Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

A GENERAL SIMULATION ENVIRONMENT FOR IP MOBILITY

Peng Sun
Sam Y. Sung

Department of Computer Science
National University of Singapore
10 Lower Kent Ridge Crescent
119260 SINGAPORE

ABSTRACT

This paper describes an advanced simulation environment that has been used to examine, validate, and predict the performance of Protocols for IP Mobility Support. It overcomes many limitations found in existing network simulators, and it provides more support on mobile-related issues. It contains several components that are common to all evaluations of IP mobility, which can model arbitrary network-topologies, arbitrary movement pattern, and arbitrary calling patterns. It also provides a set of protocol implementations that are necessary to simulating the Internet. The environment offers several desirable features including: 1) flexible metrics collection for both predefined and customized ones, 2) reuse of calling patterns, moving patterns, network topologies, etc. and 3) automatic generation of mobility pattern. Several research contributions had been made with the help of this simulation environment, and it would be useful for refining various aspects of IP mobility support.

1 INTRODUCTION

The increasing number of portable computers, combined with the growth of wireless services, makes supporting Internet mobility important. Many researchers have come to the conclusion that IP is the correct layer to implement the basic mobility support (Cheshire and Baker 1996). The greatest challenge for supporting mobility at IP layer is handling address changes. In other words, it is required to keep "un-interrupted" connections among nodes when they change their IP addresses during the movement. Many protocols have been proposed for IP mobility support, for example, Rajagopalan and Badrinath (1995), Chen and Lin (1996), Wang, Chen and Ho (1997), Ha´c and Huang (2000) and Perkins (1996).

A high quality simulation environment plays a key role in understanding existing proposals and evaluating new designs. This paper presents such a simulation environment for the eager needs. Quite a number of research contributions have been made with the help of this environment. Introducing the environment will share the simulation techniques we have used, and make it familiar to more researchers so that they can save much effort in experiments of IP mobility.

We have several significant contributions in building this environment. We identified and implemented several components that are common in almost all evaluations for IP mobility. We provide methods for reusing time-consuming and error-prone tasks such as specifying movements of mobile nodes, activity of source nodes, network topologies, network link properties, and so on. We use a simple while effective method to allow users modeling arbitrary calling patterns, movement patterns, and arbitrary network definitions.

The rest of this paper is organized as follows: Section 2 reviews current status for evaluating protocol proposals, and we show the limitations of existing network simulators. Section 3 presents our simulation environment in detail. We present the system features and their supporting architecture. Section 4 presents examples and case studies of the system, and section 5 summarizes this paper.

2 RELATED WORK

Proposal evaluations for IP mobility are far from enough to understand different aspects of them. The proposals are evaluated in different environments with different number of factors neglected. Their behaviors in the situations that do not conform to their assumptions are left unclear.

2.1 IP Mobility Protocols

IP mobility protocols are used to adapt IP address changes and make the changes transparent to the transport layers and higher layer protocols (Perkins 1998).

Although there are many methods and proposals, they share some common characteristics. Two addresses are assigned to a mobile node. One address is fixed (called home address), and the other is changeable (called care-of address). Home address is used to identify the mobile node on the network wherever they are connected. Care-of address reflects the current location of the mobile node, and it implies the geographical location of the mobile node on the network.

The biggest difference among different proposals is the methods to "map" the home address to care-of address. Some use mobile agents, some use gateway routers, some use hierarchy database, and some use broad casting. The mapping from home address to care-of address is called mobility binding. Mobility binding is the most difficult part of IP mobility support.

In general, the mobility binding is achieved through "search" and "update". Update is to propagate location information of mobile nodes into the wired network, while search is to utilize that information to find those mobile nodes.

2.2 Current Status of Proposal Evaluations

Little research work has been done on evaluating proposals for IP mobility support. Some protocols were proposed without any evaluation, for example, Chen and Lin (1996) and Prakash and Singhal (1997). Some proposals have evaluations, but their evaluations depend heavily on specific analytical models. For example in Wang, Chen and Ho (1997), the network is modeled as a grid, and the movement of a mobile node is modeled as purely random "*pop-up*" or a directional walk. In Rajagopalan and Badrinath (1995), the evaluation is done with the network modeled as a simple hierarchy, and the movement of the mobile node is modeled with random numbers that has negative exponential distribution. Some evaluations even impose specific limitations on the evaluations. For example, Chen and Lin (1997) limits the number of mobile agents within 19.

On the other hand, the metrics and self-defined parameters in simulations are quite different from one another, and the difference makes it difficult to compare two proposals based on their evaluation results. For example, Ha´c and Huang (2000) uses traffic models and call to mobility ratios (CMR) as its parameters, and the average cost savings, which is self-defined, as its metrics. Ratnam, Matta, and Rangarajan (1999) uses "average call arrival rate" as its parameter and "total cost", which is also self-defined, as its metrics. In order to compare the two proposals, it is necessary to conduct a certain kind of conversion between their parameters and metrics; however, the conversion is normally not easy. It is not trivial to convert "traffic model" into "average call arrival rate" or convert "total cost" into "average cost savings" because they are defined in different contexts.

In summary, current evaluations are far from insightful for us to understand previous proposals and compare them. They were only evaluated within special contexts, and it is unclear for a proposal to work in the conditions that do not conform to its assumptions. Finally, comparing two or more proposals requires translations among different analytical models, parameters, and metrics.

2.3 Existing Tools

A network simulator can provide unbiased evaluations if the simulator can precisely simulating IP packet delivery, because they do not rely on specific analytic models. Network simulators, in general, provide statistics that can reflect performance and other aspects of proposals being tested. It can avoid the metrics conversions since the statistics value always reflect same meaning regardless of different proposals.

Before we started developing this simulation environment, we have tried several well-known tools. Our experience showed that existing tools are not quite suitable for IP mobility support.

NS-2 (Network Simulator version 2) is a variant based on REAL (Keshav 1997), which is intended for studying the dynamic behavior of flow and congestion control schemes in packet-switched data networks. NS-2 is a powerful network simulator, and it has 60K lines of code plus 40K lines of test suites/examples. NS2 is not specially designed for IP mobility, although it addresses variety issues in network modeling. It is very difficult to simulate IP mobility, because it requires time-consuming works such as implementing network topologies. Moreover, it has little support on specifying mobility patterns, which influence the performance of routing protocols greatly. It is also difficult to debug an implementation of user-defined protocol, which occurs often in IP mobility simulation.

ParSec (PARallel Simulation Environment for Complex systems) is a C-based discrete-event simulation tool. (Bagrodia, Meyer, Takai, Chen, Zeng, Martin, and Song 1998) It adopts the process interaction approach to discrete-event simulation. It is focused on low-level event handling and process interaction. It more likes a programming language than a tool for parallel simulation. It is good at low level processing, however, it leaves a big gap between parallel processing and IP mobility simulation. It does not provide support for protocol suit of the Internet, not to say mobile-related issues.

GloMoSim (Zeng, Bagrodia, and Gerla 1998) is a scalable simulation environment for wireless and wired network systems developed on top of ParSec. Currently (version 2.0), it only support purely wireless network, which is only part of the simulation for IP mobility support because IP mobility involves many issues from the wired network such as router arrangement.

DaSSF (Calvert, Doar and Zegura 1997) is a C++ implementation of SSF (Scalable Simulation Framework) by Dartmouth University. The purpose of DaSSF is to achieve three desirable features on modeling very large-scale networks (e.g., the Internet.), namely capacity, scalability, and simplicity. It emphasize on handling large-scale networks rather than mobility modeling. It can simulate large networks, but it is difficult to in cooperate mobility patterns. Moreover, DaSSF requires users to implements the protocol suit of the Internet such as TCP, UDP, IP, and so on, and this requirement makes users frustrating when using it to simulate IP mobility.

SSF API is a base level API, and it defines five classes that are analogous to the Internet namely, Entity, Process, Event, InChannel, and OutChannel. An entity is a container of states; a process is the logic to change those states and handle incoming/outgoing events from InChannels and OutChannels. The entities are used to model network nodes, events are used to model data transmissions, and the InChannels and OutChannels are used to model network links. Through mapping InChannels and OutChannels and transmitting events within them, we can produce a similar effect of packet transmission.

SSFNet (http://ssfnet.org/) is a java-based implementation of SSF. It is better than DaSSF in sense that it provides a full set of Internet protocols in addition to the implementation of SSF API. Unfortunately, SSFNet is for wired networks, and it has little support for mobile networks. It does not provide mobile IP implementation, dynamically acquiring IP address by mobile nodes, no support for mobile agents/nodes, and no extension for user defined protocols.

3 SIMULATION ENVIRONMENT

Our simulation environment is implemented based on SSFNet 1.3.0 implementation, and it incorporates many desirable features for IP mobility simulation, which makes it distinctive from other network simulation environments.

3.1 Features

In general, the features we provided are useful for: 1) automatic and fast generation of mobility patterns that describe movement behavior of mobile nodes, 2) easily split/integrate user-defined protocols from/to the Internet protocol stack, and 3) reuse of network definitions including topology, node configuration, and link properties. Figure 1 shows a more specific comparison of our simulation environment and modern simulators.

3.2 System Architecture

Our system adopts a simple input→processing→output model. The input of our system is the user requirements and specifications on how to run a simulation. It includes environment specifications and protocol implementations (user defined protocols). The output of our system is the metrics collected from the simulation.

Simulators &. tools / Properties &. features	*NS2*	*Parsec*	*Glomsim*	*DaSSF*	*SSfNet*	*NetIDE*
Tools for mobility/calling pattern						✓
Multi-level programming						✓
Multi-level metrics collection	✓				✓	✓
User-defined metrics	✓	✓	✓	✓	✓	✓
System Protocols (TCP, UDP, IP.)	✓		✓		✓	✓
Mobile IP Implementation	✓					✓
User-defined protocols	✓					✓
Care-of address acquirement	✓		✓			✓
Platform independent					✓	✓
Extensible Network definition	✓			✓	✓	✓
Mobile related network entities	✓		✓			✓
Web-based user interface						✓

Figure 1: Comparison of Modern Simulators

During the simulation, a memory model is created by the runtime virtual machine, and the memory model is destroyed after the simulation. Metrics collection is done in parallel with running the simulation, and it can reflect the network status along the time line. Simulation time is used instead of clock time in order to increase the precision of time related properties such as bit-rate of a link. One simulation second is much smaller than one second in real life, and they have fixed proportions during the simulation. Their proportion can be arbitrarily defined before the simulation starts. Figure 2 illustrates the components in the system.

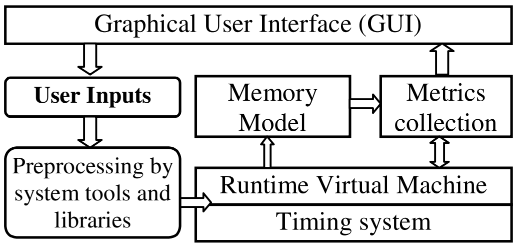

Figure 2: System Architecture

We alleviate user's burdens as much as possible. All user inputs are specified through a well-designed graphical user interface. Many works can be finished through mouse clicks. We also provide predefined system libraries to facilitate the specification of a given simulation, so that the user can focus on his design while avoiding as much repeating work as possible. For example, users can compose a large network through combining small networks in the system library. User can also adapt system source code in order to finish a protocol implementation faster. With these helps a user can finish a protocol implementation in as few as twenty lines source code.

3.3 Runtime Memory Model

The memory model is in fact a miniature of the network in real life. It contains network nodes, routers, and links between them. Each network node has its own protocol stack and TCP/IP implementations. Each router can make routing decisions based on real life routing protocols.

Figure 3 shows the memory model at runtime. All entities in the memory model are coordinated by the timing system, which triggers every time sensitive actions that include generating a call (a connection request from source node to a mobile node), generating movements of mobile nodes (dynamically change the network topology), and transmitting bit-streams at certain rates specified by the link.

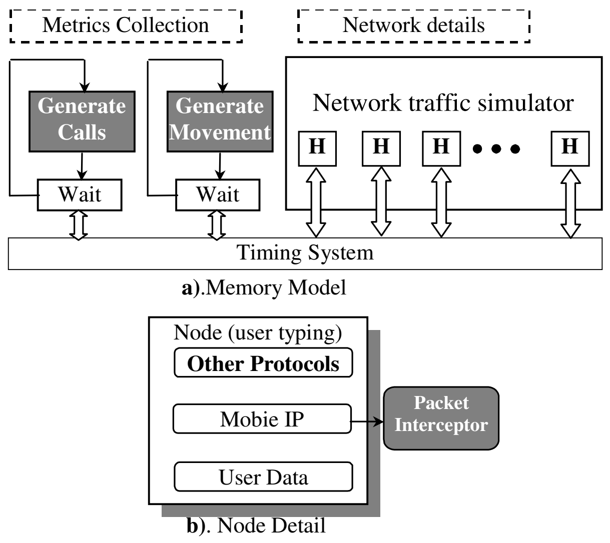

Figure 3: Memory Model at Runtime

The network traffic simulator simulates IP packet routing in real life, and all network nodes participate in the traffic simulation. Network nodes are autonomous in the sense that they make routing decisions at its own discretion. If the node is a router, it makes routing decision based on its knowledge of its surroundings, which is obtained through routing protocols. If the node is a normal host, it delivers packet to the default gateway according to its configuration. These actions are identical to real life networks.

Each node has its own protocol stack, which can also be arbitrarily specified by user before the simulation starts. The nodes in memory model also have the extra abilities for simulation purpose. Those abilities are:

1) *Assigning type information to each node for marking purpose*. For example if a stationary node frequently contacts a mobile node, the stationary node might be marked as "partner" of the mobile node, and it may have special update of location information from that mobile node. It enables an extra level of classification with regard to mobility issues.
2) *Allowing packet interceptor*. This enables user defined behavior being integrated to a specific protocol. For example, a packet Interceptor at IP can affect routing decisions. The interceptor is very flexible, and it can either employ existing implementation (e.g., IP) to fulfill normal tasks or change default behaviors. In other words, through packet interceptor, a developer can implement his own protocols for IP mobility support.
3) *Extra data storage.* This data storage represents the special knowledge that a node has for mobility support. For example, a node may remember location of certain mobile nodes like binding cache in mobile IPv6. A node may also record secrete associations in a given negotiation. The data storage is provided for protocol developers to store any information they feel proper, and the developers are totally responsible of interpreting the information.

Several techniques are use to reduce the size of memory model. The first technique is sharing. Sharing is used as much as possible. For example, although each node may have IP TCP, and UDP they share the same implementation. In other words, the routing for protocol processing is shared, while only data (i.e., input &. output of a protocol layer) are kept within a node. The second technique is virtual data packet. Although data packets are really transmitted among nodes, the data size is not necessarily very big. We only use several bytes to describe a packet, if the packet content is "pure data", which means it contains data that are not interested for IP mobility support. For example, to simulate a 10KB-packet, we only need to mark this packet size as 10KB, and we need not allocate 10KB in the memory. During the transmission, we simulate the delay for transmitting certain number of bits from one node to another, and we remaining bits that are queued for next frame.

3.4 Modeling Network Details

Define a high quality network is not an easy task. In our context, network definition has to contain at least two parts, 1) logical topology of the network node and their connections, and 2) relationship of network nodes concerning their geographical locations. The first part is necessary for correct IP routing, and the second part is necessary to restrict the movement of mobile nodes. The second part is specific to mobility issues, because the movement of mobile nodes is geographically related.

Although DML (Domain Modeling Language) is an excellent language for defining network topology, it is not designed for mobile network. It cannot describe entities in mobile network such as mobile agents and AAA servers, and it cannot describe certain properties such as capacity of a foreign agent and its address pools. Moreover, it cannot describe the geographical relationships among nodes.

With regard to the limitations of conventional DML language, we developed our extensions in order to support mobile-related entities and properties. We defined an extended DML format based on conventional DML format.

Figure 4 illustrates the procedure of producing memory model from DML files. The procedure accepts two types of inputs: 1) patches that describe modifications on existing DML files and 2) enhanced DML files.

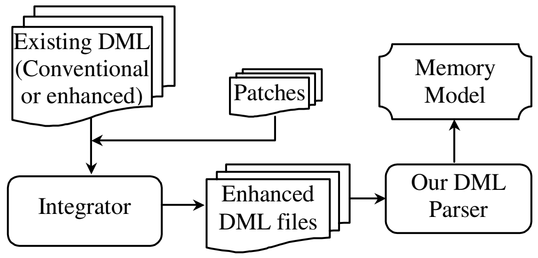

Figure 4: Extensions to Existing DML Models

The method of producing memory model greatly relieves burdens of users, while giving them enough freedom of specifying any details. First of all, describe a network in details is time consuming and error prone because it involves too much information. It would be very difficult to describe a large network from scratch. With patches, we could use existing tested DML libraries, which are provided by our simulation environment, to build robust networks with little effort. Secondly, we give the user full freedom of building their networks from scratch, because our DML parser can directly accept "Enhanced DML files".

3.5 Movements of Mobile Nodes

The movements of mobile nodes are modeled as consecutive moving and staying, as shown by figure 5-a. A single step of movement of a mobile node is modeled as assigning a new *point of attachment* to it. A single step of staying is modeled as assigning a *staying time* to that step of movement. A developer can use a plain text file to store the movements of a mobile node.

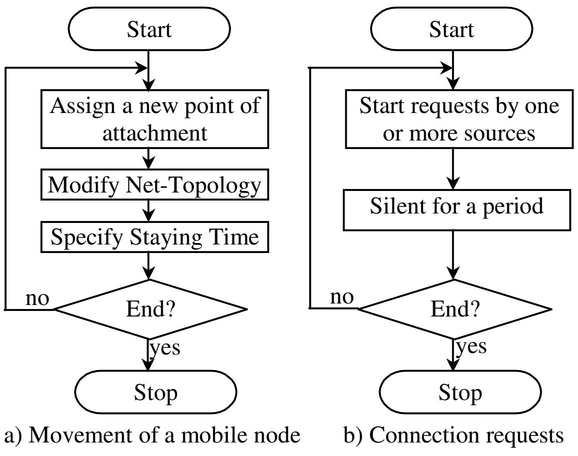

Figure 5: Movements and Calls

The file is composed of lines of text (format is shown in appendix A). Each line represents a single step of movement, a control command, or a variable definition. If it is a single step of movement, then it specifies a point of attachment with a staying time. If it is a control command, it will be one of *if-goto, stop, offline, and online.* "*if-goto*" enables a developer reusing a sequence of movements through specifying a simple logic. *Stop* will make the mobile stop there until the simulation ends. A stopped mobile node can still send and receive packets. In contrast, *offline* will forbid a mobile node unable from sending and receiving packets. The node is temporally "*disappeared*" from the simulation. *Offline* takes one parameter as the duration of offline. An online statement can only appear immediately after an offline statement. *Online* takes one parameter, and the parameter specifies a line number. The statement at that line must be a *movement* line. The point of attachment in that line specifies where the mobile node is re-connected to the network.

3.6 Calling Patterns by Source Nodes

The connection requests from source nodes are modeled from the mobile nodes' point of view. For a mobile node, at given time, it may receive zero or more requests from different source nodes. For each given mobile node, the connection requests are modeled as consecutive *connection requests* followed by a *silent* period that can be zero second, as shown by figure 5-b. During the silent period, no nodes are sending requests to that mobile node. During the period of generating requests, one or more source nodes generate connection requests to the mobile node. These connections are deemed as happening in the same time.

The connection requests can be stored on hard disk similar to the storing of movements of a mobile node. We call the file "calling-pattern file". The number of calling-pattern files is equal to the number of mobile nodes in a given simulation. One mobile node is mapped to one calling-pattern file. In fact, a "calling-pattern file" can also be a record in a database. When the number of mobile nodes increases, calling-patterns results in a database with a number of records.

3.7 Implementing User-Defined Behaviors

IP mobility support can be described as "*mobile nodes propagate location information into the network and source nodes utilize that information to locate those mobile node*". A proposal may contain many modifications on existing network (Pitoura and Samaras 1998).

In order to support different user defined behaviors we support the following modifications on the network memory model: 1) associate user-defined data with a node. It enables users assign "state", "conditions" or any information that is used for making decision, 2) associate user-defined data with IP packets. It enables the information propagation, and 3) allow user alter routing decision and packet content at any node. It gives the freedom of altering the behavior of IP layer at both network and node level.

In proposals for IP mobility support, it is very common to develop a new protocol or modify existing protocols. It requires supporting arbitrary modifications on implementations provided by our system. It is achieved through protocol interceptors. A protocol interceptor alters the behavior, inputs and outputs of a given protocol, and interceptors can be dynamically attached or detached. For example, an interceptor can be attached to mobile IP to alter its routing decisions, as show in figure 3-b. All protocol implementations provided with our simulation environment can be attached with interceptors.

3.8 Multi-Level Programming

In order to make users' implementation as easy as possible, we provide many predefined routines that hide many details when programming. For example, when user want to pass some data to a given node through IP, he can just calling system routine by given the requirement parameters, i.e., source node, destination node, and the data object. The user will not be involved in fragmentations and payload formats. On the other hand, we allow user to directly compose an IP packet, thus it give full freedom to users to define content in the packets.

The routines are source codes that are automatically generated by the system with the parameters given by the user. In other words, the system helps user with some "coding" tasks. Users have full access to the generated code, and they can also code everything in details.

The flexibility results in multi-level coding techniques. When users want to know the details, they can code the details, when users do not focus the details, they can use system routines that helps to generate code. It saves much effort of user when implementing user-defined behavior.

3.9 Flexible Metrics Collection

The metrics are collected at different levels. We can collect metrics at network level, for example, the total packets generated on the network. We can also collect metrics at node level, such as the number of packets delivered by this node. We can also collect metrics on a single protocol layer of a give network node, for example, we might be interested in workload of IP rather than other protocols.

Besides the different level of metrics collection, we also support user-defined metrics. It is achieved through adding user-defined monitors. Monitors can read the state of a network, a subnetwork, a node, and a protocol layer within a node. Monitors are similar to interceptors except that they cannot modify the behavior of protocols and nodes.

4 EXAMPLE AND CASE STUDIES

In this section we use some examples, and case studies to show the functionalities of our simulation environment. Due to page limit, we only show a small fraction of available examples, and a more comprehensive list can be found within our software package. We show the efficiency of time management, reuse of existing components, and a real test for performance evaluation.

4.1 Advanced Time Management

Our simulation environment uses *simulation time* rather than real time. Using simulation time can greatly speed up the simulation while preserving the same precision of using real time. For example, transferring 100M bits through a 100Mb network card will be simulated in *1 simulation second*, and *1 simulation second* is much smaller than 1 real second.

We also provide advanced synchronizations among events even the simulation is executed in a one-CPU computer. We assure that in the simulation world, those events that happened in the same simulation time will be treated with the same simulation time-line, although actually they might be generated in order.

Besides the time synchronization, we employed the parallelism in SSFNet. The time cost of 14 simulations with different number of events on a Pentium III 450 with 128MB memory is shown in figure 6. The time cost of a simulation is not greatly affected with a fast increase in the number of events. On the contrary, the time cost of a simulation is increased rather slowly. The parallelism makes it not time-consuming.

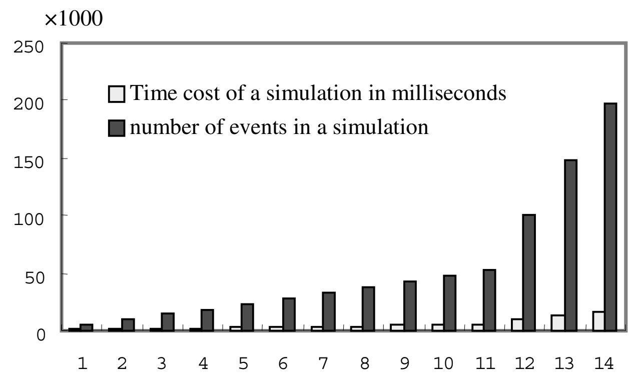

Figure 6: Time Cost with 14 Different Simulations

4.2 Use System Libraries

Using system library can greatly simplify the process of defining a network. We can compose large network through small networks. The following code combines three small nets:

```
Use <simpleNets.dml>
Net [
 Net [id 0 _extends .simpleNets.ring]
 Net [id 1 _extends .simpleNets.star]
 Net [id 2 _extends .simpleNets.tree]
 router [id 3 _extends sys.router.Interface3]
 link [attach 0:0(1) attach 3(0) delay 0.01]
 link [attach 1:0(1) attach 3(1) delay 0.01]
 link [attach 2:0(0) attach 3(2) delay 0.01]
]
```

We use router 3 combines the three small nets into a bigger net. The protocol stack of router 3 is directly obtained from system library. If we specify every node within every subnetwork, we could end up with at least one hundred lines of DML code.

4.3 Reuse Existing Components

In the above example, we also reused the file "simpleNets.dml", which defines three simple networks named ring, star, and tree, by a keyword "use" at the beginning of the network definition.

Reusing an existing DML file not only means we can reuse network definitions but also enables us to use a conventional network model In conventional network model, there is no entities to support mobility such as home agent and foreign agent, but it contains a good definition for a wired network. We can adapt a conventional model to meet the requirements with IP mobility support. Here is an example that add an home agent (type 0) and foreign agents to a network defined in "myNet.dml"

```
Use <myNet.dml >
Assign "3:4" type 0 #home agent
Assign "2:1" type 1 #foreign agent
Assign "2:6" type 1 #foreign agent
```

Network node 4 in sub network 3 is assigned as a home agent for a mobile node. Nodes 1 and 6 in sub network 2 have been assigned as foreign agents. By assigning different roles, we attach special ability and property to the node. For example, being a foreign agent will enable the node with accept mobile node registrations.

The adaptation is equal to directly coding a DML source. For example, the assign home agent "3:4" is equal to modify the DML definition of node "3:4". It equals to add "mobile_related" section to the router 4 at net 3, as shown by the following code.

```
router[ id 4 nhi_route
  [dest default
   interface 0 next_hop 0(3)]
   _extends .dictionary.client10Mb
 mobile_related [ host_type 0
  host_type_description HomeAgent]
]
```

Directly modifying the file "myNet.dml" may have some disadvantages. First, direct modification will damage the robustness of existing DML files, because it has a high risk to introduce errors into them. Second, direct modification can hardly be "split" from the original source code, and thus it is hard to "rollback" to original version. Finally, modify existing DML code here and there increases the complexity of composing large networks, because the author cannot have a clear picture of where the DML code has been changed.

4.4 Fair Comparison

We show a simulation that we have done as an example of using our simulation environment. More detailed illustrations can be found in Sun, Sung, Li, and Huang (2002). We simulated mobile IP and static update protocol with *identical* network details, calling patterns, and movement patterns. With identical environments, we ensure the two protocols are compared *fairly*. The simulation compares the total cost of the two protocols in terms of traffic load of the network. Traffic load in the simulation is defined as number of total hops of all packets.

We change the call-to-mobility ratios (CMR) from 1 to 11, and the results are shown in figure 7. Figure 7-a shows the total search cost in terms of number-of-hops. It shows that static update have significant savings in search. In contrast, figure 7-b shows the total update cost, and static update has much higher update cost than basic mobile IP. It is because the protocol updates patron nodes in addition to the home agent. Although the update cost is quite high, the total node (search + update) of static update tends to be much smaller than basic mobile IP when CMR is increased, as shown in figure 7-c. Figure 7-d makes it clearer that the savings of total cost are quite high.

5 SUMMARY

We have introduced a simulation environment for the evaluation of the protocols for IP mobility support. We showed its features and its system architecture that support those features. It overcomes the limitations that were found in existing network simulators. It provides modeling methods of calling patterns and movement patterns, which can be generated based on their storage format. It provides reuse of many logical components, and that saves much effort in performance evaluation. It enables user to avoid being involved much details while giving them enough freedom by multilevel programming. It provides flexible metrics collection method, for they can be done at different levels, and user can specify the own metrics through protocol monitors. We sincerely hope our simulation environment become a popular tool for protocol developers, especially for those interested in IP mobility support.

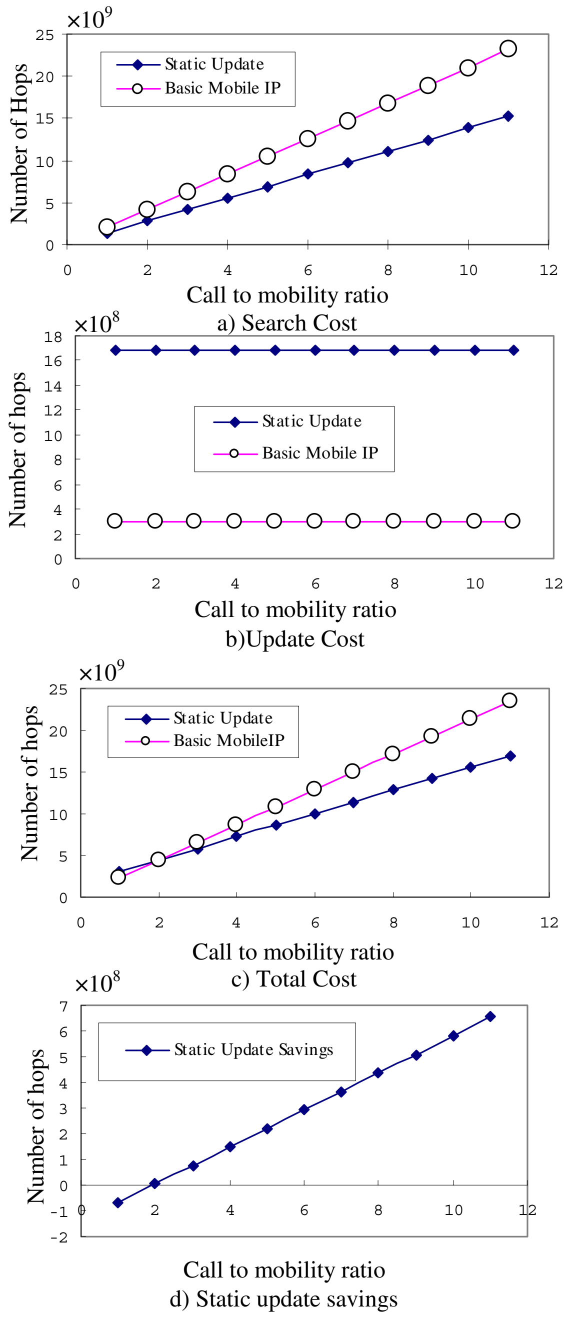

Figure 7: Simulations Results of Comparing Two Protocols

REFERENCE

Bagrodia,R., R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, and H. Y. Song, 1998. Parsec: A Parallel Simulation Environment for Complex Systems. IEEE Computer, Vol. 31(10), pp. 77-85.

Calvert, K., M. Doar and E. Zegura, 1997. Modeling Internet Topology, IEEE Communications Magazine.

Chen, W. and E. Lin, 1996. Routing optimization and location updates for mobile hosts, In the proceedings of 16th IEEE International Conference on Distributed Computing Systems (ICDCS'96).

Chen, W. and E. Lin, 1997. Dynamic location control for mobile nodes. Technical Report 97-CSE10, Southern Methodist University, pp11-16.

Cheshire, S. and M. Baker, 1996, IP Mobility 4x4, In Proceedings of 1996 SIGCOMM Annual Technical conference (SIGCOMM'96), 218-329.

Ha´c, A. and Y. Huang, 2000. Location Update and routing scheme for a mobile computing environment, International Journal of Network Management 10:191-214.

Keshav, S., 1997. REAL 5.0 Overview, Online documentation of REAL network simulator Cornell University, available online via <http://www.cs.cornell.edu/skeshav/real/overview.html> [accessed: July 12, 2002]

Liu, J. and D. M. Nicol, 2001. DaSSF 3.0 User's Manual, available online via <http://www.cs.dartmouth.edu/research/DaSSF/papers/dassf-manual-3.0.ps> [accessed, July 11, 2002]

Perkins, C.E., 1996 Mobile-IP Local Registration with Hierarchical Foreign Agents Internet Draft, Internet Engineering Task Force.

Perkins, C.E., 1998. Mobile IP, Design Principles and Practices. Wireless Communications Series. Addison-Wesley. ISBN 0-201-63469-4

Pitoura, E. and G. Samaras, 1998. Data Management for Mobile Computing. The Kluwer International Series on advances in database systems. Kluwer Academic Publishers, 1998. ISBN: 0-7923-8053-3

Prakash, R. and M. Singhal, 1997. Dynamic Hashing + Quorum = Efficient Location Management for Mobile Computing Systems, In Proceedings of the 6th annual ACM symposium on Principles of Distributed Computing.

Rajagopalan, S. and B. R. Badrinath, 1995. An adaptive location management strategy for mobile IP. In Proceedings of International conference on mobile computing and networking (Mobicom'95), 170-180

Ratnam, K., I. Matta and S. Rangarajan, 1999. A Fully Distributed Location Management Scheme for large PCS. Technical Report BU-CS-1999-010, Computer Science Department, Boston University, Boston, MA 02215

SSF Research Network, 1999. Domain Modeling Language Specifications, available online via <http://www.ssfnet.org/SSFdocs/dmlReference.html> [accessed July 12, 2002]

Sun, P., S.Y. Sung, Z. Li, and S. Huang, 2002. Performance Evaluations and Analysis of protocols for IP mobility support: A Quantitative study, In Proceedings of IEEE 35th Annual Simulation Symposium (SS'02): 219-226

Wang, Y., W. Chen and J.S.M. Ho., 1997 Performance analysis of mobile IP extended with routing agents. Technical Report 97-CSE-13, Department of Computer Science and Engineering, Southern Methodist University

Zeng, X., R. Bagrodia and M. Gerla, 1998. GloMoSim: A Library for the Parallel Simulation of Large Scale Wireless Networks, In Proceedings of Parallel and Distributed Simulation Conference (PADS'98).

APPENDIX A: MOBILITY PATTERN FORMAT

Mobility pattern describes the moving behavior of a mobile node. The pattern is stored as simple plain text file that support simple control flows. When there are many mobile nodes, each pattern file can be stored as one record of a database. The record or the pattern file shares the same format, which can be loosely defined by BNF as:

```
Pattern::= <line> <CRLF>|<lines>
<CRLF>::= char(0D)char(0A)
<lines>::= <line><CRLF>|<line><CRLF><lines>
<line>::= <movement>|<command>|<var_def>
<movement>::= <New attach point>, <time>
<New attach point>::= NHI address
<time>::= <number>
<number> ::= 1,2,3,4,5 .. ..
<command>::= <if_goto>|stop|
             <offline>|<online>
<if_goto>::= if <exp> goto <line number>
<exp>::= <variable name> <bop> <number>|
         <Random> <bop> <number>
<bop>::= > | = | <| >= | <= | <>
<var_def>::= <variable name><aop><number>|
           <variable name> = <Random>
<aop>::= += | -= | *= | /=
<Random> ::= RND (<from>, <to>,<type>)
<offline>::= Offline (<time>)
<online>::= Online (<line number>)
```

AUTHOR BIOGRAPHIES

PENG SUN is a Ph.D. candidate at Department of Computer Science, National University of Singapore (NUS); Singapore. He received his B.Eng and M.Eng from Harbin Institute of Technology (HIT), P.R.China. He is a member of ACM, and his research interest includes Next Generation Networking, Web, and database technology. His email address is <sunpeng1@comp.nus.edu.sg >.

SAM Y. SUNG is an Associate Professor at Department of Computer Science, National University of Singapore (NUS); Singapore. He received B.Sc. from National Taiwan University in 1973, M.Sc and Ph.D in computer science from University of Minnesota, in 1979 and 1982, respectively. He was with University of Oklahoma and University of Memphis in USA, before joining NUS in 1989. His research interests include information retrievals, data mining, pictorial databases and mobile computing. He has published extensively in various conferences and journals, including IEEE Transaction on Software Engineering, IEEE Transaction on Knowledge &. Data Engineering, etc. His email address is <ssung@comp.nus.edu.sg>, and his web page is <http://www.comp.nus.edu.sg/~ssung>.