H7, H8, H9: they are IO device;

If we assume that the architecture of a single node in the network is developed using the basic Tissue Pattern, and if each element of each TP is able to generate an XML data string, in order to reuse the MSDSM is enough to link the SAP of MSDSM to each Basic TP, as shown in Figure 6. The AI and the SSM may be implemented using the Java Embedded [4], executable on a ARM processor, as depicted in Figure 7. This figure shows three half-planes; one contains the embedded storage system, S; another is devoted to the processing units, that is P1 and P2; the last is dedicated to Handler modules, or else H and H1-H7. P1 is devoted to execute the AI while P2 is in charge to execute the SAPs. Communications between different elements of the architecture are provided by the Message Passing Interface (MPI) [7]. There are many versions of this standard, and one of these supports real time operation [45]. By using Message Passing Interface, each element of TP, which belongs to simulation domain or emulation domain, may send XML data strings to AI therefore to SSM and finally to SSS. Furthermore, each basic TP may be moved from host PC to platform and viceversa.

4. CONCLUSIONS

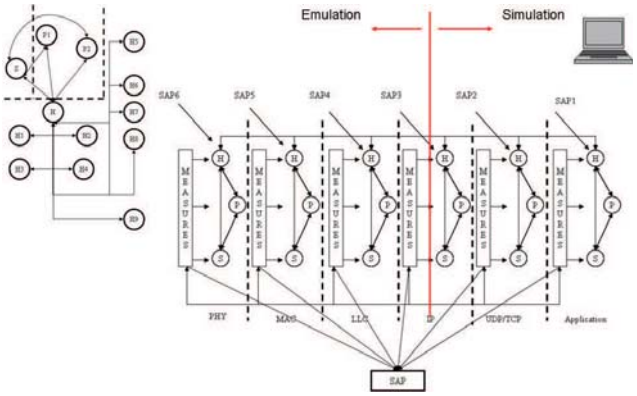

Figure 6: Simulation/Emulation domain.

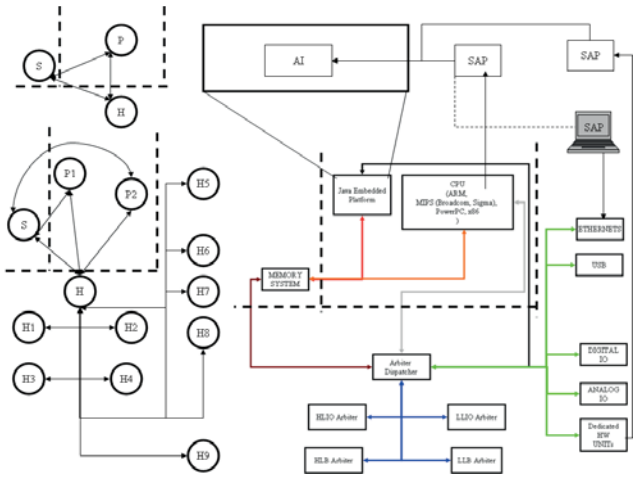

Figure 7: Extension via Tissue Methodology of MSDSM to emulation domain.

In this paper we have provided a methodology, MSDSM, to store simulation data in a distributed and scalable storage system and a list of the required tools to implement MSDSM. We have integrated all tools in a single tool chain; all the tools used are available as open source frameworks or packages. In order to verify the integrability of the tools, we have implemented the use case depicted in the Figure 2. Specifically, the SAP receives an XML data string and forwards it to an Acquisition Interface; inside that, the XML data string is formatted and it is sent to a remote System Storage Management. The SSM executes the following operations: it classifies the string, it selects a storing rule, it converts the string in a sequence of byte and stores it in scalable remote database. However, the SSM showed in the Figure 2 is a single thread application. In order to manage a simulation scenario which contains many "main" one for each protocol stack that is simulated , is need to have a multi thread SSM. As future work, we will develop a multi thread SSM application by means of Apache MINA [1] in our tools chain. At the current state, the MSDSM methodology is applicable to all MANET simulators using the C/C++ language such as OMNeT++ [10], OPNET [11], and ns-3 [9], and on every platform (Windows, Linux or Mac OS X) because the SAP is usable via JNI [5] from C/C++ code. Using the Tissue Methodology, we have extended the use of the MSDSM to simulation/emulation domains by means of a HW/SW architecture that was conceived for applications inside of the simulation/emulation domain; moreover, we have shown how to apply MSDSM on this architecture. Further developments are the implementation of an environment for concurrent execution of processes which models the protocols stack, using the JNI and JGROUPS [6], [36]. In addition we plan to evaluate the performance of the MPI application conceived for the simulation/emulation domain as described in [36] using the architecture of Figure 7.

5. ACKNOWLEDGMENTS

Luca De Nardis wishes to acknowledge the COST Action IC0902 "Cognitive Radio and Networking for Cooperative Coexistence of Heterogeneous Wireless Networks", FP7 Network of Excellence ICT-257626 ACROPOLIS "Advanced coexistence technologies for radio optimisation in licensed and unlicensed spectrum" and the Sapienza University of Rome project "Innovative Protocols and Technologies for Power Consumption Reduction in Telecommunication Networks" for their partial support to the work presented in this paper.

6. REFERENCES

[1] Apache MINA. `http://mina.apache.org/`.
[2] Common language runtime. `http://msdn.microsoft.com/it-it/library/8bs2ecf4.aspx`.
[3] Fractal programming. `http://fractal.ow2.org/documentation.html`.
[4] Java embedded. `http://www.oracle.com/technetwork/java/embedded/overview/index.html`.
[5] Java native interface. `http://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/functions.html`.
[6] Jgroups. `http://java-source.net/open-source/network-servers`.

[7] Message passing interface, mpi. http://www.mpi-forum.org/docs/docs.html.

[8] Mote, sensor node. http://www.sensor-networks.org/index.php?page=0932832814.

[9] ns-3 website. http//www.nsnam.org/.

[10] Omnet++ official website. http://newyork.ing.uniroma1.it.

[11] Opnet official website. http://www.opnet.com/.

[12] Package java.net. http://docs.oracle.com/javase/1.4.2/docs/api/java/net/package-summary.html.

[13] Scalable Network Technologies - Products - QualNet. Scalable Network Technologies Inc. http://www.qualnet.com/products/QualNet/index.html, 2004.

[14] R. Barr. *An efficient, unifying approach to simulation using virtual machine*. PhD thesis, Cornell University, 2004.

[15] R. Baumann, S. Heimlicher, and M. May. Towards realistic mobility models for vehicular ad-hoc networks. In *26th Annual IEEE Conference on Computer Communications IEEE INFOCOM 2007, MObile Networks for Vehicular Environments (Infocom MOVE'07)*, 2007.

[16] R. V. Boppana and S. P. Konduru. An adaptive distance vector routing algorithm for mobile, ad hoc networks. In *IEEE INFOCOM and Joint Conference of the Computer and Communications Societies*, volume 3, pages 1751–1762, 2001.

[17] J. Broch, D. Maltz, D. Johnson, Y. Hu, and J. J. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *ACM/IEEE International Conference on Mobile Computing and Networking*, pages 85–97, 1998.

[18] R. A. Caralli. The critical succes factor method: Establishing a foundation for enterprise security management. U.S. Department of Defense, Federal Government Contract Number F19628-00-C-0003.

[19] C. Chiang and M. Gerla. On-demand multicast in mobile wireless networks. In *IEEE International Conference on Network Protocols*, pages 262–270, 1998.

[20] D. R. Choffnes and F. E. Bustamante. An integrated mobility and traffic model for vehicular wireless networks. In *ACM VANET*, 2005.

[21] M. Colizza, M. Faccio, C. Rinaldi, and F. Santucci. A component-based architecture for protocol design and development in SDR frameworks. In *SDR'12 - WinComm*, 2012.

[22] M. Colizza, M. Faccio, C. Rinaldi, and F. Santucci. A methodology to design an advanced framework for efficient modelling and testing of manets. In *Wireless Telecommunications Symposium (WTS)*, pages 1–6, April 2012.

[23] P. David, M. Leger, H. Grall, T. Ledoux, and T. Coupaye. A multi-stage approach for reliable dynamic recon-figurations of component-based systems. In *8th IFIP Int. Conf. Distributed Applications and Interoperable Systems, DAIS 2008*, 2008.

[24] R. de Paula Herrera and A. S. Felinto. A distributed, multi-staged, high-throughput middleware for relational databases. In *IEEE/IFIP 7th Workshop on Business Driven IT Management (BDIM)*, 2012.

[25] N. Dimiduk and A. Khurana. *HBase in Action*. Manning Publications, 2012.

[26] K. Fall and K. Varadhan. The ns Manual (formerly ns Notes and Documentation). http//www.isi.edu/nsnam/ns/doc/index.html, January 2002.

[27] E. Gamma, R. Helm, R. Johnson, and J. M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[28] J. J. Garcia-Luna-Aceves and M. Spohn. Source-tree routing in wireless networks. In *IEEE International Conference on Network Protocols*, pages 273–282, 1999.

[29] L. George. *Hbase, The Definitive Guide*. O'Reilly, 2011.

[30] M. Imran, A. Md Said, and H. Hasbullah. A survey of simulators, emulators and testbeds for wireless sensor networks. In *Information Technology International Symposium (ITSim)*, volume 2, pages 897–902, June 2010.

[31] P. Johansson, T. Larsson, N. Hedman, B. Mielczarek, and D. M. Scenario based performance analysis of routing protocols for mobile ad-hoc networks. In *ACM/IEEE International Conference on Mobile Computing and Networking*, pages 195–206, 1999.

[32] D. B. Johnson and D. A. Maltz. *Dynamic Source Routing in Ad hoc Wireless Networks*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[33] W. Keiss and M. Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Networks*, 5(3):324–339, April 2007.

[34] K. K. Lau and Z. Wang. Software component models. *IEEE Transactions on Software Engineering*, 33(10), October 2007.

[35] K. Mani Chandy, M. Charpentier, and A. Capponi. Towards a theory of events. In *DEBS'07*, June 2007.

[36] Y. Maruyama and M. Itzkowitz. Profiling mpi application. http://www.oracle.com/technetwork/articles/servers-storage-dev/profilingmpi-301207.html.

[37] N. R. Mehya, N. Medvidoivc, and S. Phandek. Towards a taxonomy of software connectors. In *Proceedings of the 2000 International Conference on Topic(s): Computing and Processing (Hardware/Software)*, 2000.

[38] J. S. Otto, F. E. Bustamante, and R. A. Berry. Down the Block and Around the Corner? The Impact of Radio Propagation on Inter-vehicle Wireless Communication. In *29th International Conference on Distributed Computing Systems (ICDCS 2009)*, pages 605–614. IEEE Computer Society, June 2009.

[39] G. Pei, M. Gerla, and T. W. Chen. Fisheye state routing: a routing scheme for ad hoc wireless networks. In *IEEE International Conference on Communications*, pages 70–74, 2000.

[40] G. Pei, M. Gerla, and X. Hong. LANMAR: landmark routing for large scale wireless ad hoc networks with group mobility. In *IEEE/ACM MobiHOC*, pages

11–18, 2000.

[41] B. Raney, N. Cetin, A. Vollmy, M. Vrtic, K. Axhausen, and K. Nagel. An agent-based microsimulation model of swiss travel: First results. `http://citeseer.ist.psu.edu/raney03agentbased.html`, 2003.

[42] B. Raney, A. Vollmy, N. Cetin, M. Vrtic, K. Axhausen, and K. Nagel. Towards a microscopic traffic simulation of all of switzerland. In *International Conference on Computational Science*, pages 371–380, 2002.

[43] E. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *ACM/IEEE International Conference on Mobile Computing and Networking*, pages 207–218, 1999.

[44] S. Sicard, F. Boyer, and N. De Palma. Using Components for Architecture-Based Management: The Self-Repair Case. In *30th International Conference on Software Engineering (ICSE 2008)*, 2008.

[45] A. Skjellum, A. Kanevsky, and U. S. Dandass. Time message passing interface standard (mpi/rt-1.1. `http://www.mpirt.org`, November 2002.

[46] N. Zhou, H. Wu, and A. Abouzeid. The impact of traffic patterns on the overhead of reactive routing protocols. *IEEE Journal on Selected Areas in Communications*, 23(3), March 2005.