EXtensible Animator for Mobile Simulations: EXAMS

Livathinos S. Nikolaos
Department of Computer Engineering and Informatics
University of Patras
Patras, Greece
libathin@ceid.upatras.gr
www.livathinos.gr

Abstract— **One of the most widely used simulation environments for mobile wireless networks is the Network Simulator 2 (NS-2). However NS-2 stores its outcome in a text file, so there is a need for a visualization tool to animate the simulation of the wireless network. The purpose of this tool is to help the researcher examine in detail how the wireless protocol works both on a network and a node basis. It is clear that much of this information is protocol dependent and cannot be depicted properly by a general purpose animation process. Existing animation tools do not provide this level of information nor permit the specific protocol to control the animation at all. EXAMS is an NS-2 visualization tool for mobile simulations which makes possible the portrayal of NS-2's internal information like transmission properties and node's data structures. This is mainly possible due to EXAMS extensible architecture which separates the animation process into a general and a protocol specific part. The latter can be developed independently by the protocol designer and loaded on demand. These and other useful characteristics of the EXAMS tool can be an invaluable help for a researcher in order to investigate and debug a mobile networking protocol.**

Visualization tools; network simulator 2 (NS-2); mobile networking simulations

I. INTRODUCTION

In recent years there has been an increase in mobile devices with networking capabilities. As a result there is a need for the research community to examine and design network protocols and applications for mobile networks. Interestingly, much of this work is done with the help of network simulation environments.

According to an earlier survey [3] one of the most widely used network simulator is the Network Simulator 2 (NS-2) [6] and this trend has been enhanced since then. In a typical NS-2 usage scenario the user describes the network configuration and parameters (like mobility and traffic patterns), feeds them in as NS-2 input and after the simulation is finished NS-2 stores the simulation results in a text file being called the *trace file*. Usually trace files contain enormous amounts of data in text, encoded according to a given specification. Thus there is need for a tool that transforms the trace file to a visual playback of the simulation process.

This paper presents a visualization tool for NS-2 mobile simulations, called the *EXtensible Animator for Mobile Simulations* (EXAMS). EXAMS' goal is to help the researcher understand the mobile routing protocol with an exact visual representation of the NS-2 simulation outcome, while at the same time enabling him to control the animation process. It can visualize network layout, animate mobile nodes' movements, playback packets exchange at routing and agent level, give user the ability to freely move forward and backwards in simulation time and depict the network partitions according to given radio range. As regards data transmissions EXAMS can portray source and destination address, network level (routing or agent), direction, flow sequence and other. Moreover EXAMS keeps detailed statistics about data being sent, received, forwarded or dropped on a node and network basis.

However what makes EXAMS different from other NS-2 animators, like *iNSpect* [4] or *Huginn*[5], is its extensible architecture which permits the protocol designer to specify the information shown during the simulation playback and how the program responds to user events. It is obvious that each protocol has its own unique packet types, data structures, internal variables and states. For instance one protocol may use routing tables while another may not. Thus a proper visualization of the simulation process should be done according to the protocol's *interpretation* and it is the protocol developer's responsibility to do it the right way. In line with the above idea, EXAMS does not handle the visualization itself but provides the developer with a *visualization infrastructure* which enables him to complete the animation process through *protocol extensions*. As each extension is developed by the protocol designer himself, EXAMS makes it possible to depict a more detailed level of information like transmission details and values of node's internal data structures. This may include entries of routing tables, values for protocol's state variables, protocol specific packet types - in fact anything inside the trace file.

Additionally the visual environment of EXAMS incorporates many user friendly features like:

- Fast jump to a specific line of the trace file.
- Zoom in/out capability.
- Filtering of network traffic according to routing and/or agent layer.

This work is supported by Alexander S. Onassis Public Benefit Foundation with a scholarship for doctorate studies.

- Screenshots capture to external image file.
- Color and shape codes for data transmission.
- Extensive usage of keyboard shortcuts.

Figures 3 and 4 depict some screenshots of the EXAMS tool.

II. USAGE MODEL

The most outstanding feature of EXAMS is its ability to extend its functionality and incorporate new routing protocols. Figure 1 depicts a high level overview of EXAMS extensible architecture. We can see that EXAMS receives as input the NS-2 trace file, a proper implementation of the occupied routing protocol and the user's visualization preferences. The NS-2 trace file provides the simulation data which constitute the basis for the visualization, the routing protocol implementation encapsulates the semantics of the protocol which controls how the simulation data are presented and finally the visualization preferences are a simple way to handle additional animation issues like color preferences and file locations.

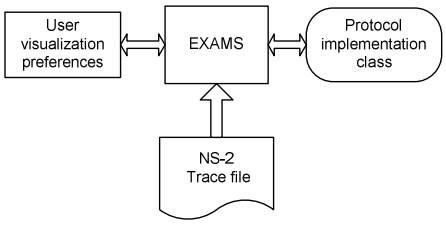


Figure 1. EXAMS architecture overview

As the trace file is the primary source for the visualization process, it is absolutely necessary to provide EXAMS with all the necessary data for a proper presentation of the protocol under examination. Fortunately the NS-2 layered architecture permits each protocol to add its own information to each related event inside the trace file. This enables the protocol designer to serialize protocol's related data and store them as a string representation. Then as the trace file is fed to EXAMS input, it is able to get back this information and visualize it.

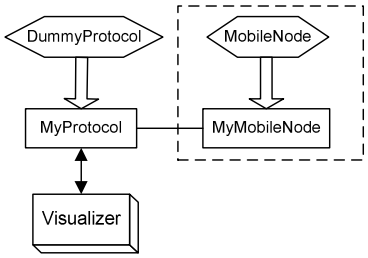


Figure 2. Protocol programming model

Programmatically a protocol implementation for EXAMS is a Java[2] class. As EXAMS parses the trace file it identifies the name of the routing protocol and expects to find a class file with exactly the same name under a specific classpath. Then this Java class is used to handle everything that is protocol dependent. Much effort has been put on making the programming of the protocol implementation class as easy as possible. As seen in Figure 2, the only thing needed is the subclassing of a couple of general purpose implementation classes. Additionally, EXAMS visualization services are exposed to the programmer through a single static object (Visualizer in Figure 2), relieving him from the complexity of EXAMS internals.

In a common usage scenario the following steps enable the user to fully exploit EXAMS capabilities:

- Export to NS-2 trace file all the information which is to be visualized by EXAMS like protocol's values or other NS-2 specific data not included by default.
- Develop Java code which handles the visualization details about the protocol specific issues.
- Load the trace file to EXAMS.

III. PERFORMANCE OPTIMIZATIONS

EXAMS has been designed to be a development tool, so it is important to provide a robust and efficient environment. One important performance improvement is the utilization of a network state caching scheme. This enables user to make fast event jumps and quick replays of previously seen simulation regions. Without caching and due to NS-2 trace file architecture, any transition to a random event requires the sequential application of every event starting from the initial network state up to the desired event. Obviously this is an unacceptable slow process. In order to eliminate cache misses it is important to make good predictions about user's next jumps. According to [1] we can adopt the *locality principle* on predicting the user's behavior and apply it by using a Least Recently Used (LRU) cache replacement algorithm. Locality principle implies that the user will tend to jump to event locations that are local to the current one either temporal or spatial. Temporal locality refers to events that the user had previously visited and spatial locality to events that are close to the current one. Therefore cache boosts EXAMS performance as regards event jumps and replays while at the same time keeps memory usage low.

Another optimization has to do with the trace file parsing. Trace files can be very large and thus make it impossible to be entirely loaded from disk to memory. On the other hand we need reasonable response times for file accessing. EXAMS makes possible to satisfy these constraints by the occupation of two in memory indexes. The lines index correlates each line of the trace file with its offset from the beginning of the file and the events index maps each EXAMS event with the corresponding line number. Thus when a specific event needs to be loaded, EXAMS initiates a two step process. First it uses the events index to find the corresponding line of the trace file and then uses the lines index to find the starting and ending byte of that line inside the file stream. This way we can fetch quickly a trace line without having to load the entire file into memory. Moreover an internal buffer eliminates the need for file accesses in cases of subsequent calls for the same event.

IV. CONCLUSION

In this paper we propose EXAMS as a visualization tool for wireless NS-2 simulations. EXAMS has been designed with the protocol developer in mind and provides a complete environment to study, test and debug wireless network protocols. EXAMS makes an exact visualization of the simulation process while at the same time provides all the available data produced by NS-2. Its unique characteristic is its extensible architecture which makes it possible to visualize network protocol's internal data structures (e.g. routing tables, state variables etc). Technically this is achievable through

protocol extensions which are developed by the protocol designer himself and are responsible to visualize protocol's data according to its own semantics. Besides this, a number of user friendly features and performance optimizations make EXAMS an efficient tool suitable for every day usage.

ACKNOWLEDGMENT

I would like to thank my advisor Professor Athanasios Tsakalidis for supporting and reviewing my work and Assistant Professor Xristos Makris for reviewing my work.

REFERENCES

[1] Denning J. Peter, "The Locality Principle", Communications of the ACM, Vol 48, No.7, July 2005.

[2] Gosling J. and McGilton H., "The Java Language Environment", 1996. A White Paper available at http://java.sun.com/docs/white/langenv (page accessed on June 2009)

[3] Kurkowski Stuart, Camp Tracy and Colagrosso Michael, "MANET Simulation Studies: The Current State and New Simulation Tools", Technical Report MCS-05-02, The Colorado School of Mines, 2005.

[4] Kurkowski Stuart, Camp Tracy and Colagrosso Michael, "A Visualization and Analysis Tool for NS-2 Wireless Simulations: iNSpect", Proceedings of the IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), 2005.

[5] Scheuermann Bjorn, Fubler Holger, Transier Matthias, Busse Marcel, Mauve Martin and Effelsberg Wolfgang, "Huginn: A 3D Visualizer for Wireless ns-2 Traces", Proceedings of the 8th ACM/IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM), October 2005.

[6] "The network simulator – NS-2". Available online at http://www.isi.edu/nsnam/ns/ (page accessed on June 2009).

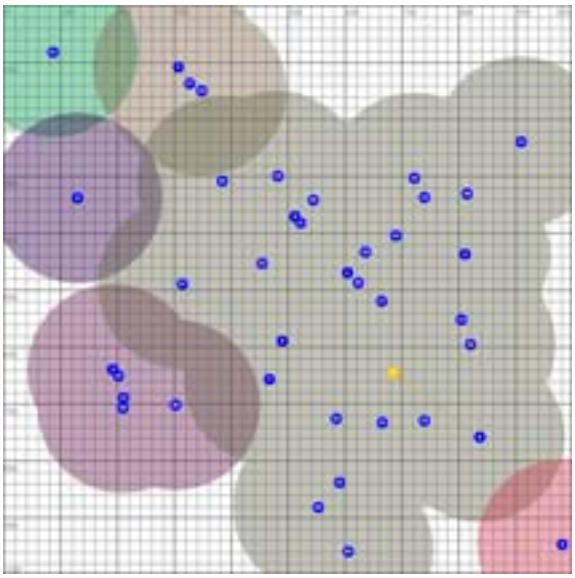
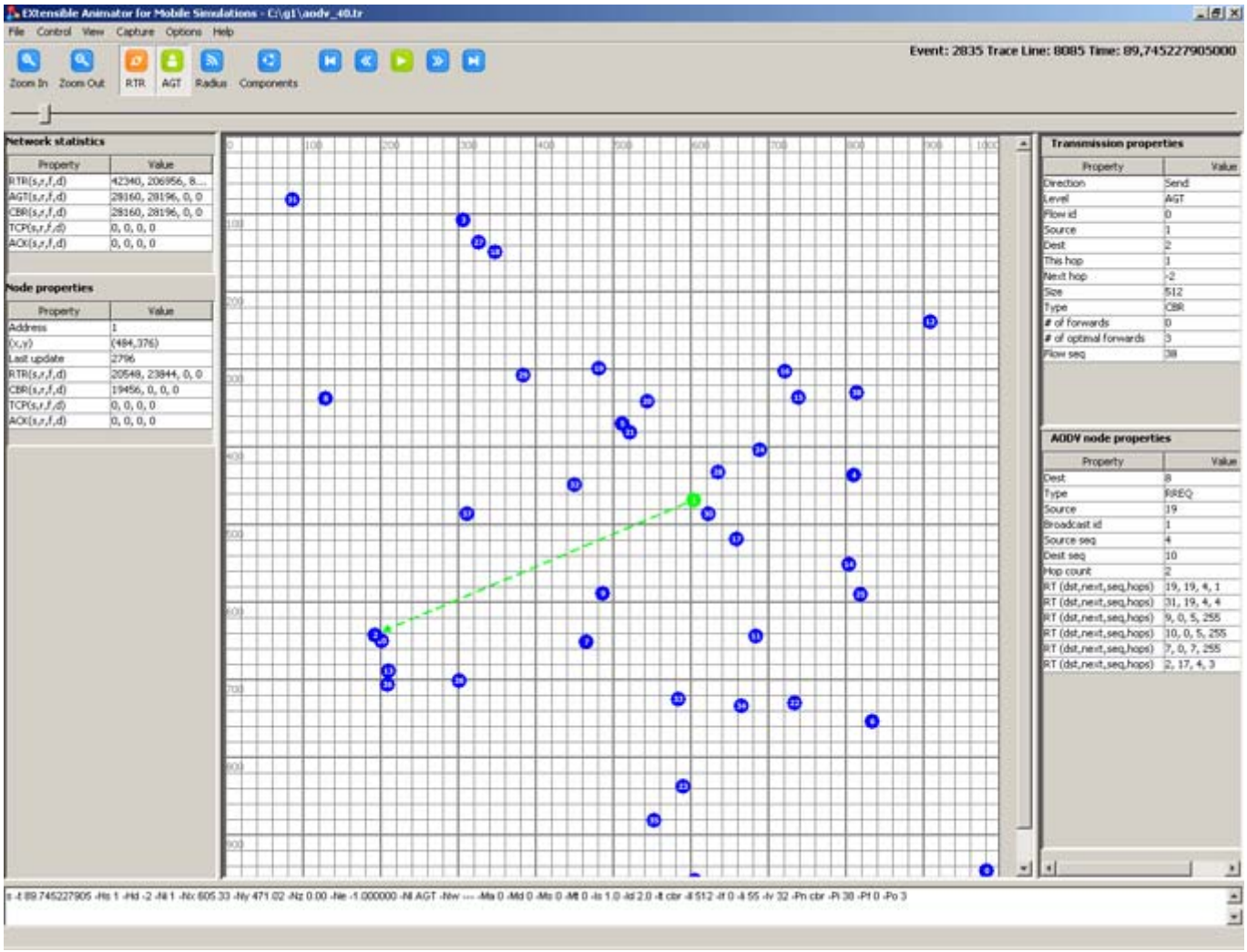


Figure 3. Network partitions



Figue 4. EXAMS Environment