ARMY RESEARCH LABORATORY

Flexible Client-server Architecture Designed for Testing Optimized Link State Routing (OLSRv2) for Component-based Routing (CBR)

by Justin James

ARL-TN-0375 **November 2009**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Adelphi, MD 20783-1197

ARL-TN-0375 **November 2009**

Flexible Client-server Architecture Designed for Testing Optimized Link State Routing (OLSRv2) for Component-based Routing (CBR)

Justin James
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
November 2009	Final	June to August 2009

4. TITLE AND SUBTITLE
Flexible Client-server Architecture Designed for Testing Optimized Link State Routing (OLSRv2) for Component-based Routing (CBR)

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Justin James

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: RDRL-CIN-T
2800 Powder Mill Road
Adelphi, MD 20783-1197

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TN-0375

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report discusses a client-server architecture designed for testing the Optimized Link State Routing (OLSRv2) protocol for Component-based Routing (CBR) implementation. The client-server application was devised to combine maximum performance flexibility with minimal structural ambiguity. The proposed client-server architecture is tremendously adaptable and permits the user to specify any number of parameters related to information exchange. This architecture is necessary because conventional studies have shown that the performance of a routing protocol in a Mobile Ad-hoc Network is stoutly reliant on the network environment and/or desired result(s). Consequently, to accomplish the most favorable routing performance in a dynamic network atmosphere, the routing procedure itself should be dynamic. One proposed resolution is CBR. In CBR, researchers use a compilation of fundamental component modules from existing routing protocols merged to construct a distinctive, best possible, on-demand protocol for any set of network circumstances. Consequently, this obliges the implementation of flexible client-server architecture. Since the network constantly changes, the client-server needs to constantly change. A flexible client-server architecture ensures maximum compatibility between client-server information exchanges. The client-server application produced will be used to test the performance of the component modules and overall routing system of OLSRv2. The results of the performance test will be used to implement CBR more effectively.

15. SUBJECT TERMS
Component-based routing, client, server, OLSR

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Rommie Hardy
Unclassified	Unclassified	Unclassified	UU	16	19b. TELEPHONE NUMBER (*Include area code*) (301) 394-1189

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

INTENTIONALLY LEFT BLANK.

1. Introduction

Traditional research has proven that the performance of routing protocols in a Mobile Ad-hoc Network (MANET) is strongly dependent on the desired outcome(s) and on the operating network environment. In fact, for any given routing protocol, the performance is heavily dependent upon the situation. Consequently, the exact same protocol that performs optimally in a high mobility scenario might operate below satisfactory standards in a low mobility situation or vice-versa. In a similar sense, all routing protocols are limited to operating within the protocol's tolerable conditions. Therefore, to attain optimal routing performance in a dynamic network environment, the routing protocol itself should be dynamic in order to adapt to perform optimally for the network metrics and requirements at any particular instance of time. However, for this approach to work as desired, details must be known about the performance of each component of each routing protocol under a variety of likely network conditions and scenarios.

2. Background

One solution to this new area of research has become known as Component-based Routing (CBR). However, CBR is not a newly design single standalone protocol; but instead, a collection of basic component modules from existing routing protocols fused together to produce a unique optimal on-demand protocol for any network or scenario. This technique is achievable because nearly all routing protocols perform the same core and/or auxiliary operations. Core functions of a routing protocol may include route discovery, route selection, route formation, data forwarding, route maintenance, and routing metrics. Auxiliary functions of a routing protocol may include neighbor discovery, neighbor maintenance, hierarchical structure, multicast, and security. Consequently, a mesh of modules of components from different routing protocols can be assembled to produce a new unique fully functional routing protocol. To illustrate this point of interchangeability, consider the core routing protocol function of data forwarding, which exists for all routing protocols. Some data forwarding modules will maximize reliability, while other modules will ensure energy efficiency. The goal of CBR is to consider the user requirements and network environment, and select the proper modules to compose the most suitable on-demand routing protocol. However, in order to know which modules should be used, an advanced understanding of the impacts, advantages, and disadvantages of each module are necessary.

Research of this nature contributes not only to the development of CBR, but also to the general sphere-space of MANET and routing knowledge. Such a detailed study of these routing protocols may yield phenomena not yet discovered or explored by previous researchers.

Through analyzing the performance of each module instead of the performance of overall the protocol, a more in-depth and information-rich set of data should be acquired, in comparison to research results obtained in similar studies using the more traditional approaches. The full benefits of CBR research are still undefined; however, since CBR employs such a novel research approach, researchers are confident that this process will produce some interesting and unique results.

Recently, CBR techniques were used to evaluate the performance of the Dynamic Source Routing (DSR) protocol. CBR simulation analysis of DSR was performed using an "Analytical Software Tool" designed by collaborating researchers at the University of Maryland, College Park. The simulation was considered a great success in analyzing the performance of the DSR component modules and overall routing system. Each core and auxiliary routing component was analyzed and results were acquired for each. However, emulation results have not yet been obtained to validate the simulation findings. In the near future, emulation results will be obtained using the Mobile Ad-hoc Network Emulator (MANE). If the emulation results match the simulation results, the simulation results will be considered valid, which would in turn validate the software analytical tool. Validating the software analytical tool provides several advantages. These benefits include providing a much less cumbersome testing platform than MANE for evaluating future routing component modules and systems.

Currently, research efforts are focused on evaluating the performance of the Optimized Link State Routing (OLSRv2) protocol and its component modules. However, this task is not trivial with many obstacles and challenges that must be overcome. Therefore, this project employs a team of collaborators from several academic and industrial institutions with each providing specific contributions necessary to achieve the desired goals and performance of CBR.

In order to test the performance of the routing components for OLSR, the routing protocol itself had to be deconstructed into its core and auxiliary functions and components. However, to do this, each class of components that form a MANET routing protocol had to be identified and appropriately defined. Despite the fact that not every routing protocol contains components from each class, every protocol is composed of elements identified during this process. The dependencies of classes of components were also investigated and considered. Once the classes of routing components were identified and defined, the performance of each the defined component modules of a particular protocol could be investigated.

The data acquired includes component module performance and overall routing performance. Component performance focuses primarily on the performance of a component as a standalone module. The overall routing performance focuses on the performance of a group of cooperating components which make up a routing protocol. The evaluation of each component has two parts: (a) identifying relevant variables and (b) investigating the effect of each of these relevant parameters.

Through a careful comparison of the component performance and overall routing system performance, inferences can be made about the relationship of each component to the performance of the overall system. Using this knowledge, designers can prioritize the overall systematic weight of each component. While some components may affect the overall performance greatly, others might not have such an influential effect on the overall routing system.

Once the performance evaluation has been performed, the system will be deployed. However, several issues arise from the deployment of CBR in itself. One potential problem is component module incompatibilities within a network that would prevent or disrupt data flow. Another issue is security. Security add-ons may affect the performance drastically depending on the scheme employed due to increase overhead and resource allocation. Other deployment questions including component module update intervals and network convergence time also exists.

3. Experiment

OLSRv2 operates as a table driven, proactive protocol developed for MANETs. In OLSRv2, each router selects a set of its neighbors as "Multi-point Relays" (MPRs). The usage of MPRs reduces the number of flooded messages, thus reducing network traffic (overhead) and increasing network performance. In OLSRv2, a MPR of a router must be selected from a node's willing one-hop symmetric neighbors. OLSRv2 uses two types of messages, Topology Control (TC) and Hello. These messages are disseminated through the network, and nodes use the information within these messages to maintain neighbor information and network topology. Messages generated in OLSRv2 employ User Datagram Protocol (UDP) at the Transport layer of the Open System Interconnection (OSI) model. In a MANET, each node behaves as a peer-to-peer (P2P) router that operates as both a client and server. But unlike in a pure P2P network, in OLSRv2, only nodes selected as MPRs have the authority to forward data.

To evaluate the OLSRv2 protocol for CBR, a client-server architecture was designed that will be used to test the performance of the component modules and overall routing system. The contributions made by myself with the assistance of others included researching, designing, and debugging this client-server architecture, which will be implemented to investigate the performance of OLSRv2. The client-server program is a distributed application architecture that divides the workload between service requesters (clients) and service providers (servers). Servers are usually high performance hosts that share resources with clients. Servers should always be available, waiting in a listen state for any service request from a client. A client is not required to share its resources with the server; however, the responsibility is on the client to initiate communication.

The client-server architecture developed assumes that each server can meet the needs of one-to-many clients. Even though OLSRv2 uses UDP, the architecture that was designed is also capable of operating in Transport Control Protocol (TCP) mode. The client-server application was designed for maximum performance flexibility and minimal structural ambiguity. As a consequence, this architecture followed a three class Unified Modeling Language (UML) model. The largest of these three classes was an abstract class called node. The node class had two generalized sub-classes, client and server. The node class contained attributes and behaviors common to both the client and server classes. Through the generalization transactions, the client and server classes inherit all operations and attributes of the node class. For this reason, node is purely an abstract class. The client and server classes were connected through an association transaction. Finally, operations and attributes were assigned to the appropriate classes. However, the implementations of the operations of the classes were not defined immediately. Before coding the implementations of the UML model, a C++ application was created.

This C++ client-server application was used as working skeleton of the UML model, which needed to be implemented. When researching how to create the client-server application, the fact that the usage of programming sockets would be necessary was immediately evident. Socket functions allow application programmers to access and control the transport OSI layer. Through the usage of socket functions, a programmer assigns ports, IP addresses, address families, and much more. Once a socket has been created, network communication is possible.

Several functions are necessary in the creation, management, and destruction of a network socket. Even though both clients and servers use sockets, they do not use them the same way. A server uses the following operation: WSAStartup(), socket(), bind(), listen(), accept(), recv() / recvfrom(), send() / sendto(), close(), and WSACleanup(). However, a client uses WSAStartup(), socket(), connect(), send() / sento(), recv() / recvfrom(), close(), and WSACleanup(). Although many functions are common, others are not. Only servers need bind(), accept(), and listen(). Likewise, only clients need connect(). Since clients and servers perform different roles, they rely on different resources to operate. The common functions were placed in the abstract node class of the UML model. However, the other functions unique to either a client or server were placed in the appropriate class in the UML representation.

In order for the client-server application to work, the server program had to be run first. This makes perfect sense, because a client cannot request a service from a service provider if the provider does not exist. When the server program runs, its behavior depends on the operating parameters. In TCP, the program runs and waits in the accept() state. This is due to the fact that TCP is a connection-oriented service; therefore, the connection must be established and approved prior to data transfer with a client. However, in UDP, the server runs and waits in the recvfrom() state. Since UDP is a connectionless service, the server is waiting and ready to receive messages from potential clients. At this point, the server is idle. Clients are run to request service from the server. In TCP, the server accepts the connection, and then receives the service request. But, in UDP, the server simply receives the service request from the client.

Once the server has received a request, the server performs some desired task(s) and sends information back to the client. The client then receives the response message from the server with the information requested by the client. If the client has more requests, then it sends them to the server. Otherwise, the client application closes. However, after handling the requests of all the requesting clients, the server goes back into an idle accept or receive-from state depending on the transport protocol. The server program does not close unless initiated by the user; however, in the near future, a wait-time timeout will be implemented to conserve resources.

4. Results

Client-server applications were constructed using three different programming Integrated Development Environments (IDEs), which included Microsoft Visual Studio 2008, KDevelop, and IBM/Telelogic Rational Rhapsody. Client-server applications were developed for both Windows and Linux. Initially, KDevelop and Microsoft Visual Studio 2008 were used to create the application using a conventional C++ programming approach. After the client-server application had been compiled and debugged using these IDEs, the code was implemented in IBM Rational Rhapsody employing UML. The generated C++ code from KDevelop and Visual Studio was used to create an object-oriented model diagram (OOMD). To bridge to gap between C++ programming and UML modeling, researchers participated in a tutorial based on Telelogic's Rhapsody UML Tutorial (version 2.1). The fundamentals acquired during the UML tutorials were used to construct the OOMD. Using these techniques, flexible client-server architecture was successfully constructed for testing OLSRv2 for CBR.

The client-server architecture designed is tremendously adaptable and permits the user(s) to specify any number of parameters related to information exchange. A flexible client-server architecture is essential, because conventional studies have shown that the performance of a routing protocol in a MANET is firmly dependent on the network environment. Therefore, to accomplish the best performance in a dynamic network atmosphere, the routing procedure itself should be dynamic. Consequently, this obliges the implementation of a flexible client-server architecture. Since the network constantly changes, the client-server needs constantly change. A flexible client-server architecture ensures maximum compatible between client-server information exchanges.

5. Conclusions

The client-server application produced will be used to test the performance of the component modules and overall routing system of OLSRv2. The results of the performance test will be used to implement CBR more effectively. With an enhanced knowledge of OLSRv2 and its modules performance, advantages, and disadvantages in particular scenarios and operating conditions, the decision making process of which class component modules to use in CBR can be improved. Improvements in class component module selection should lead to improved routing efficiency for dynamic networks. With newly acquired data, researchers might be able to exploit properties that were previously uninvestigated. At worst, this research will only widen the area of related research knowledge, questions, phenomena, and interests.

6. References

1. Stallings, W. *Wireless Communications and Networks*; Prentice Hall, November 2004.
2. Reese, G. *Distributed Application Architecture*; Sun Microsystems, 127–145, November 2000.
3. Stevens, W.; Fenner, B.; Rudoff, A. *Unix Network Programming*; The Sockets Networking API, 3, Addison Wesley, 2003.
4. Schildt, H. *C++ The Complete Reference Third Edition*; Osborne McGraw Hill, August 1998.
5. Hunt, J. *The Unified Process for Practitioners: Object-oriented Design, UML, and Java*; Springer, 2000.
6. Johnson, D.; Maltz, D.; Broch, J. *The Dynamic Source Routing Protocol for Mobile Ad-hoc Networks*; IETF Internet Draft, April 2003.
7. Jacquent, P.; et al. *Optimized Link State Routing Protocol*; IETF Internet Draft, November 2000.
8. Scott, L. Personal communication. ARL-CSID, Adelphi, MD, June 2009.
9. Bohacek, S. Personal communication. ARL-CSID, Adelphi, MD, June 2009.

List of Symbols, Abbreviations, and Acronyms

CBR	Component-Based Routing
DSR	Dynamic Source Routing
IDE	Integrated Development Environment
MANE	Mobile Ad-hoc Network Emulator
MANET	Mobile Ad-hoc Network
MPRs	Multi-point Relays
OLSRv2	Optimized Link State Routing
OOMD	Object-oriented Model Diagram
OSI	Open System Interconnection
P2P	Peer-to-Peer
TC	Topology Control
TCP	Transport Control Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language

NO. OF COPIES	ORGANIZATION
1 ELEC	ADMNSTR DEFNS TECHL INFO CTR ATTN DTIC OCP 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1	DARPA ATTN IXO S WELBY 3701 N FAIRFAX DR ARLINGTON VA 22203-1714
1 CD	OFC OF THE SECY OF DEFNS ATTN ODDRE (R&AT) THE PENTAGON WASHINGTON DC 20301-3080
1	US ARMY INFO SYS ENGRG CMND ATTN AMSEL IE TD A RIVERA FT HUACHUCA AZ 85613-5300
1	COMMANDER US ARMY RDECOM ATTN AMSRD AMR W C MCCORKLE 5400 FOWLER RD REDSTONE ARSENAL AL 35898-5000
1	US ARMY RSRCH LAB ATTN RDRL CIM G T LANDFRIED BLDG 4600 ABERDEEN PROVING GROUND MD 21005-5066
3	US ARMY RSRCH LAB ATTN IMNE ALC HRR MAIL & RECORDS MGMT ATTN RDRL CIM L TECHL LIB ATTN RDRL CIM P TECHL PUB ADELPHI MD 20783-1197
3	PRAIRIE VIEW A&M UNIVERSITY ATTN DEPT. OF ELECTRICAL AND COMPUTER ENGINEERING D VAMAN J ATTIA J JAMES PO BOX 519, MAIL STOP 2520 PRAIRIE VIEW TX 77446-0519

NO. OF COPIES	ORGANIZATION
4	US ARMY RESEARCH LAB ATTN RDRL CIN T B RIVERA L SCOTT R HARDY ATTN RDRL DO V EMERY ADELPHI MD 20783-1197
1	US ARMY RESEARCH OFFICE ATTN RDRL ROI N R ULMAN BLDG 4300 RESEARCH TRIANGLE PARK DURHAM NC 27703

TOTAL: 17 (1 ELEC, 1 CD, 15 HCS)

INTENTIONALLY LEFT BLANK.