

Building the Blocks of Protocol Design and Analysis — Challenges and Lessons Learned from Case Studies on Mobile Ad hoc Routing and Micro-Mobility Protocols

Fan Bai
Dept. of Electrical Engineering
Univ. of Southern California
Los Angeles, CA 90089
+1-213-740-2685
fbai@usc.edu

Ganesha Bhaskara
Dept. of Electrical Engineering
Univ. of Southern California
Los Angeles, CA 90089
+1-213-740-2685
bhaskara@usc.edu

Ahmed Helmy
Dept. of Electrical Engineering
Univ. of Southern California
Los Angeles, CA 90089
+1-213-821-1329
helmy@ceng.usc.edu

ABSTRACT

With the emergence of new application-specific sensor and Ad-hoc networks, increasingly complex and custom protocols will be designed and deployed. We propose a framework to systematically design and evaluate networking protocols based on a 'building block' approach. In this approach, each protocol is broken down into a set of parameterized modules called "building blocks", each having its own specific functionality. The properties of these building blocks and their interaction define the overall behavior of the protocol. In this paper, we aim to identify the major research challenges and questions in the building block approach. By addressing some of those questions, we point out potential directions to analyze and understand the behavior of networking protocols systematically. We discuss two case studies on utilizing the building block approach for analyzing Ad-hoc routing protocols and IP mobility protocols in a systematic manner.

Categories and Subject Descriptors

C.2.2 [Computer-Communication Networks]: Network Protocols

General Terms

Performance, Design

Keywords

Building Block, Protocol Design, Protocol Analysis, Mobile Ad Hoc Network, Micro-Mobility

1. INTRODUCTION, MOTIVATION AND CHALLENGES

Due to the ubiquity of small, inexpensive wireless communicating devices, wireless sensor and ad-hoc networks are emerging as new fields of networking. These networks are application-specific, and often require custom protocol stacks and network components to achieve their objective efficiently. Traditional protocols, including Internet protocols, were designed based on experience and feedback from implemented systems, rendering the design and evaluation of networking protocols costly and time-consuming. For the newly emerging application-specific networks, designers cannot obtain such feedback. Moreover, most of these systems cannot be easily upgraded or modified once deployed. Hence, changing the

software or hardware for correcting design errors or improving performance is either impossible or very expensive. The lack of systematic design, evaluation and test methodologies is becoming a major concern for protocol designers with the increase in protocol complexity. A systematic methodology or tool to analyze, fine-tune or synthesize protocols from reusable parameterized components to meet performance requirements would be ideally suited for the new networking paradigms. The same methodology could also be extended to the design and evaluation of Internet protocols.

One simple approach to develop such systematic methodology may be to provide a library of protocol mechanisms that can be re-used. Even if these protocol mechanisms are relatively well understood and simple in isolation, reusing the library of mechanisms may prove to be difficult due to the complex interaction between the various distributed mechanisms, which depends on the environment in which they are deployed. Effective reuse of library components requires a systematic way in which the protocol composed of such modules can be designed, tested and evaluated across the scenarios under which it is expected to operate. This requires explicit modeling of (a) the protocol mechanisms, (b) their interactions and (c) the effects of the environment, including the physical phenomena sensed, mobility, wireless channels, among others.

Note that this library-based approach may be used to address several problems. We identify two main problems in the science of protocol design: (a) protocol synthesis and (b) protocol analysis. In protocol synthesis, one may define a high level functional requirement that should be achieved using a combination of the library mechanisms. Although this protocol synthesis problem is quite challenging and interesting, we plan to consider it in our future work and we do not address it in this paper. In protocol analysis, on the other hand, an (initial) protocol is given and the goal is to develop deep, micro-level, understanding of its performance and limitations over a vast array of operating conditions. The insight developed through this understanding helps in refining existing protocols through an iterative process. This protocol analysis problem is the focus of this paper.

As an attempt to address the above problem, we propose a 'building block' based framework in which the protocol is broken down to its constituent mechanistic building blocks. The break down is based on functionality, thus resulting in functionally separated modules. The modules along with their

interaction represent the overall protocol behavior. By decomposing the protocol into this set of mechanistic building blocks, we hope to convert the complex problem of modeling the overall protocol into a set of simpler sub-problems of modeling the building blocks and their interaction. By modeling both the blocks and their interaction, we aim to develop a library-based tool by which protocol can be designed, analyzed and evaluated systematically.

Integrated with this tool would be the environmental conditions and effects, such as wireless channel models and mobility. We also propose to consider these environmental effects as consisting of building blocks. This facilitates traversal of various dimensions of the spaces of operational conditions, to provide rich, meaningful, evaluation scenarios.

In order to be able to evaluate the utility of our building block approach for protocol design, evaluation and analysis, we need to understand the following challenges:

- (1) In general, how do we define and represent the building block modules and the interaction between them to make them amenable to the required analysis?
- (2) How do we break down a given protocol into its constituent building blocks, and how do we organize the set of building blocks back into a protocol?
- (3) How do we model the underlying environments, where the protocol is expected to be deployed, in a systematic manner?
- (4) How do we use the building block approach for the design, analysis and refinement of various protocols for a given environment?

In this paper, we present an attempt to define the building block modules and their interaction in a formal way by capturing their unique functionality and key characteristics. Based on these fundamental definitions, we further propose a hierarchical building block framework to model the protocol at different levels of abstraction. In such a hierarchical structure, the building blocks at each level may be refined using more detailed building blocks successively. Considering that the underlying environment usually plays an important role in affecting the protocol behavior and performance, we also present a scheme to model the environment in a systematic way.

To demonstrate the utility of our building block approach for protocol evaluation and analysis, we present two case studies on analyzing wireless networking protocols. In the first we study classes of ad hoc routing protocols and in the second we study classes of micro-mobility protocols. We show that by using our approach, we develop a deep understanding of the interplay between the parameterized protocol mechanisms and the underlying environment. Several interesting lessons about the design choice of protocol mechanisms and the generation of evaluation scenarios are presented and discussed. For example, in MANET reactive routing protocols (e.g., AODV[17], DSR[18]), flooding and caching seem to have a great effect on performance, while salvaging in DSR barely seems to have an effect on the protocol performance.

The purpose of this paper is not to provide a complete solution for the protocol design and analysis problem. Rather, it is to discuss the various problems faced in designing new protocols for wireless networks, and to identify and clearly define a set of problems and research questions that need to be addressed in order to realize a more comprehensive solution.

In that sense, this paper attempts to address the challenges in building the blocks of protocol design and analysis and discuss the potential directions.

The remaining of this paper is organized as follows. Related work is discussed in Section 2. Section 3 gives an overview of our hierarchical building block approach, and describes the model for building blocks and channels. A method to model the underlying environment in a systematic manner is given in Section 4. In section 5, the method to design, analyze and refine the protocols through the hierarchical building block approach is briefly discussed. Two case studies on ad hoc routing and micro-mobility protocols, and the lessons learned are discussed in Sections 6 and 7. We discuss some open questions in Section 8 and conclude in Section 9.

2. RELATED WORKS

The building block methodology itself is not a new concept in the field of distributed systems. The Internet is the most obvious example of a system based on layered building blocks. The layers of the protocol stacks make up the building blocks of the Internet. Each layer has well defined functions and interfaces and one layer makes no assumption about the internals of the other. This open architecture enables one layer to perform seamlessly over the other as long as their interfaces match. However this transparency comes at the cost of potential duplication of functions at various levels. Furthermore, the individual layers themselves are not designed or implemented with any explicit layering or components. Thus the design of each of the layers itself is complex. One of the reasons for the seamless performance of the various layers is that the Internet protocol stack has enjoyed unprecedented success and has been used by millions over decades thus flushing out bugs in both designs and implementations by sheer brute force. However, the new application-specific stacks, protocols and applications will not have this luxury and hence a systematic methodology is needed to design, develop, test and evaluate such systems.

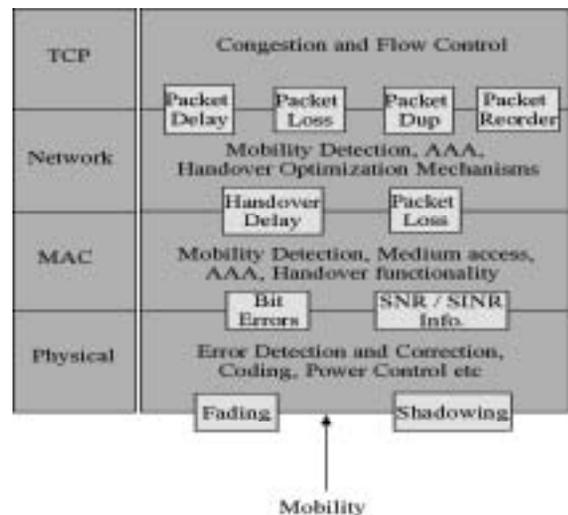


Figure 1: The Effect of Environment on Building Blocks

The design of each layer affects the overall performance of the protocol stack. Fig.1 shows the effects of each layer on the higher layers. For example, fading at the physical layer will manifest itself as higher bit error rate, which in turn will show

up as packet loss at the MAC layer. However, introducing additional mechanisms like channel coding or ARQ at the MAC layer to counter these effects, may lead to decreased effective bandwidth or increased packet delay. We need a systematic way in which we can capture the inter-layer effects so that we can evaluate their effects on the performance of higher layers. Such a methodology would aid in refining existing designs to get the required performance. This example of building block based design presents an insight into the methodology we propose to use.

Our work was partially inspired by VLSI CAD tools [16]. In VLSI design the system is modeled at different levels of abstractions and the model at each level is refined using finer and more detailed models (Behavioral → Structural → Physical). We wish to use similar techniques for the design of network protocols. However, in VLSI any Boolean function can be represented by a universal representation like NAND or NOR gates. There exists nothing similar in the field of networking protocols. The hierarchical techniques work extremely well in VLSI CAD as the characteristics of the universal representations are very well understood and modeled. Due to the small set of the universal building blocks, specifying them and testing for correctness is well understood as compared to protocols. We aim to study the feasibility of a similar hierarchical technique based on successive refinement for systematic protocol design and analysis.

Significant work has been done in the field of protocol composition from components [1],[2],[3],[4],[5]. The Ensemble and the Horus projects [1] stand out as they are able to do both formal proofs of protocol stacks as well as code generation. They represent systems based on a library of micro-protocols which are rather coarse grained and whose properties have already been verified. The components are drawn from the library and the required protocol is built in a strict vertical fashion from the specification. The emphasis in such systems is on protocol correctness and code generation. Since the coarse grained library of building blocks acts as a set of black boxes, extending the protocols is not easy in such frameworks. BAST is another system that uses an object-oriented library of reliable distributed protocols. As in the previous case implementation and code generation are emphasized. The system in [3] is also based in Ensemble, however it focuses on optimization of the design within the Ensemble framework. This mainly deals with implementation optimizations rather than protocol design optimization. In [4] and [5] category theory is used to provide guidelines to build functional primitives or building blocks. They also address the issue of interaction between building blocks. Though this list of references is not exhaustive, most of them are concerned with correctness of protocols and also with implementation or code generation. Few have methodologies using which we can analyze protocol performance and almost none of them model performance based on the building block approach. Furthermore, they do not address the issue of systematically analyzing protocol performance in a given environment or generating scenarios that can be used to provide good insights into protocol performance and refinement.

3. THE HIERARCHICAL BUILDING BLOCK FRAMEWORK

Network protocols are designed to achieve certain functionality or objective. Protocols that achieve similar

objectives may be categorized into the same class. For example, ad hoc routing protocols aim to provide valid routing paths functionality suitable for the wireless mobile environments. Also, micro-mobility protocols are used to maintain the network connectivity of a mobile node while it moves between subnets within the same domain.

A common practice in the networking research community is to study each protocol as a whole entity through simulation or analysis. The evaluation of network protocols are done in a heuristic, rather ad hoc, fashion. Unlike traditional methods, in our proposed hierarchical building block approach, the protocol is decomposed into a set of parameterized mechanistic components called 'building blocks', each of which is in charge of a specific well-defined functionality used in the protocol. Then, these building blocks are glued together to interact with each other over 'channels' in a proper fashion. For the different protocol instances falling into the same category, the organization and exact parameter settings of building blocks are different in each protocol instance. The actions of the building blocks themselves and the interaction between them via channel determine the behavior of the protocol for a given environment.

First, we introduce the two basic elements of our building block approach: 1) building blocks and 2) channels. We then describe the dynamic behavior of the building blocks and their interaction, and provide the hierarchical structure of our building block approach as the basis for protocol analysis, design and synthesis.

3.1 Building Blocks

The building blocks, the bricks used to construct the network protocol, are a set of separated modular components that are common to a broad class of network protocols attempting to accomplish a similar goal. Thus, each building block is a constituent of a protocol that addresses one (or several) functionality, depending on its level of granularity. Conceptually, each building block is specified in terms of a number of variables to be stored and modified by the building block as well as a series of actions conducted over those variables. The variables are manipulated by the protocol, such as the routing tables, packets, timers, etc. These variables also determine the state of the building block, and the state of the protocol. The actions define how the individual building block behaves in the face of different conditions. Once the building block is called upon, it follows its specification and conducts the appropriate actions over the variables, based on the particular circumstance at that time.

The set of actions specified in building block only defines how the building block reacts to various input events in general trend. The specific behavior and performance of a building block, however, also depends on its parameter settings (e.g., timer values, range of flooding). Hence, in practice, the behavior of network protocols may differ considerably, even though they consist of the same set of building blocks, organized in a similar way. The parameterized building block approach allows us to capture this difference between protocols belonging to same category.

In order to distinguish and represent the building blocks with different functionalities and parameter settings, it is essential to capture the inherent characteristics and properties of building blocks. Based on the above discussion, formally, we describe the building block as tuple

$[V(\text{variables}), E(\text{vent}) \rightarrow E(\text{effect}), P(\text{arameters})]$

where

(1) The $V(\text{variables})$ describe the variables kept at each building block, used to model the state of the building block.

(2) The $E(\text{vent}) \rightarrow E(\text{effect})$ includes a set of rules regulating the transitions from the incoming event to the outgoing effect generated by the building block. In effect, it defines the functionality of the building block under various input conditions. The $E(\text{vent})$ describes the stimulus that invokes this building block, including procedure call, message passing, or timer event. After the event occurs, the building block performs the specific actions over the corresponding variables and possibly generates a resulting event represented by the $E(\text{effect})$.

(3) The $P(\text{arameter})$ defines the parameter settings used for the building block reflecting the implementation details, such as timer expiration values (or intervals) or caching policy. These parameters may be used to adjust the performance of the building block.

Thus, a building block is composed of three major components: (a) variables kept in the building block, (b) the actions taken by the building block and (c) the parameter setting of the building block.

To illustrate, in a mobile ad hoc routing protocol, for example, the function of maintaining the caching table is considered as a single building block. In addition to a number of cache entries (i.e., variables), the building block also includes the basic operations (or transition rules), including caching table initiation, insertion, deletion and lookup. Parameters of such building block include maximum number of caching entries and cache expiration timers, the setting of which is expected to affect the detailed behavior and performance. In the case study shown in section 6, we conduct further detailed investigation on this building block.

3.2 Channels

Each individual building block is responsible for one specific function that is only part of whole protocol mechanisms. Therefore, various building blocks with different functionalities are organized together in certain fashion to realize the protocol as a whole. Specifically, the building blocks are connected with each other via their interfaces on well-defined channels.

A channel is introduced to model the connection between building blocks. The building blocks may interact with each other within the same node, or between different nodes. Typically, interface calls between building blocks in a local node can be modeled by a channel that delivers the interface call reliably and instantaneously. However when the building

blocks are located in different nodes, such interface calls may be lost, duplicated, reordered, delayed, etc. The concept of channels enables us to model and represent the different type of connections between the building blocks in a uniform way.

A building block is linked to another building block via a channel if and only if there is an interface call between these two building blocks. In other words, if the function of one building block is called upon by another building block, or if some messages are passed, a channel must exist between these two building blocks.

It is also essential to capture the inherent properties of the channel between building blocks. We describe the channel as a tuple

$[I(\text{nport}) \rightarrow O(\text{utport}), C(\text{haracteristics}), M(\text{essages})]$

where

(1) The $I(\text{nport})$ designates the input interface of a building block on one end of a channel, the $O(\text{utport})$ designates the output interface of another building block on the other end of the channel.

(2) The $C(\text{haracteristics})$ describe the properties of the channel, including the characteristics of delay, or loss experienced by the packets.

(3) The $M(\text{essages})$ describe the type of the messages, if any, transferred over channel between the two building blocks.

3.3 Dynamic Behavior of Building Blocks and Channels

Once the building blocks and the channels are determined, the network protocol could be represented as a graph consisting of building blocks and channels, where the parameterized building blocks are the vertices and the channels connecting the building blocks are the edges. Conducting the operations of individual building blocks in an appropriate order, we are able to implement the protocol mechanisms. This reflects the static aspect of protocol mechanism.

The network protocol is deployed and operated under a variety of different conditions and environments, which generates a sequence of stimulating events causing the protocol to act. Those events are of various types, including link breakage caused by node mobility, service interruption caused by node failure, or service requests placed by the applications and users. Upon receiving the input stimuli, the building blocks react to the incoming events, conducting the proper operations and interacting with each other, in accordance with the transition rules regulated by the functionality of the building blocks and channels.

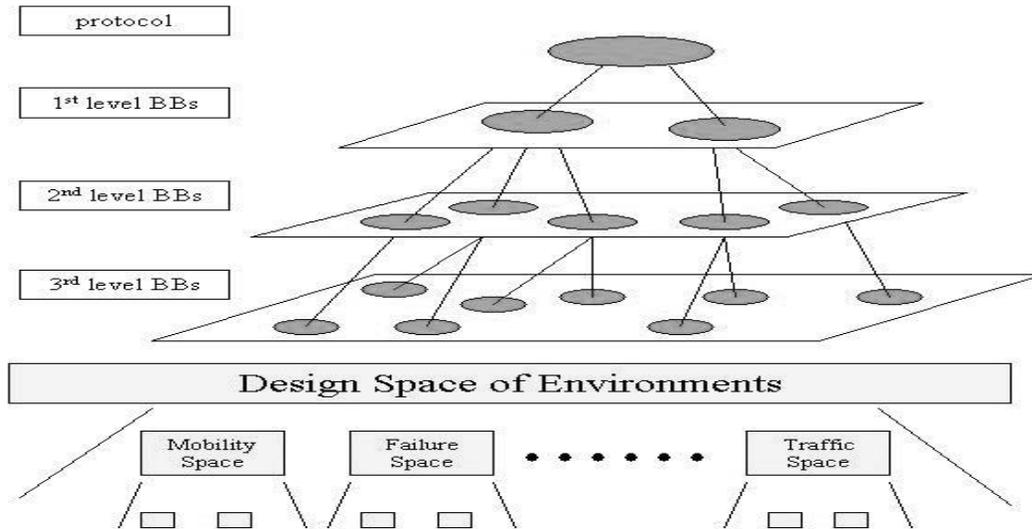


Figure 2: The Protocol Design, Analysis and Refinement Framework Through Building Block Approach

The tuples defined for the building blocks and channels in section 3.1 and section 3.2 only reflect their static structure, including their functionalities and their inherent characteristics. The dynamic behavior of the building blocks could be estimated if the sequence of input events is known, since the behavior of the building blocks is deterministic. To be exact, we could describe the dynamic behavior of building block as

$$\{[V(\text{variables}), E(\text{vent}) \rightarrow E(\text{ffect}), P(\text{arameters})], E(\text{vent})\}$$

where tuple $[V(\text{variables}), E(\text{vent}) \rightarrow E(\text{ffect}), P(\text{arameters})]$ identifies the functionality of the building block and $E(\text{vent})$ specifies the sequence of events injected into building block. Similarly, the interaction between building blocks could be estimated if the events occurring over the channel are known. We also describe the dynamic behavior of the interaction between building blocks as

$$\{[I(\text{nport}) \rightarrow O(\text{utport}), C(\text{haracteristics}), M(\text{essages})], E(\text{vent})\}$$

where tuple $[I(\text{nport}) \rightarrow O(\text{utport}), C(\text{haracteristics}), M(\text{essages})]$ identifies the key properties of interface call conducted over the channel and $E(\text{vent})$ describes the sequence of events in the channel.

The performance of building blocks with different parameter settings may vary under various environments. To analyze the performance of a building block, it could be modeled as a mechanistic ‘black box’ with certain parameter settings. The performance for building block could be formally described as

$$\text{Performance} = f(P_i, E)$$

where P_i are the values of parameter settings for building block i , E represents the underlying environments, and **Performance** is a certain performance metric of the building block. Function $f()$ reflects the mechanism of building block, which may or may not be written in closed form.

To illustrate, we take the example of the remote cache lookup building block in the Dynamic Source Routing (DSR) [17] protocol in MANET under mobility scenarios. In this building block, the cache, in effect the routing table, is looked up once an existing route breaks. One metric capturing the mobility environment is the frequency of link failure. One performance metric for the building block is the overall overhead to conduct this lookup. By adjusting the size of cache table and how the cache tables are updated, we achieve different performances under the same mobility scenarios.

An individual building block with specific parameter setting achieves certain performance under some environmental conditions. However, the building blocks interact with each other in a complex fashion. The overall protocol performance is the result of the individual building block performance in addition to the interaction between building blocks. Hence, careful examination of the interaction between the building blocks is needed to understand the overall performance. By appropriately addressing this issue, a micro-level analysis of the protocol mechanisms is conducted in a systematic fashion. This way, we are able to synthesize and analyze high-level building blocks from their smaller, simpler, lower-level building blocks.

As an example, the performance of a high-level building block consisted of three low-level building blocks can be described as follow

$$\text{Performance} = G(f_1(P_1, E), f_2(P_2, E), f_3(P_3, E), h_{12}(), h_{13}(), h_{23}())$$

Where $f_1(P_1, E)$, $f_2(P_2, E)$, $f_3(P_3, E)$ describe the performance model of the three low-level building blocks respectively, and $h_{12}()$, $h_{13}()$, $h_{23}()$ describe the interactions between those building blocks.

3.4 The Hierarchical Organization of Protocols in terms of Building Blocks

The whole network protocol is initially broken into a set of building blocks with different functionalities. These building blocks interact with each other via channels. Each building

block has its own specification and behavior. The overall behavior of a network protocol for a given environment is a combination of the behaviors of different building blocks and their interaction.

Sometimes, analyzing and modeling the first level building blocks (BBs) is not simple enough. One approach in this case would be to decompose these building blocks further. That is, the functionality of a high-level building block could be further decomposed into a number of low-level building blocks, each implements part of the functionality of the high-level building block. The division of building blocks is done successively, until the level where the resulting low-level building blocks are simple enough to be well-defined, parameterized and their interaction simply modeled. Since the decomposition of protocol is done in a hierarchical manner, we call it the Hierarchical Building Block framework. Fig.2 illustrates a hierarchical building block approach for a specific network protocol.

In decomposition, the set of low-level building blocks with their interaction should be equivalent to the original high-level building block. In other words, several rules should be satisfied during the decomposition process, including

- (1) The set of low-level building blocks in concert accomplish the same functionality of the high-level building block;
- (2) The structure of low-level building blocks together with their interactions agree with the structure of high level building block;
- (3) The interfaces of the set of low-level building blocks are compatible with the set of interfaces of its high-level counterpart;
- (4) The set of low-level building blocks achieves the same behavior of high-level building block under various network scenarios;

As long as above conditions are satisfied, the decomposition of high-level building block into set of low-level building blocks could be done in different ways, depending on the designer's preference.

4. THE MODELING OF ENVIRONMENT

Network protocols are deployed in various kinds of environments where complex and unexpected events may occur. For example, intra-domain routing protocols are deployed over a variety of different topologies. Also mobile ad hoc networks may be used in different scenarios where the node mobility patterns and communication traffic patterns may vary widely. Furthermore, wireless sensor networks, that collect and monitor physical phenomena, may be used in a mixture of applications ranging from habitual environment monitoring to object tracking. The protocol performance will depend heavily on the deployment environment. Hence, understanding the micro-level protocol behavior and performance across a wide array of operating conditions is essential for the design of robust, efficient protocols.

It is essential to evaluate and analyze the performance of designed protocol in a variety of environments before the deployment, in a systematic way and to be able to gain a deeper understanding into how the protocols, and its

composite building blocks, behave under different test cases. Furthermore, through examining the effect of building block parameters, those parameters could be adjusted to achieve the desirable performance under a given scenario. This is particularly important for the cases where the requirement of the network protocol is application-specific, as the case in classes of sensor network.

Modeling the underlying environment in a systematic and faithful way plays an important role in the evaluating, analyzing and refining the network protocols. The environment is thought of as an n-dimensional evaluation space, with each dimension representing a particular factor of the environment. Each factor represents a certain class of events with common properties. For example, the underlying environments to test mobile ad hoc and sensor networks potentially include several factors, such as node mobility pattern, communication traffic pattern, node failure pattern and power consumption pattern, etc. Moreover, each factor of the environment is also an m-dimensional subspace, consisting of several small elements with different characteristics. For instance, the mobility space includes several dimensions like relative velocity between nodes, spatial dependence of velocity between nodes, temporal dependence of velocity between time, etc. [8]. The communication traffic space includes the dimensions such as duration of communication traffic, location of communication traffic and type of communication traffic etc. Fig. 2 illustrates an example for the evaluation space of environment spanning several dimensions.

To thoroughly study the effect of the environment on the protocol performance, we propose to evaluate the protocol over a rich set of models that span the design space of the environment. To do so, the first step is to determine the dimensions of evaluation space and its composite subspaces. Once these are determined, metrics that quantitatively measure their key characteristics should be defined. By taking the characteristics of each environment space dimension into consideration, a set of parameterized environment models could be obtained, resulting in a good coverage of the proposed environment metric space by producing a rich set of environmental models. This set of environmental models is used as an underlying "test-suite" to evaluate and analyze the protocol and its mechanistic building blocks in future research.

5. DESIGN, PERFORMANCE ANALYSIS AND REFINEMENT OF PROTOCOLS

5.1 Design

Protocol design usually starts with a high level functional description which is later refined into additional functional requirements based on the correctness and performance requirement in a given environment. This type of monolithic design is extremely complex when the protocol requires many functional components that interact in a distributed fashion. So we advocate a modular and hierarchical design approach in which the functional requirement of the protocol is achieved by having coarse-grained building blocks that interact with each other to

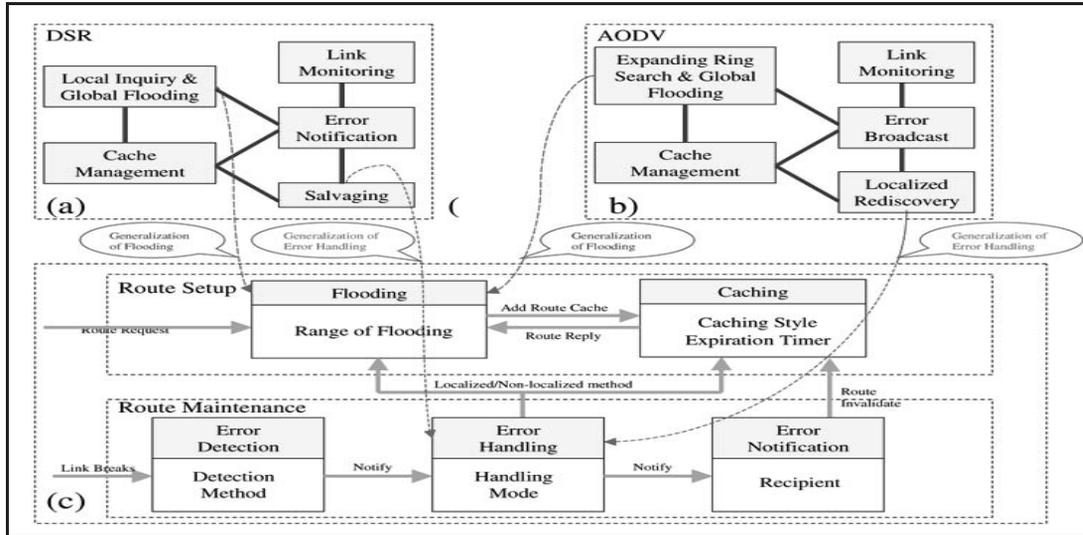


Figure 3: Diagram of Building Block Approach for MANET Reactive Protocols

produce the required functionality. Once the functional requirements of the building blocks and their interaction are known, the interfaces, states, variables and parameters can be defined. Based on the interaction between the building blocks and the environmental conditions in which they are expected to perform, they can be connected by appropriate channels. Depending on the channel characteristics, additional mechanisms may need to be added to each of the building blocks so as to meet the functional requirements of the building blocks under various channel characteristics. This process can be repeated continuously till we reach the required granularity.

An important thing to note here is that there may be many ways in which the protocol can be split into building blocks and each combination may have the same or different performance. Implementability of the functions of the building blocks, complexity of implementation and extensibility of the protocol are some of the things that need to be kept in mind while using the building blocks approach.

5.2 Performance Analysis and Refinement

The ability to analyze the performance of a protocol based on the building blocks approach is essential during the design of new protocols or when the existing protocols need to be studied or refined. While designing new protocols, there may be many ways in which the protocol can be divided into functional components. Performance is one of the criteria used to select one type of functional division over the other. Once we know the functional building blocks and their interaction, we can evaluate the performance under the given operating environment as described in Section 4.

Refinement of existing protocols or newly designed protocols essentially involves either tuning the parameters of the building blocks or adding / deleting building blocks from the original design. With the operating environment represented as n-dimension evaluation space, we need to translate the parameters of the environment into interface calls of the building blocks that directly take inputs from the environment. For example, fading, a physical layer effect

caused by environmental changes, translates to some distribution of BER, which is the input to the physical layer building blocks. Once we translate these environmental changes to interface calls with the required properties (temporal, probabilistic, stochastic etc), they can be used to understand and analyze the effects of the environment on protocol performance based on the performance metrics of the building blocks and channels that link them together.

Performance tuning involves optimal or near optimal setting of parameters of building blocks so that the best possible performance is obtained in the given set of environmental conditions. The building blocks approach allows us to understand how the protocol building blocks performance affects the overall performance and hence performance tuning can be done in a systematic manner. When entire building blocks or a set of building blocks are replaced as in the case of protocol re-design or refinement, it is much easier to understand the effect of the new building blocks and their interaction on the performance of the overall protocol.

6. PERFORMANCE ANALYSIS OF BUILDING BLOCKS FOR MANET REACTIVE ROUTING PROTOCOLS

A mobile ad hoc network is a collection of mobile nodes forming a network without any existing infrastructure. Previous studies (e.g. [8]) observe that the mobility factor plays a significant role in affecting the MANET routing protocols. Therefore, one of the main challenges in mobile ad hoc networks research is understanding the effect of mobility on the performance of routing protocols. In this case study, we carry out a preliminary building block based analysis for the impact of mobility on two reactive routing protocols, DSR [17] and AODV [18], after identifying the basic building blocks of MANET reactive routing protocols and their parameter setting. Thus we can extract the relative merits of different parameter settings and achieve a better understanding of various building blocks of MANET routing protocols, which will serve as a solid cornerstone for the development of more efficient MANET routing protocols.

The part(a) and part(b) of Fig.3 show the building block architecture for DSR and AODV respectively, the part(c) of Fig.3 shows a generalized building block architecture for reactive MANET protocols.

6.1 Building Blocks for DSR and AODV

First we discuss the functionality, organization and design choices (parameter settings) of the identified building blocks of reactive MANET routing protocols and specific parameter settings for DSR and AODV¹. We pose some questions about the utility of the various design choices made by these protocols. In section 6.2, we attempt to answer these questions.

Reactive ad hoc routing protocols such as DSR and AODV are composed of two major phases: *Route discovery* and *Route maintenance*.

Route Discovery is initiated if there is no cached route available to the destination. This mechanism consists of the following building blocks:

Flooding building block: The flooding building block takes responsibility to distribute the route request messages within the network. Here, the key parameter is *the range of flooding*, generally described by TTL field in the IP header. For the *range of flooding*, DSR conducts a non-propagating direct-neighborhood inquiry (TTL=1) before the global flooding (TTL=D, D is network diameter). Similarly, AODV uses the expanding ring search (TTL=1,3,5,7) before the global flooding is initiated. Here, we want to answer the following question: ***How useful are non-propagating route requests?***

Caching building block: The caching building block helps to efficiently and promptly provide the route to the destination without referring to the destination every time. One key parameter of this block is *whether aggressive caching is allowed*, i.e. whether multiple cache entries are allowed for the same destination and whether a node can cache the route information it overhears? As we know, DSR uses *aggressive caching*, while AODV does not. For caching, we are interested in the following questions: ***How useful is caching? and Is aggressive caching better than non-aggressive caching?***

The *Route Maintenance* phase takes the responsibility of detecting broken links and repairing the corresponding routes. This phase is made up of the following building blocks:

Error Detection building block: It is used to monitor the status of the link of a node with its immediate neighbors. Here, the parameter is *the mode of error detection used*. Since both DSR and AODV can use similar choice, we do not investigate this building block in our analysis.

Error Handling building block: It finds alternative routes to replace an invalid route after a broken link is detected. One of the parameters to this block is *what recovery scheme should be used*. In DSR, on detecting a broken link, the upstream node will first search its cache to replace the invalid route (this scheme is called salvaging), although the found alternative route may also be invalid in some scenarios. While in AODV, the upstream node detecting the broken link will initiate a localized flooding to find the route to the destination. For this

building block, we are interested in the following question: ***Which is a better scheme for localized error handling: cache lookup or localized flooding?***

Error Notification building block: It is used to notify the nodes in the network about invalid routes. The key parameter to this building block is *the recipient of the error message*. Either only the source is notified or the entire network is notified. Since both DSR and AODV only notify the error to the source, so we do not investigate this building block in our analysis.

Besides these three questions about the design choices, we are also interested at the explanation for the observation we made in [8]: DSR outperforms AODV in most mobility scenarios except the Freeway and Manhattan model with high mobility.

6.2 Experiments to Evaluate and Analyze the Building Blocks

We identified parts of the network simulator (ns-2) code [13] which implement these building blocks and profiled them during our simulations. Following the methodology of modeling the environment introduced in Section III, the mobility scenarios are generated to include a set of random waypoint, RPGM, Freeway and Manhattan models [8] whose maximum velocity vary from 5m/s to 60m/s, which is believed to span the whole evaluation space for mobility factors. The performance of building blocks under those mobility scenarios is discussed as follow and several questions asked above are answered.

Flooding: We measure the likelihood of finding a route to the destination from the source's neighborhood. Through simulations, we find that non-propagating route request is frequently used (more than 30% for DSR and more than 10% for AODV in most scenarios). However, the ratio for DSR is almost twice as large as that for AODV across all mobility models. A possible reason for this comes from the fact that DSR uses aggressive caching as compared to AODV. When such a caching scheme is coupled with the mechanism of non-propagating route requests, it translates to low routing overhead and high throughput as was shown in our study and several other comparative studies. Thus, it seems that caching has a significant impact on the performance of DSR and AODV. Hence we study it next.

Caching: To measure the effectiveness of caching, we evaluate the ratio of the number of route replies coming from the cache to the total number of route replies. Fig.4(a) and Fig.4(b) show that this ratio is high for Random Waypoint, Manhattan and Freeway models, which implies that most of the route replies for these mobility models come from the cache (more than 80% in most mobility scenarios).

The ratio difference for DSR and AODV is greater than 20% for all mobility models. DSR uses aggressive caching as compared

¹ The process of protocol decomposition for both protocols, which follows the methodology introduced in this paper, is omitted because of the limited space. More details are included in Ref.[6].

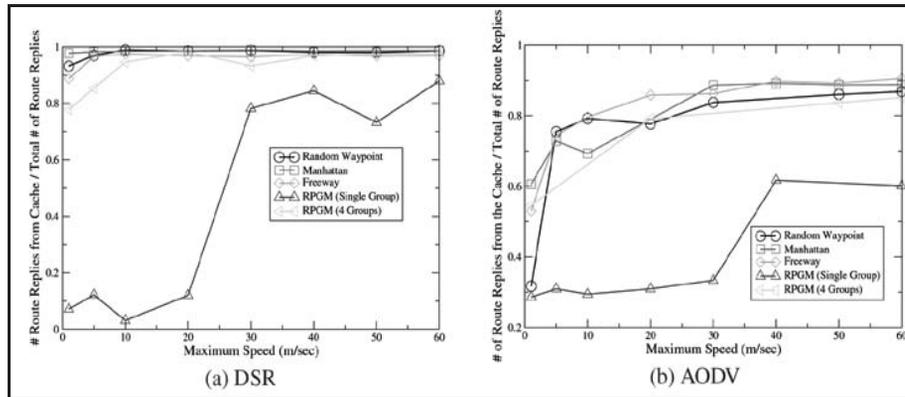


Figure 4: Ratio of Route Reply from the Cache

to AODV. Thus, the likelihood of a route reply coming from a cache is higher in DSR than in AODV. Therefore, fewer route requests will be needed and thus the routing overhead of DSR is lower than AODV as we observed in [8]. Thus, aggressive caching seems to be a good design choice.

To completely evaluate the caching strategy, we also need to examine the validity of the cache entries. We evaluate the ratio of invalid cache entries to the total number of cache entries for DSR. In experiments, we find the invalid cache ratio increases from RPGM (around 10%) to Random Waypoint to Freeway (around 60%) to Manhattan (around 80%) mobility models. It means that caching may have adverse effects in mobility models with a high relative speed and it may lead to cache invalidation. Packets may be sent on invalid routes, which might lead to packets being dropped and route request retries. This leads to a lower throughput and higher overhead for DSR for the Freeway and Manhattan models as was shown in our study.

On the other hand, in mobility models with very high relative speed like Manhattan and Freeway, AODV seems to achieve as good a throughput as DSR (and sometimes better). AODV does not use aggressive caching, thus the ratio of the number of route replies coming from the cache to the total number of route replies is lesser for AODV than DSR. Thus, the likelihood of getting invalid routes from the cache is lesser for AODV than for DSR. This may explain why AODV outperforms DSR in Freeway and Manhattan models with high mobility.

Moreover, at high relative speeds, the number of routes broken is greater. Thus, a protocol that has a better error-handling mechanism at higher relative speeds might perform better in such situations. This line of reasoning leads us to evaluate the next building block of interest - Error Handling.

Error Handling: To study the effectiveness of error handling, we focus on localized error handling. We evaluate the ratio of the number of localized error handling to the total number of route errors for both DSR and AODV. For DSR, we notice that salvaging accounts for less than 2% of the total number of route errors. Moreover, if we take invalid cache entries into account, the effect of salvaging on the protocol performance is further lowered. On the other hand, in AODV, a route request is initiated by the upstream node which detects the broken link if it is closer to the destination. In AODV, the frequency of initiating localized flooding is between 40% and 50% for Freeway and Manhattan models. Moreover the routes obtained

by this mechanism are more up to date than those from the cache salvaging in DSR. This is another factor which explains the better performance of AODV as compared to DSR in the Freeway and Manhattan models.

6.3 Discussion for Refinement of Building Blocks

The above study of the building blocks has given us greater insight into the design of the reactive routing protocols for MANETs. Decomposing a protocol into building blocks and evaluating these building blocks have shown us scenarios in which the chosen parameters can give a better performance. From the above study, we learnt the following principles of protocol design:

1. *Caching helps reduce the protocol overhead.* However, whether aggressive caching should be used depends on the scenarios in which the protocol will be deployed. For low mobility scenarios, aggressive caching might be useful, while for higher mobility scenarios, the more stale cache entries incurred by aggressive caching might affect the protocol throughput adversely.
2. *Non Propagating route requests, when combined with caching also reduce the protocol overhead.* If caching is widely done in the network, it may be more advantageous to do non propagating route requests (or expanding ring search) than globally flooding the route request. In DSR, due to aggressive caching, it may be more useful to do expanding ring search (from the source) on a route error than doing a global flooding (from the source).
3. *The nature of localized error handling also has a significant impact on protocol performance.* Re-initiating a route request from an intermediate node can be more advantageous than doing a local cache lookup in high mobility scenarios, while a cache lookup might be more advantageous for low mobility scenarios.

Thus, no particular parameter setting of these building blocks is the most optimal for all scenarios. This further strengthens our conclusion that there is no clear winner among the protocols across all mobility scenarios. A promising direction in this area may be to design mobility-adaptive protocols.

7. BUILDING BLOCK ANALYSIS FOR MICRO MOBILITY PROTOCOLS

Mobile IP supports mobility of the IP hosts. However, frequent handoff leads to frequent registration with the home agent, leading to increased packet loss and delay. Micro-mobility protocols reduce this delay and loss by hiding mobility of the host from the Home agent as long as the mobile node (MN) is with the same domain. Extensive research in the field of micro mobility has led to the development of a large number of protocols like HAWAII [12], CIP [10] and M&M [11]. Most of these protocols use a combination of customized mechanisms for routing and handoff. Micro mobility protocols need to work in a wide variety of scenarios, such as varied underlying infrastructure support, mobility patterns, MAC and physical layer. To explore the design and evaluation space, we partition the functionality of micro mobility protocols the following common mechanistic building blocks: (1) addressing, (2) routing and packet forwarding, (3) association and de-association detection (mobility detection), (4) buffering, (5) handoff optimization and signaling, (6) paging and (7) authentication, authorization and accounting (AAA). In addition, we recognize the need for additional mechanisms: (1) address mapping, and address map distribution and (2) distribution tree root selection and announcement. Different versions of different micro mobility protocols have different instances (appropriate subset) of the building blocks. Fig. 5 depicts the building blocks (except AAA) and the relationship between them. The dotted lines indicate the information required by different building blocks whereas the solid lines indicate one building block utilizing or triggering mechanisms of the other building blocks.

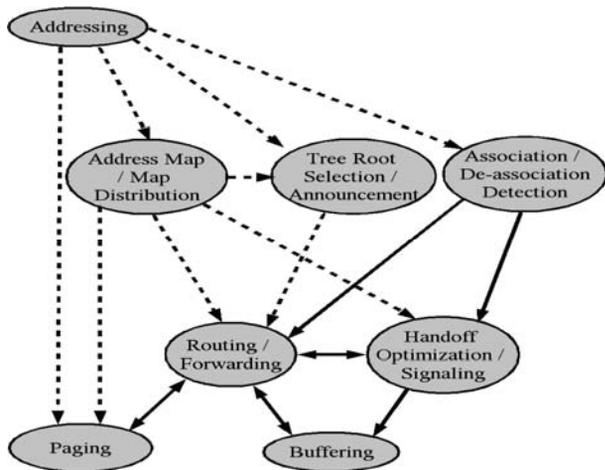


Figure 5: Building Blocks for Micro Mobility Protocol

To get a better understanding of the building blocks in different micro mobility protocols, the next part of this section describes where each building block is used in different micro mobility protocols and how packets arriving at the BR are delivered to the MN (in a foreign domain).

When an MN moves from one domain to another, it incurs MIP handoff. The MN acquires a unicast address, which it retains as long as it remains within that domain. In M&M, the unicast address is also used to generate a unique multicast address using an algorithmic mapping. In contrast, CIP and HAWAII do not use any kind of mapping mechanism. In these

protocols, when a border router(BR) of the foreign domain within which the MN resides receives packets destined to the MN, it either looks for a forwarding entry in its routing table or a tunnel to the next agent in the hierarchy. If neither is found, it can optionally buffer packets and/or page the MN. For the BR to recognize that the packet is destined to an MN (so that BR initiates paging for packet destined only to MN), there must be a mechanism by which BR can recognize the association. Therefore mechanisms that map and announce the association of the MN's address are required. The MN (or its serving access router or base station(BS)) responds to paging and initiates route setup. To initiate the creation of the delivery tree, the initiator must know where to send the route update messages (usually towards the root of the delivery tree). Thus there must be a mechanism by which the root of the delivery tree can be selected (statically or dynamically) and announced².

7.1 Analysis using Building Block Approach

Packet delivery performance of a micro-mobility protocol is a strong function of the type of handoff optimization mechanism being used. Typically, handoff delay and jitter are a function of association/de-association detection (mobility detection), AAA, route setup/repair and handoff optimization delays.

$$T_{handoff} = f(T_{mobilityDetection}, T_{AAA}, T_{routeRepair}, T_{handoffOpt}, T_{gap})$$

where $T_{mobilityDetection}$ is the time it takes for the MN to detect that it has entered into the coverage of a new BS (association), or for the old BS to realize that the MN has moved out of its coverage (de-association), T_{AAA} is the time taken to complete AAA functions at the micro mobility level, $T_{routeRepair}$ is the time it takes for the routing entries to be installed on the route to the MN after it has moved, $T_{handoffOpt}$ is the time required to setup buffering and forwarding functionality (not necessarily in that order) and T_{gap} is the time for which the MN is not in the radio coverage of any BSs.

Association and de-association detection building block is responsible for triggering route repair and handoff optimization mechanisms. As the granularity of the $T_{mobilityDetection}$ becomes coarse, handoff jitter tends to increase. When this approaches the order of magnitude of link delays, $T_{handoff}$ increases. However, scenarios in which the MN can simultaneously communicate with more than one BS, the granularity of $T_{mobilityDetection}$ is not an issue as long as there is sufficient overlap in the radio coverage.

The time taken for route repair is a function of the delay of the path on which the update messages traverse. $T_{routeRepair}$ in bi-cast and CAR-set handoff optimization schemes is of the order of link delays from the new BS to the fork router. In buffer and forward schemes like HAWAII MSF, it is twice as much since route update message travels from the new BS to the old BS (typically this is twice the magnitude of the delays from BS to fork router). In buffer and forward schemes like MSF, the time required to forward packets is of the order of link delays from the old BS to the new BS, whereas in forward and buffer

² For the detailed discussion about the functionality of building blocks in the micro mobility protocols, please check Ref.[7].

schemes like triggered CAR-set, the forwarding time is of the order of the wireless link delay.

7.2 Evaluation Scenarios

With an understanding of the effect of the building blocks on performance metrics, we can generate parameterized scenarios to stress the building blocks. Following parameters can be used to generate a rich set of evaluation scenarios: Radio technologies (reactive and non-reactive handoff), Uniformity of radio coverage (varying gaps in radio coverage), Link delays (wired and wireless), Topology (tree of varying depths, non-tree), MN mobility patterns and Granularity of association and de-association detection. To target the handoff related building blocks, we generated scenarios with varied radio technologies (MN having the ability to simultaneously communicate with two or more BSs, MN with the ability to communicate with only one BS at a time), radio coverage (different overlaps, gaps in radio coverage), different link delays and tree depths. In scenarios where there were no gaps in wireless coverage, and MN was able to simultaneously communicate with more than one BS, bi-casting yields negligible loss and zero handoff delay (for both CIP and M&M), but at the cost of increased packet duplication. In this scenario, the MN continues to receive packets from the old BS while the mobility detection and route repair occurs (as long as it is in the coverage of both the BS, $T_{overlap}$). As long as the following condition is satisfied, bi-cast handoff optimization mechanism does not incur packet loss.

$$T_{overlap} > T_{mobilityDetection} + T_{AAA} + 2 * T_{routeRepair}$$

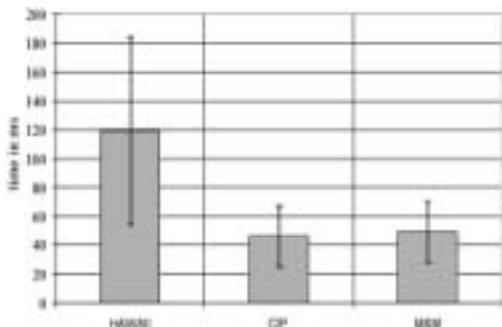


Figure 6: Handoff Delay and Jitter

However, bi-cast handoff scheme incurs high packet loss in scenarios in which the MN cannot simultaneously communicate with more than one BS (reactive handoff scenarios). Fig. 6 shows the handoff delay and jitter performance of CIP and M&M with bi-cast in reactive scenarios. Here, $T_{overlap}$ is effectively zero and the handoff delay for bi-cast given by the following formula

$$T_{handoff} = T_{mobilityDetection} + T_{AAA} + 2 * T_{routeRepair}.$$

Since packets are not buffered, all packets during handoff are lost when bi-cast handoff optimization is used in reactive handoff scenarios. Though HAWAII incurs handoff delay, it does not suffer any packet loss as it buffers packets. For the buffer and forward scheme like MSF handoff delay is given by

$$T_{handoff} = T_{mobilityDetection} + T_{AAA} + T_{handoffOpt}.$$

In MSF, the $T_{handoffOpt}$ is effectively the RRT between the new BS and the old BS. Since this is typically twice that of $T_{routeRepair}$, MSF suffers from higher handoff delay and jitter. Fig. 7 shows the packet loss performance of HAWAII with MSF, M&M with pro-active CAR-set and CIP with bi-cast, handoff optimization mechanisms in reactive handoff scenarios. In the pro-active CAR-set scheme, packets are simultaneously transmitted to all the BS adjacent to the BS to which the MN is associated with. Therefore, the handoff delay is given by

$$T_{handoff} = T_{mobilityDetect} + T_{AAA}.$$

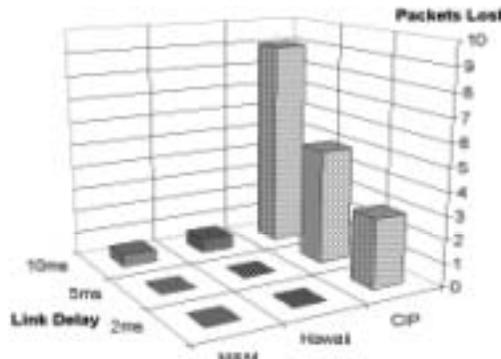
$T_{routeRepair}$ is zero since the BS to which the MN hands-off will already be receiving packets. This scheme does not use buffering. Here, the CAR-set handoff optimization mechanism trades off extra bandwidth to reduce packet loss, handoff delay and reordering. Non buffering schemes like bi-cast and pro-active CAR-set do not perform very well in scenarios in which there are gaps in radio coverage. Mechanisms using buffering perform better in scenarios where there are gaps in radio coverage. In this scenario, M&M uses triggered CAR-set handoff mechanism. Here, the old BS senses that the MN is out of range and triggers packet delivery to the BSs in the CAR-set. Packets are buffered at each BS and forwarded to the MN when the MN moves into its coverage. Packets are lost from the point at which the old BS realizes that the MN is out of range until the BSs in the CAR-set start receiving packets (after initiating route repair). This is typically the time it takes to perform signaling between the old BS and the CAR-set BS and the time it takes to perform route repair from the new BS. Handoff duration for triggered CAR-set is given by

$$T_{handoff} = T_{gap} + T_{mobilityDetect} + T_{AAA}.$$

However, for buffer and forward schemes like MSF, the time take to handoff is given by

$$T_{handoff} = T_{gap} + T_{mobilityDetect} + T_{AAA} + T_{handoffOpt}$$

Therefore, the MSF scheme incurs slightly higher handoff delay along with packet reordering. The triggered CAR-set handoff optimization mechanism trades off a little packet loss to reduce bandwidth utilization, handoff delay and packet reordering. Fig. 8 illustrates the packet loss performance of M&M (with triggered CAR-set), HAWAII (with MSF) and CIP



(with bi-cast).

Figure 7: Packet Loss during Reactive Handoff

To target the routing building block, we evaluated the protocols on different topologies (tree and non-tree with varying link delay and tree depth). Routing in both CIP and M&M establishes the shortest path from the root of the delivery tree to the MN. This is because both protocols send route repair messages towards the root of the delivery tree. HAWAII (MSF) establishes the shortest routes only in tree topologies. In non-tree topologies, HAWAII establishes sub-optimal routes due to the tight coupling between the handoff optimization and the routing building blocks. In HAWAII MSF, after association detection, the MN sends a route update message from the new BS towards the old BS. As long as the fork router is in the path between the old and new BS, this scheme establishes shortest routes. However, if fork router does not lie in the shortest path from the old BS to the new BS, forwarding paths are established from the old BS. This not only leads to sub-optimal routes, but also to increased bandwidth utilization and increased mobile specific states in the network.

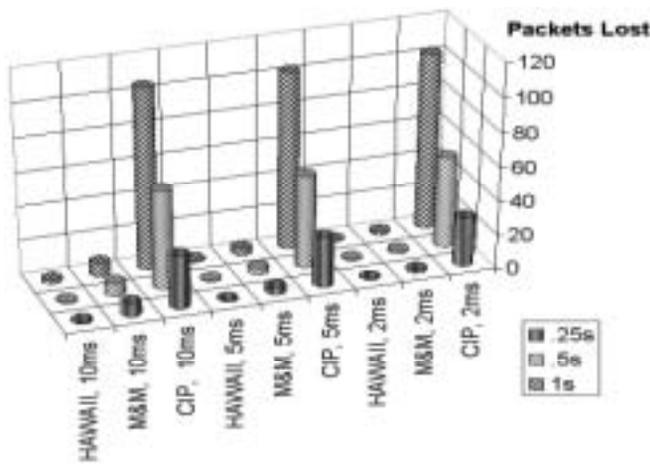


Figure 8: Packet Loss in Scenarios with Non-uniform Radio Coverage

7.3 Observations

Depending on the performance requirement and the scenarios in which we expect the protocol to operate, a single handoff optimization mechanism may not be sufficient. A protocol that can adapt or select an appropriate handoff optimization mechanism to the scenario at hand will invariably perform better than an instance of the protocol that cannot adapt.

Using the building block approach we were able to clearly identify and isolate the factors that influence the protocol performance. Furthermore, the approach also enabled us to understand the effects of different building blocks in different scenarios. In our experience, using the building block approach facilitates the systematic study (by generating scenarios targeting specific building block) of the effects of various handoff mechanisms (bi-casting, buffering) on packet delivery performance (packet loss, handoff delay, packet duplication) and route setup on route optimality and scaling behavior. This gives us an important insight into the design of micro-mobility protocols, enabling us to target specific building blocks to achieve the required performance in various scenarios.

8. DISCUSSIONS

This work represents the first step in our effort to evaluate, analyze, model and design network protocols in a systematic way through the building block approach. The fundamental idea and generic framework of our building block approach were described and several key concepts were introduced in this challenges paper. However, we should acknowledge that a number of open questions in this framework remain unsolved until now and bear further research.

One open question is the formal methodology to break down the protocol into building blocks by which the decomposition of protocol into building blocks and interactions could be automated. Our current solution is still a heuristic method where the procedure of decomposition, organization, generalization and parameterization of building blocks are conducted manually based on the designer's experience [6]. It is a well-known fact that the bad modular design, caused by a human-driven error-prone process, could result in unnecessarily complex and inefficient systems of functionality modules. For example, improper abstraction of mechanistic building blocks based on functionality may give rise to the complicated pattern of interactions between building blocks, which defeats to our original objective of reducing the complexity of protocol evaluation and analysis through the building block approach. The break down of the protocol into a set of building blocks in a meaningful way may be achieved in several ways. Currently we are investigating a minimum-interaction decomposition scheme resulting in functionality-independent building blocks based on graph theory.

The protocols in the same category normally consist of similar sets of building blocks with particular functionality. The functionality of the building blocks is similar while their parameter setting and implementation details across protocols may vary. To utilize this commonality, one approach is to establish a library of building blocks for a given class of protocols attempting to achieve the same objective. The protocol design process then is to pick the proper set of building blocks and adjust the parameter settings based on the deployed environments. For example, IETF reliable multicast transmission (rmt) charter suggests a set of building blocks including data reliability building block, congestion control building block, security building block, group management building block, session management building block [14,15] should be used to compose a protocol for the purpose of reliable multicast transmission. Thus, one possible direction in this area is to establish libraries of building blocks for the various classes of networking protocols. An effort to standardize the building blocks for the different purposes may ultimately facilitate the design and analysis of networking protocols, with the caveat that no one set of building blocks is sufficient to conduct all kinds of protocol evaluation studies. In addition, each library of building blocks should include a set of test-suites that are rich enough to span the environmental dimensions of the target application.

9. CONCLUSIONS

The emergence of progressively more complex protocols, such as classes of application-specific wireless mobile ad hoc and sensor networks, demands a systematic methodology and tools to evaluate, analyze and model the protocol performance under various environments. In this work, we propose a

hierarchical building block approach to decompose networking protocols into sets of building blocks. Each building block performs a particular functionality and connects with other building blocks via channels. The overall behavior of a network protocol under a given environment is determined by the building blocks as well as the interaction between them. As a consequence, the complex problem of evaluating and analyzing the protocol performance may be reduced to a set of sub-problem of evaluating and modeling simple building blocks. By studying the impact of parameter settings of each building block and the interaction between building blocks on protocol performance, greater insight is developed into the design choice of building blocks for different environments. This insight could be used to fine-tune the parameters of protocol or refine the protocol design to improve its performance for the target applications. To illustrate the utility of this building block framework, two case studies were presented on ad hoc reactive routing protocols and micro-mobility protocols. The studies describe how the protocols are decomposed into a set of building blocks connected via well-defined interfaces on the channels. Through systematic evaluation of the protocols, several lessons about the protocols' design as well as the underlying test suites are discussed. In this work, we attempt to point out a potential direction to analyze and understand the behavior of protocols based on the building block approach. We hope that this research direction lead to tools that aid the 'art' of protocol design using the 'science' of protocol design.

10. ACKNOWLEDGMENTS

We would like to thank Dr. Sandeep Gupta, Narayanan Sadagopan and Karim Seada for their helpful discussion during the development of this work; and the reviewer and editors for their comments. This work was made possible by a grant from NSF Career Award 0134650.

11. REFERENCES

- [1] Birman Kenneth, Robert L. Constable, Mark Hayden, Jason Hickey, Christoph Kreitz, Robbert van Renesse, Ohad Rodeh, and Werner Vogels. The Horus and Ensemble Projects: Accomplishments and Limitations, In *Proceedings of DARPA Information Survivability Conference and Exhibition (DISCEX '00)*.
- [2] B. Garbinato and R. Guerraoui, Flexible Protocol Composition in Bast, In *Proceedings of the 18th International Conference on Distributed Computing Systems (ICDCS-18)*, IEEE Computer Society Press, Amsterdam, The Netherlands, 1998.
- [3] Xiaoming Liu, Christoph Kreitz, Robbert van Renesse, Jason Hickey, et al. Building Reliable, High-Performance Communication Systems From Components, In *Proceeding of Symposium on Operating Systems Principles*, 1999.
- [4] Purnendu Sinha, Neeraj Suri: On Simplifying Modular Specification and Verification of Distributed Protocols. In *Proceeding of HASE 2001*: pp. 173-181.
- [5] P. Sinha, N. Suri. Modular Composition of Redundancy Management Protocols in Distributed Systems: An Outlook on Simplifying Protocol Level Formal Specification and Validation, In *Proceeding of ICDCS-21*, Phoenix, pp. 255-263, April 2001.
- [6] F. Bai, N. Sadagopan, A. Helmy, BRICS: A Building-block approach for analyzing Routing protocols in Ad Hoc Networks - A Case Study of Reactive Routing Protocols, in *Proceeding of IEEE International Conference on Communications (ICC)*, June 2004.
- [7] G. Bhaskara, A. Helmy, S. Gupta, Micro Mobility Protocol Design and Evaluation: A Parameterized Building Block Approach, in *Proceeding of IEEE Vehicular Technology Conference (VTC)*, October 2003.
- [8] F. Bai, N. Sadagopan, A. Helmy, IMPORTANT: A framework to systematically analyze the Impact of Mobility on Performance of Routing protocols for Adhoc Networks, in *Proceeding of IEEE INFOCOM (The 22nd Annual Joint Conference of the IEEE Computer and Communications Societies)*, pp. 825-835, March/April 2003, San Francisco.
- [9] A. Helmy, S. Gupta, D. Estrin, The STRESS Method for Boundary-point Performance Analysis of End-to-end Multicast Timer-Suppression Mechanisms, in *IEEE/ACM Transactions on Networking (ToN)*, February 2004.
- [10] A. T. Campbell and J. Gomez,, IP Micro - Mobility Protocols, in *ACM SIGMOBILE Mobile Computer and Communication Review (MC2R)*, 2001
- [11] A. Helmy, M. Jaseemuddin, G. Bhaskara, Efficient Micro - Mobility using Intra - domain Multicast-based Mechanisms (M&M), in *ACM SIGCOMM Computer Communications Review CCR*, October 2002.
- [12] Ramachandran Ramjee, Thomas La Porta, Sandy Thuel, Kannan Varadhan, and Shie-Yuan Wang, HAWAII: A Domain-based approach for Supporting Mobility in Wide-area Wireless Networks, *IEEE/ACM Transactions on Networking*, Vol 6., No. 2, June 2002.
- [13] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, H. Yu, Advances in Network Simulation, *IEEE Computer*, vol. 33, No. 5, p. 59-67, May 2000.
- [14] B. Whetten, L. Vicisano, R. Kermode, M. Handley, S. Floyd, M. Luby, Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer (RFC 3048).
- [15] R. Kermode, L. Vicisano, Author Guidelines for RMT Building Blocks and Protocol Instantiation documents (RFC 3269).
- [16] M. Abramovici, M. Breuer, and A. Friedman, *Digital System Testing and Testable Design*, AT&T Labs., 1990.
- [17] D.B.Johnson, D.A.Maltz and J. Broch. DSR:The Dynamic Source Routing Protocol for Multi-HopWireless Ad Hoc Network. In *Ad Hoc Networking*, edited by C.E.Perkins, Chapter 5, pp.139-172, Addison-Wesley, 2001.
- [18] C. Perkins. Ad hoc On Demand Distance Vector Routing(AODV), internet draft, draft-ietf-aodv-00.txt.

