**In compliance with the
Canadian Privacy Legislation
some supporting forms
may have been removed from
this dissertation.**

**While these forms may be included
in the document page count,
their removal does not represent
any loss of content from the dissertation.**

Performance Behavior of Mobile Agents in the Distributed Search

By

Abdul-Rahman Mawlood-Yunis

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of
Master of Science in Information and Systems Science

School of Computer Science
Carleton University
Ottawa, Ontario

September 2, 2003

© Copyright

2003, Abdul-Rahman Mawlood-Yunis

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisisitons et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-88779-0

Our file Notre référence

ISBN: 0-612-88779-0

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

Canada

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis,

Performance Behavior of Mobile Agents in the Distributed Search

submitted by

Abdul-Rahman Mawlood-Yunis

Professor Douglas Howe
(Director, School of Computer Science)

Professor Nicola Santoro
(Thesis Supervisor)

Dr. Amiya Nayak
(Thesis Co-Supervisor)

Carleton University

September 2, 2003

Abstract

In this thesis, we study the performance behavior of mobile agents in the distributed search. The search problem is analyzed from two different standpoints: the single agent search and the multiple agents search. In the case of multiple agents search, we suggest three different algorithms and study their difference. We provide an architecture to implement a distributed search, which involves single and multiple agents as well as other components that guarantee agent coordination. Several experiments were carried out to measure the performance of the mobile agents to solve the distributed search problems under a variable network size, network topologies and number of agents. The test results of two different mobile agent platforms were also compared. The results indicate that the two mobile agent platforms have similar behavior but their performance varies with underlying implementation, and the multiple agents approach performs better than the single agent approach in large networks. Moreover, the hypercube topology provides the best performance among all topologies considered.

Acknowledgments

I would like to thank my thesis supervisors Professor Nicola Santoro and Dr. Amiya Nayak for their guidance and encouragement throughout the research. The thesis has been made considerably more intelligible through their painstaking reviews. It has been a very rewarding experience to work with them.

I would like to express my most sincere gratitude to Dr. Doron Nussbaum for his time and valuable suggestions.

Many thanks to Mr. Elio Velazquez for his help. His meetings and discussions were very useful and they saved me a lot of time and pain.

Many thanks to my friend William Couch for his thoughts and comments on the grammar and English sentences.

Most of all, I would like to express my deepest gratitude to the most important friend in my life, my wife Arkhawan.

Contents

1	Introduction	1
1.1	Overview	1
1.2	Previous Works	3
1.3	Thesis Contribution	12
1.4	Thesis Outline	14
2	Agent and Mobile Agent Systems	15
2.1	Agent Overview	15
2.2	Mobile Agent Platforms	19
2.2.1	D'Agent	19
2.2.2	Messenger	20
2.2.3	Concordia	21
2.3	AGLET and TACOMA Mobile Agent Systems	22
2.3.1	TACOMA	22
2.3.2	AGLET	26
2.3.2.1	AGLET Component and Operations	26
2.3.2.2	AGLET Transfer	28
3	Mobile Agent Architecture	31
3.1	Introduction	31
3.2	The Architecture Components	32

3.2.1	Blackboard	32
3.2.2	Whiteboard	33
3.2.3	Router	34
3.2.4	Log	35
3.3	Distributed Search Algorithms	36
3.3.1	Single Agent Search	36
3.3.2	Multiple Agent Search (Flooding)	37
3.3.2.1	Remote Flooding	38
3.3.2.2	Local Flooding	38
3.3.2.3	Regular Flooding	38
4	Architecture and Algorithm Implementation	39
4.1	Introduction	39
4.2	Architecture Implementation	40
4.2.1	Blackboard	40
4.2.2	Whiteboard	41
4.2.3	Router	43
4.2.4	Log	44
4.3	Algorithm Implementation	45
4.3.1	The Single Traveling Agent Procedure	47
4.3.2	The Single Traveling Agent Algorithm	50
4.3.3	Flooding Agent Procedure	52
4.3.4	Local Flooding Algorithm	56
4.3.5	Remote Flooding Algorithm	58
4.3.6	Regular Flooding Algorithm	60
5	Implementation and Experimental Results	64
5.1	Introduction	64

5.2	Test Environment	65
5.3	Test Results	66
5.3.1	Variable Network Size	66
5.3.2	Variable Number of Neighbors	70
5.3.3	Different Topologies	72
5.3.4	Three Different Flooding Algorithm	80
5.3.5	Platform Overhead	84
5.3.6	Cost of Overhead Associated with the Search	85
5.4	Summary of Experimental Result	90
6	Conclusions	91
6.1	Thesis Summary	91
6.2	Future Work	93
	Bibliography	97
A	Remote Flooding With Variable Network Size	102
B	Local Flooding With Variable Network Size	117
C	Regular Flooding With Variable Network Size	132
D	Single Agent With Variable Network Size (BFS)	146
E	Single Agent With Variable Network Size (DFS)	154
F	Average Execution On Different Network Size And Algorithm	162
G	Variable Number of Neighbors and Algorithm	166
H	Different Topology and Variable Network Size	169
I	AGLET and TACOMA'S Moving Cost	175

List of Tables

5.1	Performance of Single and Multiple AGLET and TACOMA Agents on Different Networks	71
5.2	Single and Multiple AGLET and TACOMA Agent Performances with Variable Number of Neighbors	74
5.3	Performance Comparison of Three Flooding Algorithms	85
A.1	Remote Flooding (N =3)	103
A.2	Remote Flooding(N =5)	104
A.3	Remote Flooding(N =7)	105
A.4	Remote Flooding(N =9)	106
A.5	Remote Flooding(N =11)	107
A.6	Remote Flooding(N =13)	108
A.7	Remote Flooding(N =15)	109
A.8	Remote Flooding(N=17)	110
A.9	Remote Flooding(N=19)	111
A.10	Remote Flooding(N =21)	112
A.11	Remote Flooding(N =23)	113
A.12	Remote Flooding(N =25)	114
A.13	Remote Flooding(N =27)	115
A.14	Remote Flooding(N =28)	116

B.1	Local Flooding ($N = 3$)	118
B.2	Local Flooding ($N = 5$)	119
B.3	Local Flooding ($N = 7$)	120
B.4	Local Flooding ($N = 9$)	121
B.5	Local Flooding ($N = 11$)	122
B.6	Local Flooding ($N = 13$)	123
B.7	Local Flooding ($N = 15$)	124
B.8	Local Flooding ($N = 17$)	125
B.9	Local Flooding ($N = 19$)	126
B.10	Local Flooding ($N = 21$)	127
B.11	Local Flooding ($N = 23$)	128
B.12	Local Flooding ($N = 25$)	129
B.13	Local Flooding ($N = 27$)	130
B.14	Local Flooding ($N = 28$)	131
C.1	Regular Flooding ($N = 3$)	133
C.2	Regular Flooding ($N = 5$)	134
C.3	Regular Flooding ($N = 7$)	135
C.4	Regular Flooding ($N = 9$)	136
C.5	Regular Flooding ($N = 11$)	137
C.6	Regular Flooding ($N = 13$)	138
C.7	Regular Flooding ($N = 15$)	139
C.8	Regular Flooding ($N = 17$)	140
C.9	Regular Flooding ($N = 19$)	141
C.10	Regular Flooding ($N = 21$)	142
C.11	Regular Flooding ($N = 23$)	143
C.12	Regular Flooding ($N = 25$)	144
C.13	Regular Flooding ($N = 27$)	145

D.1	Single Agent (BFS)(N = 3)	147
D.2	Single Agent (BFS)(N = 5)	147
D.3	Single Agent (BFS)(N = 7)	148
D.4	Single Agent (BFS)(N = 9)	148
D.5	Single Agent (BFS)(N = 11)	149
D.6	Single Agent (BFS)(N = 13)	149
D.7	Single Agent (BFS)(N = 15)	150
D.8	Single Agent (BFS)(N = 17)	150
D.9	Single Agent (BFS)(N = 19)	151
D.10	Single Agent (BFS)(N = 21)	151
D.11	Single Agent (BFS)(N = 23)	152
D.12	Single Agent (BFS)(N = 25)	152
D.13	Single Agent (BFS)(N = 27)	153
D.14	Single Agent (BFS)(N = 28)	153
E.1	Single Agent (DFS)(N = 3)	155
E.2	Single Agent (DFS)(N = 5)	155
E.3	Single Agent (DFS)(N = 7)	156
E.4	Single Agent (DFS)(N = 9)	156
E.5	Single Agent (DFS)(N = 11)	157
E.6	Single Agent (DFS)(N = 13)	157
E.7	Single Agent (DFS)(N = 15)	158
E.8	Single Agent (DFS)(N = 17)	158
E.9	Single Agent (DFS)(N = 19)	159
E.10	Single Agent (DFS)(N = 21)	159
E.11	Single Agent (DFS)(N = 23)	160
E.12	Single Agent (DFS)(N = 25)	160
E.13	Single Agent (DFS)(N = 27)	161

E.14 Single Agent (DFS)(N = 28)	161
F.1 Average Execution Time with Various Network (Remote Flooding) .	163
F.2 Average Execution Time with Various Network (Local Flooding) . .	163
F.3 Average Execution Time with Various Network (Regular Flooding)	164
F.4 Average Execution Time with Various Network (DFS)	164
F.5 Average Execution Time with Various Network (BFS)	165
G.1 Average Execution Time with Various Number of Neighbors (Remote Flood- ing)	166
G.2 Average Execution Time with Various Number of Neighbors (Local Flooding)	167
G.3 Average Execution Time with Various Number of Neighbors (Regular Flood- ing)	167
G.4 Average Execution Time with Various Number of Neighbors (Single Agent (DFS))	168
G.5 Average Execution Time with Various Number of Neighbors (Single Agent (BFS))	168
H.1 Bidirectional Ring with Flooding (N = 16)	170
H.2 Unidirectional Ring with Flooding (N = 16)	171
H.3 Hypercube with Flooding (N = 16)	172
H.4 Bidirectional Ring with Single Agent (N = 16)	173
H.5 Unidirectional Ring with Single Agent (N = 16)	173
H.6 Hypercube with Single Agent (N = 16)	174
I.1 AGLET and TACOMA'S Moving Cost	176

List of Figures

2.1	Mobile Agent Structure	17
2.2	Briefcase (with folders) and File Cabinets (with folders)	23
2.3	Transferring of a Briefcase	25
2.4	Meet Operation	25
2.5	Hosting of Incoming Aglet	26
2.6	AGLET Transfer	30
3.1	The Proposed Architecture	32
3.2	Distributed Structure with Agent Architecture	37
4.1	Communication Between Agent and Blackboard	42
4.2	Agent Method Invocation	43
4.3	Collaboration in the Itinerary Pattern	47
4.4	The BFS Search	49
4.5	The DFS Search	50
4.6	Remote flooding algorithm	52
4.7	Local flooding algorithm	53
4.8	Regular flooding algorithm	54
5.1	Average Execution Time for Single and Multiple AGELT Agent . . .	69
5.2	Average Execution Time for Single and Multiple TACOMA Agent .	69
5.3	AGLET VS. TACOMA with Variable Network Sizes	70

5.4	Single and Multiple Mobile Agents with Variable Number of Neighbors (AGLET)	72
5.5	Single and Multiple Mobile Agents with Variable Number of Neighbors (TACOMA)	73
5.6	AGLET Vs. TACOMA with Variable Number of Neighbors	73
5.7	Execution Time of Single Agents on Different Topologies (AGLET) .	77
5.8	Execution Time of Single Agents on Different Topologies (TACOMA)	77
5.9	AGLET VS. TACOMA with Different Topology (Single Agent) . . .	78
5.10	Execution Time for Multiple Agents on Different Topologies (AGLET)	78
5.11	Execution Time for Multiple Agents on Different Topologies (TACOMA)	79
5.12	AGLET Vs. TACOMA with Different Topologies(Flooding)	79
5.13	Single vs. Multiple with Different Topologies (AGLET)	80
5.14	Execution of Remote Flooding, Local Flooding and Regular Flooding for Various Network Size	84
5.15	Execution Time for AGLET and TACOMA	86
5.16	Execution Time of the Single Agent With and Without Overhead . .	87
5.17	Execution time of the Multiple Agents With and Without Overhead .	88
5.18	The Overhead Time Produced from Contacting the Router for Single and Multiple Agents	89

Chapter 1

Introduction

1.1 Overview

This thesis studies the performance behaviors of mobile agents in a distributed search. The search has been conducted under variable network sizes, network topologies and number of agents. An application can divide its task into sub-tasks and assign each specific task to an agent by deploying several agents into the network. Deploying multiple agents into a network raises some issues such as coordination between agents and prevention from duplication effort. These issues and their solutions have been discussed in this thesis.

An architecture to implement a distributed search, which involves single and multiple agents as well as components that guarantee agent coordination, has also been provided. Several experiments are carried out to measure the performance of mobile agents and to discuss solutions to the distributed search problem. The results on two different mobile agent platforms are then compared.

Agent [9] is a computational entity which acts on behalf of other entities in an autonomous fashion. It performs its actions with some level of proactiveness and/or reactivity, and exhibits some level of the key attributes of learning, cooperation and mobility.

A *mobile agent* inherits some characteristics of an *Agent* and has a unique ability to *migrate* from one node to another in a network. The *mobility* is divided into two classes. One is the *Strong Mobility* and the other is the *Weak Mobility*. The difference between them is in the mode of resuming the execution. In the *Strong Mobility*, the agent can be stopped, say on line 20 in the procedural code and then the execution would be resumed on line 21 in the other host. Whereas in the *Weak Mobility*, the less restrictive transfer of the code occurs without migration of the execution state.

In [9], a mobile agent is described as an entity which exists in a software environment. It must contain all of the following models: a life_cycle model, a computational model, a security model, a communication model and finally a navigation model.

The *security* issue has always been considered a weak point of mobile agent technology. Mobile agent security can be divided to two areas:

- The security of the host that may receive a potentially malicious agent.
- The security of the agent that should be protected from potentially malicious hosts.

In order to understand the agent programming paradigm completely, we must also consider the software or mobile agent environment in which the mobile agents exist.

A *mobile agent environment* is a software system which is distributed over a network of heterogeneous computers. Its primary task is to provide an environment in which mobile agents can execute. The *mobile agent environment* provides an implementation to most of the models that are in the mobile agent definition. It may also provide support to services that are related to the mobile agent environment itself; the environments on which the mobile agent environment is built, as well as other mobile agent systems, and non_agent_based software environments.

There are reasons as to why so much research and efforts have been dedicated to the software agents and mobile agents. The mobile agents are particularly attractive in distributed information retrieval. By moving to the location of an information resource, the agent can search the resource locally, eliminating the transfer of intermediate results across the network and reducing network traffic. For more information on the reasons of using mobile agents reader can refer to [16].

The numbers and types of application domains in which agent technologies are being applied to or investigated include workflow management, network management, air_traffic control, business process re_engineering, data mining, information retrieval/management, electronic commerce, education, personal digital assistants (PDAs), scheduling diary/management, etc.

1.2 Previous Works

The agent-based distributed search problem and its performance have been subject to several published research articles. Both [19] and [5] formulate a method namely

the Traveling Agent Problem (TAP), which is analog to the well-known Traveling Salesman Problem (TSP). It decides the sequence of nodes to visit to minimize the total execution time until the desired information is found. If all sites must be visited, then the problem reduces to the classical TSP which is known to be NP-complete. Formally, the TAP is stated as follows:

There are $n + 1$ sites, where s_i is with $0 \leq i \leq n$. Each site has a known probability, $0 \leq p_i \leq 1$, of being able to successfully complete the agent's task and a time, $t_i \geq 0$, required for the agent to attempt the tasks at s_i regardless of whether it is successful or not. Travel times or latencies for the agent to move between sites are also known and given by $l_{ij} \geq 0$ for moving between sites i and j . When the agent's task has been successfully completed at some site, the agent must return to site 0 (i.e. s_0) from which it started. It has been verified in [19] that the problem could be solved in a polynomial time if the latencies l_{ij} between all nodes are the same and equal to l , where the computed time t_i and probabilities p_i are arbitrary. The optimal tour of the TAP is attained if the nodes are visited in decreasing order of $p_i/(t_i + l)$.

TSP adapted to the mobile software agents, which is called Mobile Agent Planning (MAP), has been dealt with in [2]. Although MAP is quite similar to the TSP, agent planning has a different objective function from that of TSP. TSP deals with optimal total routing cost, while MAP attempts to minimize the execution time to complete tasks of information retrieval. Mobile agents can visit any node more than once in MAP, but this is not normal in TSP. MAP, unlike TAP, allows parallel computation.

In [2], two heuristic algorithms for finding the minimal number of traveling agents for retrieving information from a distributed environment are given, while it has been proposed to keep the latency minimal. [2] first assumes that the mobile agents know

the network statistics such as the latency and bandwidth of the links between hosts as well as the loads on each host which are collected by network monitoring module in the system. Next, it is assumed that the network latencies vary and agents are created only at home and are not cloned in other nodes. Furthermore, the probability of success in retrieving the required data at each node is not considered as it is the case with [19]. To get the latencies of all pairs of nodes, they process the all-pairs shortest-path algorithm, and construct a shortest-latency network graph before entering the main body of their algorithm.

When all the smallest latency information (i.e. $L_s(h_i, h_j)$) between any two nodes known as h_i and h_j) are available, the MAP can be described as follows:

1. Sort the nodes in decreasing order of the routing time of each node, h_i , which is $TourT(h_i)$.
2. Set the threshold δ , with the execution time of the first node in the sorted list.
3. Partition the given network into several parts by gathering nodes so that the execution time of each part does not exceed the threshold and then build a routing path for each partition.
4. Run the TSP algorithm to optimize each routing path.
5. Allocate and send an agent for each partition.

The extended version of [2] has been presented in [1] to support the time constraints. Consider the nodes that present correct information only for some time intervals such as the stock exchange or currency rate. If an agent is sent and arrives earlier than a specified update time to gather information, it may retrieve useless or corrupted information. To cope with the aforementioned situation, an extended version of [2] is presented in [1]. [1] follows all steps and the procedures of [2] and imposes some extra

conditions. For example, the nodes, which can be involved during the planning time must satisfy the following:

$\max\{\text{Ready}(h_i), L_s(h_i, H)\} + \text{Comp}(h_i) \leq \text{Dead}(h_i)$, where $1 \leq i \leq n$, $\text{Ready}(h_i)$ is a time that information is ready at node h_i , $\text{Dead}(h_i)$ is deadline for information pick up at node h_i , $\text{Comp}(h_i)$ is computation time at node h_i and $L_s(h_i, H)$ is shortest latency between nodes h_i and Home

The Dynamic Mobile Agent Planning (DMAP) called the *n-array agent chaining* has been proposed by [3] rather than Static Mobile Agent Planning (SMAP), which has been presented by [2]. DMAP is similar to the SMAP in the sense that both use current statistical information about network traffic conditions and network status history to establish an itinerary plan. DMAP is sensitive to the condition of the network with the ability to change the agent's itinerary appropriately. DMAP uses the information related to turn-around time, itinerary and alternative paths. This is provided by SMAP and used to find a new itinerary taking into account dynamic traffic fluctuations on the network. When a mobile agent arrives at a node, it will process its work. The network-sensing facility reports to the agent the current latency of the path (itinerary), which the agent then presents. As soon as the on-line network information is returned to the agent, the next node is decided. In some cases in which an agent decides to skip some nodes, it produces clones to handle these skipped nodes. The aim of the cloned agent is to visit more uncovered nodes, and then return to the home node within the turn-around time. Cloned agents do not guarantee the visiting of all nodes, although they are guaranteed to return home within a given turn-around time.

An algorithm has been presented in [20] to determine the agent's tour. This algorithm does not depend on known network latencies. For mobile agents, the distance

between the sites is often unknown or get outdated in a short time compared to the time between two visits. The basic idea of [20], in contrast to the former publications, is that the cost matrix is unknown at the beginning. As the agent migrates through the network, it measures the latency taken for each step. So the cost matrix gradually gets filled. The optimization is done by comparing the examined strategy with the common approach of taking a *random tour* and keeping it for the repetitions. [20] shows that the optimal threshold point depends on the number of visits the agent takes or, equivalently, the number of tours taken within that period where the collected network information does not get outdated. Using the proposed algorithm, [20] claims that the total latency time can be halved (against the common strategy of a random way) when taking 30 runs, without producing additional network traffic.

The performance aspects of the mobile agents have been a subject of several studies. A common performance concern about the mobile agent systems is that they shift much of the processing load from the clients to the server. This shift is a significant advantage in some environments, where the client could be small computers with limited memory and computational power, while the server could be a large multi-processor computer. On the other hand, the shift increases the server workload and raises questions about scalability. In [7], the scalability of the server, the performance effects that are fundamental to the mobile agent idea, and, the performance effects due to implementation choices made by the different mobile agent platforms have been addressed. They compare a traditional client/server approach with a mobile agent approach on four mobile agent's platforms in the context of a simple informational retrieval (IR) application.

The IR task in [7] involves two steps: a keyword query selects a set of documents from the collection, and then a filter procedure scans the selected documents to return those that contain a specific string. The document server implements only the keyword

query operation. In the client/server implementation, the selected documents were returned to the client, which ran the filter. In the agent implementation, the agent filters the documents at the server and then it carries resulting subset back to the client host.

In the experiments presented in [7] the number of clients was varied from (1 to 20, each on a separate machine), the network bandwidth to the server (1, 10, 100 mbps), shared by all clients. Although in both implementations all documents were scanned, only a fraction of (5% and 20%) of the document brought back ignoring the actual results of the filter. The experiment was run on a set of a twenty-one identical Linux workstations. Twenty of the machines acted as a client and one acted as the document server.

[7] concludes that the mobile agent approach outperforms the client-server approach as long as the application involves analysis of enough information, and there is enough reduction of the data returned to the client to outperform the overhead of sending the mobile agent in the first place. Mobile agents can be beneficial in situations with low network bandwidth and plentiful server capacity. Indeed, in many environments it is easier to add more server capacity than to add network capacity, particularly those with wireless networks. The four mobile agents have similar behaviors, but their absolute performance varies with underlying implementation choices.

In [23], is analyzed the mobile agent performance in network management and its comparison with the client/server model used by the SNMP (Simple Network Management Protocol). The Mole platform is used for the mobile agent computation. Performance results are obtained from *implementation* and *simulation* studies. Prototypes of an application that gathers MIB-II (Management Information Base-II) variables have been created and tested on a LAN.

The mobile agent implementation consists of one mobile agent, which migrates to all network elements to be managed. In SNMP implementation, the manager sends a SNMP packet to an SNMP agent that responds to this manager. The manager sends requests to all elements to be managed, one after the other until the last network element receives the request and sends the response to the manager.

The considered performance parameters are bandwidth utilization and response time in retrieving the MIB-II variable `ifInErrors` from elements to be managed. This variable denotes the number of received packets discarded because of errors.

[23] concludes that the agent performs better than the SNMP when the number of managed elements range between two limits determined by the number of messages that passed through the network and by the mobile agent size that grows with the variables collected on the network elements.

The mobile agent size increases with the number of visited nodes and, consequently, migration becomes difficult. [23] evaluates the performance gain of a strategy on which the mobile agent *returns* to the management station to reduce the mobile agent size.

The experiment considers that the mobile agent visits a fixed number of nodes per trip, returns to the management station to unload the collected data, and then restarts the task of collecting variables on the remaining nodes.

Results show that when the network size is large, for a small number of managed elements per trip, response time improves sharply. As the number of visited nodes per trip increases, response time decreases up to a point when response time starts to decline due to agent migration difficulty related to the agent size. For the number of bytes, it improves when the number of managed elements per trip increases.

The strategy of *sending* the results to the management station, instead of returning

to this station, is also considered. This strategy performs better than the one of *returning* to the management station when the number of elements managed per trip is small. As the number of elements per trip increases, both approaches behave almost in the same way due to the fact that the size of the data gathered by the agent is much greater than the size of the mobile agent's code and because of the smaller number of returns to the management station.

For a smaller number of managed elements per trip, bandwidth utilization improves sharply when the size of the network increases. As the number of visited nodes per trip continues to rise, bandwidth utilization continues to decrease up to a point where it approaches a value because of the greater size of the agent that generates fragmentation and therefore more acknowledgments.

[23] concludes that significant improvements are achieved when the mobile agent returns or sends data to the management station after visiting a fixed number of nodes. The influences of other elements, such as the protocol in use, the size of the query, the topology and the platform have an effect on the performance have also been studied in [23].

Performance evaluation of mobile agents for e-commerce applications have been studied in [12]. Agent shipment and messaging costs for Aglet, Concordia and Voyager mobile agent platforms have been evaluated, and experiments to evaluate the different mobility implementation strategies for mobile agents and client/server have been performed.

In [11], an analytical model that examines the claimed performance benefits of mobile agents over client/server computing for a *mobile* information retrieval scenario have

been developed. [11] evaluate the analytical model by studying the effect of the probability of loss over the wireless link on the client/server and mobile agent paradigms in two cases. First, if the size of the messages for the client/server calls and the size of the mobile agent are the same, the probability of loss over the wireless link increases the expected latency. When using mobile agents, this increase is far less than the expected latency increase with client/server calls. Second, [11] consider the fact that the size of the mobile agent may be M times the size of the client/server calls. In this case, they also assume that the processing time for the mobile agent is also going to be M times the processing time required for client/server calls. The probability of loss over the wireless link is fixed at 0.5. With this case, they find that as the size of the mobile agent increases, the advantage of mobile agents over client/server calls is lost. In both tests, the increased size of client/server response is ignored.

[11] conclude that it is not clear that the mobile agent is always preferable to client/server computing, or vice versa. In particular, if the code size of the mobile agent exceeds that of the corresponding client/server request message for a given query, the response time using mobile agents can be significantly more than using client/server calls. On the other hand, if the user has a wireless link connecting the client device to the fixed wired network of the virtual enterprise, the improvement in response time using mobile agents compared to client/sever computing increases as the quality of the link decreases.

A performance evaluation of the mobile agent model by comparison with the client/server model has been presented in [10]. This evaluation was performed in the Java environment using RMI, the AGLET mobile agent and a MINIMAL MOBILE AGENT that have been developed locally. The performance evaluation reveals the cost of the basic mechanisms involved in the implementation of a mobile agent's platform on top of Java, and an evaluation of the client/server and mobile agents. [10] concludes that

the use of mobile agents can lead to significant performance improvements, especially when the agent migration allows saving communications between a client and invoked server.

In [26], the author has proposed a model to solve a generic search problem based on software agent technology. The architecture involves two major approaches, mobile agents and stationary agents. It also comprises several components to guarantee the agent coordination and interactions as well as some fault tolerance mechanisms. For experimental purposes, the agents have been implemented on a TACOMA agent platform. The author has reduced the distributed search problem in a network of arbitrary topology to the implementation of a traversal algorithm applied to a logical spanning tree constructed on top of the original topology. Solutions for the single and multiple traveling agents have been provided along with supporting components to implement agent coordination. The author has explored several variables that may affect the performance of a mobile agent solution, such as topology, size of the network and number of agents involved in the search. The test results show that the number of agents has a positive direct impact on performance when the size of the network increases. Furthermore, the network topology does impact the overall performance. For these particular experiments, the binary spanning tree shows the best performance among other topologies tested.

1.3 Thesis Contribution

The objective of this thesis, which extends the work of [26], is to determine the optimal conditions under which mobile agents will produce the best performance in distributed search.

This thesis has two major contributions. First, we provide an architecture to implement a distributed search using single and multiple agents. The proposed architecture includes two major parts: an agent and components that guarantee agent coordination. Unlike [26], our work is based on a different mobile agent platform (Aglet) and consider RMI server as an alternative to the stationary agent for implementing stationary component in our model. Several experiments are carried out to measure the performance of mobile agent's solution to the distributed search problem. The search has been conducted under a variable network size, network topology and number of agents. The result shows that the multiple agent search performs better than the single agent search when the network size becomes large. The hypercube topology also provides the best performance among all topologies considered. We consider three different algorithms for implementing multiple agent search whereas [26] considers just one such algorithm.

The second contribution is that we compared the performance behavior of mobile agents on two different mobile agent platforms, namely the AGLET and TACOMA systems. Unlike [26], we measure the overhead time associated with contacting the stationary components during search to get a better understanding of the actual search time and the overhead. The test results show that the overall performance behavior of both systems are similar. It seems to indicate that the overall behavior of the mobile agents in a distributed search is platform independent. Secondly, for all tests we performed, the performance of AGLET agents were much better than performance of TACOMA agents. Our results show that the performance of the mobile agent is platform dependent.

1.4 Thesis Outline

The structure of this thesis is as follows: In Chapter 2, we define Agents and some of its general behavior, then we describe some mobile agent platforms and their essential characteristics. Since the TACOMA and AGLET mobile agents are an essential part of this thesis, we devote a complete section to describe the TACOMA and AGLET mobile agent system. In Chapter 3, we give an overview of the proposed mobile agent architecture and the distributed search algorithms. In Chapter 4, we describe the implementation of different components of the proposed architecture and the distributed search algorithms. In Chapter 5, the experimental results will be presented and discussed, and finally in Chapter 6 the conclusions, limitations and possible enhancements to the current model will be discussed.

Chapter 2

Agent and Mobile Agent Systems

In this chapter we first define the Agent and some of its general behaviors, and then we describe some mobile agent platforms and their essential characteristics. Finally we discuss mobile agent systems, which are an essential part of this thesis.

2.1 Agent Overview

In some way or another, we have all had some experience with an *Agent*. For example, if we considered a real estate agent working for the property owners in order to help sell their property. This demonstrates the first key property of the Agent, *acting on behalf of others*. [9].

It is possible that the real estate agent arranged appointments for the unoccupied properties without permission of the owners. This demonstrates the second key characteristic of an Agent, which is that it has some degree of *autonomy*. This allows the Agent to act without involvement from the user by using the data and logic that it encompasses. As such, the user can perform other activities at the same time. This

autonomy characteristic assists in reducing the task time since the interaction with the user has been eliminated. [17].

The degree of *pro-activeness* and/or *re-activeness* is the third key characteristic of an Agent. In general, Agents respond quickly and appropriately to changes, which occur to their environment. Not only do they react to their environment, but they are also goal oriented with respect to their response to the changes in the environment.

Finally, characteristics such as *learning*, *cooperation* and *mobility* might also be exhibited. In [9], the concept of Agent is summed up as a computational entity that acts on behalf of other entities in an autonomous way carrying out its actions with some degree of pro-activeness and/or re-activeness. It might own some other attributes such as learning, cooperation and mobility.

A *mobile agent* inherits some of the characteristics of an Agent and has the unique ability to *migrate* from one host in a network to another. In [9], the *mobile agent* is described as a software environment with a computation entity that must hold all the following models: a *life cycle model*, a *computational model*, a *security model*, a *communication model* and finally a *navigation model*. Figure 2.1 is an illustration of the structure of a mobile agent defined by [9].

The computational aspect of the agent structure is central to describing how the agent deals with other agents, hosts and resources. Moreover, it also impacts the other models or parts of the agent structure. For example, the type of life cycle and security model agent process is affected by the way the computational model has been built.

Figure 2.1: Mobile Agent Structure

Security has always been considered to be a vulnerable aspect of mobile agent technology. According to [18], mobile agent security can be divided into two areas:

- The security of the host that may receive a potentially malicious agent.
- The security of the agent that should be protected from potentially malicious hosts.

The *life cycle model* defines the agents various states. The *agent model* defines the intelligent agent feature of a mobile agent such as learning and collaboration. In order for the agents not to be corrupted by other agents or hosts, the *communication model* is strongly dependent on the security aspect of the agent. Finally, the agent has mobility and is able to move to the host where the data is stored through the *navigation model*.

By moving the agent to the data's location, transferring large volumes of data over the network can be avoided. Once this move takes place, the data will be processed locally with the results being sent back to the user. Consequently, we are able to better use the network resources (i.e. request less bandwidth) and able to maintain low traffic in the network. [22].

Until now, we have discussed all aspects of the agent. By considering the software environment or the mobile agent environment in which the mobile agents could exist should aid us in completing our understanding of the agent environment.

From [9], a mobile agent environment is a software system that is distributed over the network of heterogeneous computers and is primarily used to fix up an atmosphere in which the mobile agents can execute.

There are various advantages from using mobile agent paradigms over other distributed computing paradigms. Distributed systems frequently use the communication protocols that involve multiple interactions to accomplish a given task resulting in higher volumes of network traffic. By packaging the interaction and dispatching it to a destination site where the interactions take place locally, mobile agents are able to remove the need for multiple interactions. Moreover, mobile agents can also reduce the flow of raw data in the network. When very large volumes of data are stored at remote sites, it is advantageous to locally process the data rather than having it transferred over the network. In essence, agent-based programming is used to move the computation to the data rather than data to the computation. For more information on the reasons of using mobile agents, the reader is referred to [16]

Application areas in which agent technologies are being applied to or investigated include workflow management, network management, information retrieval /management, electronic commerce, data mining, air traffic control, weather alarm system, etc.

2.2 Mobile Agent Platforms

In this section, a brief description of three well-known mobile agent platforms is provided, namely D'Agents, Messenger, and Concordia. The description is focused on the objective of the mobile agent, its programming language and the implementation of the migration process.

2.2.1 D'Agent

First released in 1995 and developed at Dartmouth College to support information retrieval-application, D'Agent was once known as Agent Tcl [7] [8]. D'Agents 2.1, which is the current version of D'Agents, supports Tcl, Java, and Scheme. Strong agent mobility is supported by D'agent systems implemented in Java and Tcl. That is not only the code and the data of the agent will be moved, but the thread control of the agent is also captured and moved along the code and data. Encryption is used for authentication and privacy of transporting agents. It also enforces the limitation on the resource that could be used by the agent.

By supporting multiple languages of the D'agent system, the agent programmer has the ability to choose the most suitable language for the task at hand. Furthermore, it

has an optimized set of string routines allowing for improved performance for string-intensive information retrieval tasks.

The most favorable targets for the D'agent project are the supporting application that requires information retrieval, organization and presentation. Some research areas of the Dagent project include the following:

- Security system mechanisms.
- Support for mobile and partially connected computers.
- Navigation, network sensing, and resource discovery tools.
- Automatic indexing, retrieval, and clustering techniques for text and other documents.

2.2.2 Messenger

Messenger¹ is a distributed system based on the principle of an autonomous object developed at the Department of Information and Computer Science, at the University of California, Irvine [4].

Messenger is written in a language called Messenger, which is a subset of C language. The Messenger language is an enhanced C language for the network navigation, coordination and operations on the strings.

Messenger supports the weak mobility and carries its own data and code as it moves throughout the network nodes. It is used for the generally distributed application

¹There are two systems with the name Messenger. Here, we refer to the system developed by the University of California.

and currently its key application area is the biomedical simulation, where the main objectives include using parallel processing to achieve faster processing as well as authorizing interactive open-ended applications through dynamic functional composition.

2.2.3 Concordia

The Concordia framework was developed at the Horizon Systems Laboratory of the Mitsubishi Electric Information Technology Center America (MEITCA). A full Concordia system is made up of several stand-alone components joined together, with the Concordia server as the main building block, where different Concordia managers exist.

Concordia is made up of a Java Virtual Machine (JVM), a Concordia Server, and mobile Agents if we looked at it from a high-level overview. The Concordia Server and mobile agents are Java programs, while the JVM is a standard environment. The design goal of the Concordia system is to provide a complete mobile agent framework. Essentially, it aims to provide a mobile, reliable, secure, persistence and comparative agent system.

Concordia support goes beyond just providing the mobility of code and data by being a strong mobility mechanism. It provides for the transmission of state information detailing where the agent has been, what it has accomplished, as well as where it is going and what it still has to do [27] [6].

2.3 AGLET and TACOMA Mobile Agent Systems

An important part of this study is the AGLET and TACOMA mobile agent systems. Here, we describe their objectives, main components and navigation model. We try to understand the main difference in their navigation model for the two systems.

2.3.1 TACOMA

The TACOMA project is a collaboration project between the Department of Computer Science, University of Tromso, Norway; the Department of Computer Science, Cornell University, Ithaca, New York, USA; and the Department of Computer Science, University of California, San Diego, USA. It focuses on the operating system for agents and can be used to solve problems, which are conventionally solved by operating systems. TACOMA is implemented in C, where Version 1.2 is based on UNIX and TCP. However, the system support agents are written in C, Tcl/Tk, Perl, Python, and Scheme[14].

The TACOMA agent is the computation unit that can be stationary or mobile with the ability to act on behalf of other users. A simple TACOMA agent can be a simple C program or Tcl script. The TACOMA system distinguishes between the process and the agent even though they are both sequences of instructions. The agent is considered a process by TACOMA. Moreover, not all of the agents instructions have to be processed on the same node.

The concept of the folder, briefcase and file cabinet is used with TACOMA. They are used for leaving data on a certain site as well as carrying data when an agent is about to move. Each site can have one or more file cabinets, while each file cabinet

can contain more than one folder. Moreover, each folder inside the file cabinet has an ASCII name and is used for holding the data at the site. On the other hand, the briefcase moves with the agent and can contain multiple folders. Items can be stored or retrieved from the folders by an agent.

The TACOMA system has several general-purpose folders, which are needed in the same way we would need a common interface between the client and the server. For example, once an agent moves from one site to another, its code is stored in a special folder called the Code folder. The Data folder, on the other hand, is used to carry the data that is associated to the agent. Additionally, the TACOMA briefcase could carry a folder called a Contact folder. The Contact folder specifies the agent to contact within the remote site, while it is saved in another folder called the Host folder. Figure 2.2 illustrates these concepts, where a *briefcase* is sent between two sites.

Figure 2.2: Briefcase (with folders) and File Cabinets (with folders)

The TACOMA agents can contact each other by simply by invoking a *meet* operation. The TACOMA *meet* operation is executed by initiator agent, for example Ag_A, to cause the execution of target agent, for example Ag_B, on some host. TACOMA agents will exchange briefcase rather than contacting directly. For example, a service

may be requested from a system agent passing a *briefcase* with a folder containing service specification. A system agent can then replay to the client agent by sending a briefcase that contains a *Result folder*.

An environment for guest agents to arrive to the remote site is provided by the TACOMA system. A single entry point at the destination site is used for agent authentication, access control, counting and fault-tolerance.

Incoming agents are authenticated in the TACOMA system and their access to the local resources is limited by using a special agent called `tac_firewall`. The `tac_firewall` agent records the briefcase to a disk. The code of the agent is then extracted from the Code folder inside the briefcase and executed. Activating these incoming agents can be time consuming especially if the agent code has to be recompiled and linked.

The `tac_firewall` agent responsible for receiving incoming agents passes the activating agents task to another agent called `tac_exe` since there is only a single entry point for guest agents and activating the incoming agents is time consuming in the TACOMA system. As such, the host sites quickly accept new incoming agents. Better performance results from handing over the incoming briefcase from the `tac_firewall` to the `tac_exec`. Several `tac_exec` can exist and run several incoming agents at the same time while the TACOMA keeps only one `tac_firewall` on each entry point to receive the guest agents.

The `tac_exec` also restricts the execution atmosphere for the guest agents by allowing them to be executed only in a restricted place. In Figure 2.3, the agent `ag_local` is not running in the place allowed by the `tac_exec` for guest agents to exist in. However,

it is still running on Site 2. A guest agent activated in the place at Site 2 cannot communicate with ag_local since it is outside the place.

Figure 2.3: Transferring of a Briefcase

Figure 2.4 illustrates how the arrived briefcase has been passed to the tac_exec. The tac_exec agent activates the original agent specified in the Contact folder, ag_B in this example. The arriving agent can be a Tcl script or a C program that extracts the source code from the code folder and runs the arrived agent. Consequently, the tac_firewall is now free to accept a new arrived agent.

Figure 2.4: Meet Operation

2.3.2 AGLET

2.3.2.1 AGLET Component and Operations

The AGLET mobile java agent supports autonomous execution and dynamic navigation, where the AGLET API is an agent development kit, which is a collection of Java classes and interfaces that allow the creating of AGLET agents. The Aglets Software Development Kit (ASDK) is an implementation of this AGLET API developed by a research team at the IBM Tokyo Research Laboratory in Japan.

The AGLET concept could be considered as an extension of Java Applets with its name AGLET being derived from the word applet and the word agent [17]. Like the applet, the class files for an AGLET agent can move across a network. However, with the applet, the code only can move, whereas with the AGLET, the code and the state are capable of moving across the network. Both the AGLET and the applet require a server in order to execute on the remote site. A web browser hosts applets, while an AGLET server hosts AGLETS. The AGLET server provides an environment where Aglets can execute, while the Java Virtual Machine (JVM) runs AGLETS. See figure 2.5.

Figure 2.5: Hosting of Incoming Aglet

The Aglet, Proxy, Context, Message, Future reply, and Identifier are the key abstractions of the AGLET platform. The following is a brief description of each of those abstractions [15]:

- *Aglet*: It is a mobile java object has the ability to move around in a network if the Aglet-enable environment has been installed at each node of the network.
- *Proxy*: It represents the Aglet to the outside world and prevents direct access to the public methods of the Aglet.
- *Context*: It is a stationary object providing an execution environment for the mobile java objects, where each server could host multiple contexts and several servers could execute on one node.
- *Message*: A message is an object exchanged between Aglets, where the Aglets can synchronously and asynchronously communicate by exchanging messages.
- *Future reply*: It is an object used in asynchronous message-sending, where future replay allows the message sender to asynchronously receive a result.
- *Identifier*: A unique identifier is used by each Aglet and is kept Aglet for its life time.

Each Aglet is capable of performing several fundamental operations including: creation, cloning, dispatching, retraction, deactivation, activation, disposal, and messaging. Invoking such operations could bring agents to life, make a clone from an existing one, move agents in the network, destroy agents, and put agents to sleep state or activate them from sleeping state.

The AGLET programming model allows one to add a *listener* for each event, where listeners are objects, which catch events in the life cycle of an Aglet and allow the programmer to take proper action accordingly. For example, when an Aglet is dispatched, there are different events that can happen and different listeners could be captured accordingly such as the Clone Listener for cloning events, the Mobility Listener for mobility events and the Persistence Listener for persistent events.

Thus far, we have described the main components and operations of the AGLETS model. To finish our understanding about AGLET programming model, the next section describes a very important operation of the AGLETS, namely, the dispatching and receiving of an AGLET.

2.3.2.2 AGLET Transfer

Different elements can start the transferring agent from original site to the destination site. By calling the dispatch method, the agent itself could start the transferring operation. The transfer operation can also initiated by other agents living in the same place or non-agent systems outside the place.

Contacting the destination site for a travel request starts the process. If the request is approved, the travel process starts otherwise the operation fails. The mobile agent informs the local agent system that it is about to travel to the destination site if the request is granted and the destination is known.

When the local agent system is informed about the trip, it performs the following operations [17]:

1. Suspend the agent. The local agent system asks the agent to complete its task. When this is done, its execution thread is halted.
2. Serialize the object. Both the agents class code and the present value of the instance variables of the agent object are serialized. This is the process of creating a persistent representation of the object, where the object can be transported over the network.
3. Encode the serialized object. The engine encodes the serialized object so that it can be transferred for a chosen transport protocol.
4. Transfer the agent. Connection to the destination will be established and the agent will be transferred.

Since the sender has already been authenticated to the receiver, the following steps to reconstruct the agent on the receiver side are followed:

1. Receive the agent.
2. Decode the agent: The engine decodes the incoming data stream.
3. De-serialize the agent: The agent class is created and values are assigned to the instance variables.
4. Resume agent execution: A recreated agent starts execution in the new thread.

Figure 2.6 illustrates this process.

The AGLET class and other related classes should be available in order for an agent to resume its execution on the receiver side. Various methods to make those classes available on the remote site include the following: (1) Use a classes at the destination

Figure 2.6: AGLET Transfer

if they are available, (2) transport classes from origin, or (3) bring the code on demand where the code is at the remote site. Once required, it is requested and shipped to the local host.²

²For more information about the AGLET transfer refer to [17].

Chapter 3

Mobile Agent Architecture: Concepts, Components and Algorithms

3.1 Introduction

To be able to perform a search in a distributed environment with mobile agents, we have to choose a mobile agent platform and design a structure that will help in the navigation of the agents. In the previous chapter, we described the characteristics of the AGLET agents which we will use for our experiment later on. Here, we describe the system structure that we will use to perform our distributed search and some related issues such as the coordination between agents.

3.2 The Architecture Components

The proposed architecture consists of four main components: *Blackboard*, *Whiteboard*, *Router*, and *Log*. Figure 3.1 summarizes the architecture with all the components involved.

Figure 3.1: The Proposed Architecture

We can use all or some of these components in a distributed search depending on which experiment we are running. For example, the purpose of having the *Whiteboard* component is to avoid multiple visits to same node. This situation will not occur with *Single agent* traversal. Hence, when we search the network using a *Single agent*, there is no need for including the *Whiteboard* component in the system structure of the distributed search. Below is the description of each of those components in detail and when they are used.

3.2.1 Blackboard

The purpose of the *Blackboard* is to help terminate the flooding algorithm, which will be described in Section 3.3.2.

The *Blackboard* will support the following operations:

1. Add an agent ID (i.e. insert a new agent ID into *Blackboard*).
2. Delete an agent ID (i.e. remove an agent ID from *Blackboard*).
3. List the agents registered in the *Blackboard* at any point in time.
4. Count the number of agents in the *Blackboard*.
5. Clean the *Blackboard* from all agent IDs.
6. List all agents registered on *Blackboard* during the search.

The delete function returns the number of agents that remain in the *Blackboard*. To be able to return the number of agents, the delete function calls the count function. All these functions are synchronized; in other words they are atomic in order to avoid any race condition or deadlock. The search terminates when the last agent has been removed from the *Blackboard* (i.e. number of agent ID in *Blackboard* = 0).

3.2.2 Whiteboard

The purpose of having a *Whiteboard* is to avoid multiple visits to the same node by different agents. Depending on the topology and the algorithm used, there is a possibility that each node can be connected to several neighbors. For example, in the hypercube topology, each node is connected to several other nodes. With multiple agents traversing the hypercube network, it is possible that agents from different neighbors will try to visit the same node. To avoid that from happening, we will associate a *Whiteboard* component with each node in the network. The *Whiteboard* will hold the state of the node specifying if it has been visited or not. Before an agent visits a node, it has to first check the *Whiteboard*'s state. If the agent discovers that the node has been previously visited, it will return to *Home*. Due to the hierarchical structure of the network, when a node has been visited, it is correct to assume that the

neighbors below this node have also been visited, or atleast there are agents traveling towards them. The *Whiteboard* will provide the following list of services which can be used by the agent:

1. Check to see if the host has been visited.
2. Set the host for “visited”.
3. Delete all visitors for a particular host.
4. List all visitors for a particular host.

3.2.3 Router

The *Router* component is a very essential component of our system structure. A *Router* is associated with each node in the network. It will be used with both the single and multiple agent search described in Section 3.3. It will act as a global and local *Router* to enable agents to move around the network. When an agent arrives to the site, it will contact the *Router* asking about the next set of immediate neighbors for that particular site. The *Router* will then provide the set of next immediate neighbors. Depending on the number of agents used for the search, dealing with the newly gathered information will be different.

The *Router* will provide a way to recover from the node failure. At each *Router*, we can choose not only to save the information about local neighbors for that node, but also information about all other nodes, we call this the *Global* information. The agent will contact a *Router* again to find the set of neighbors for a failed node when an agent fails to dispatch to a particular node. By doing so, the agent guarantees that only the faulty node will be skipped and that the sub-tree below will be explored.

The list of services the *Router* will provide are as follows:

1. Get the local neighbors.
2. Set the local neighbors.
3. Get the global neighbors.
4. Set the global neighbors.
5. Delete the local neighbors.

Operations 3 and 4 are used for recovery from a node failure. We can apply the same mechanism to guarantee functional behavior and recovery for multiple node failures in the network. We can achieve this by letting the *Router* hold the global information.

3.2.4 Log

Another component which has been used in the proposed architecture is the *Log* component. A *Log* component is associated with each node in the network. The purpose of this component is to terminate the flooding algorithm without using the *Blackboard*.

The list of services the *Log* component will provide are as follows:

1. Read the number of neighbors in the *Log*.
2. Write the number of neighbors in the *Log*.
3. Reset the number of neighbors in the *Log*.
4. Write the partial result to the *Log*.
5. Read the partial results to the *Log*.

3.3 Distributed Search Algorithms

In this section, we give an overview of the distributed search algorithms that we intend to implement on the proposed mobile agent architecture. In this study, we concentrate on a search algorithm instead of the search criteria. We assume all nodes have to be visited. Thus, a complete search algorithm will be implemented. The search will be carried out by an initiator node, using either a single or multiple agents. The agent(s) will visit each node in the network, open a file at each node it visits and carries along the content of the file back to the initiator node.

The search will start by constructing a logical spanning tree on the top of the physical network. After the spanning tree has been constructed, the search problem reduces to the problem of tree traversal. Figure 3.2 represents a spanning tree on the top of a network with all the components of our proposed model for distributed search agent base.

3.3.1 Single Agent Search

Similar to [26], we apply the idea of [5] and [19] to our logical spanning tree in order to implement the traversal algorithm. It has been shown from [5] and [19] that the TAP can be solved in polynomial time if the nodes are visited in decreasing order of $p_i/(t_i+l)$, where p_i is the probability to complete the task and t_i is the time required to accomplish the task. For simplicity, [26] assume that the probability p_i to complete the task at any given nodes s_i is constant and equal to p . With this assumption, the ordering depends basically on time t_i , which is required to accomplish the task at the site s_i . The spanning tree traversal can be of two types: depth-first search (DFS) or breadth-first search (BFS).

Figure 3.2: Distributed Structure with Agent Architecture

3.3.2 Multiple Agent Search (Flooding)

Flooding is another way to traverse the spanning tree and solve the problem of distributed search. Flooding is forwarding a message from any node to every other node attached to that node except the node from which the message arrived. Flooding is a way to distribute routing information, where it updates quickly to every node in a large network. It is also sometimes used in multicast packets (from one source node to many specific nodes in a real or virtual network).

In a similar manner, in a mobile agent environment, we could flood the network with mobile agents. The agent starts from a node and replicates itself as many identical agents as necessary. All these replicas travel simultaneously to the neighboring nodes,

and the process is repeated.

In this thesis, we will consider three different flooding algorithms: *Remote*, *Local*, and *Regular Flooding*.

3.3.2.1 Remote Flooding

In *Remote flooding*, the agent and the *Blackboard* communicate remotely. Here, the agent contacts the *Blackboard* for removing of its ID once it is at the end of the search path and just before moving back to the initiator node.

3.3.2.2 Local Flooding

With *Local flooding*, the agent and *Blackboard* communicate locally on the initiator node. This contact happens just before the agent reports its partial result and after it returns back to the initiator node. The agent contacts the *Blackboard* to remove its ID and reports the partial result to the initiator node after its ID is removed.

3.3.2.3 Regular Flooding

In the case of *Regular flooding*, agents start moving from the initiator node all the way down to the end of their search path. After they reach the end of the search path, each agent reports its partial result to its immediate parent (i.e., the node they came from). This will continue until the final result reaches the initiator. Once the last agent returns back to initiator, the algorithm terminates.

Chapter 4

Architecture and Algorithm Implementation

4.1 Introduction

In Chapter 3, we introduced and explained the different components of the proposed mobile agent architecture and the distributed search algorithms. In this chapter, we present the implementation of the proposed architecture and algorithms using AGLET mobile agent. Since part of this study is comparing the results we obtained from the distributed search using the AGLET mobile agent with the results obtained by [26] using the TACOMA mobile agent, we follow the similar algorithm and steps which have been implemented in [26].

4.2 Architecture Implementation

4.2.1 Blackboard

We implement the *Blackboard* component as a multi-threading HTTP server listening on a port (8080) for requests from the agent. It waits for connections and agents requested information. Moreover, it runs on the initiator node, where this component is used only in cases of the flooding algorithm. The components sole purpose is to help terminate the flooding algorithm.

The request for *Blackboard* component comes from the nature of the flooding algorithm. In case of flooding, the agents duplicate themselves any time they arrive at a new node. Hence, the number of active agents at any given time is unknown and this depends on the chosen topology. For example, in the tree topology, the initiator will receive as many agents as the number of leaf nodes of the defined tree. Each agent will traverse a path of the tree. When an agent reaches a leaf node in a tree, it will return to the initiator with the gathered partial information. Unfortunately, the number of leafs is unknown to the initiator to terminate the search. Hence, the request for a mechanism to terminate the flooding algorithm arises, with the *Blackboard* playing this role.

The basis on which the *Blackboard* accomplishes the termination task depends on the initiator node registering the birth and arrival of agents. When a new agent is created, its ID will be added to the *Blackboard*, and the agent ID counter will be increased by one. In a similar manner, when an agent arrives at the initiator node after completing its search, its ID will be removed from the *Blackboard* and the agent counter will be decreased by one. All agents know the initiator, where they all carry a string variable called *HOME* whose value is the home address of the initiator. When the agent counter is equal to zero, the search will terminate and the partial results

will be reported to the initiator node.

Since the *Blackboard* is implemented as a multi-threading HTTP server, we need to design a protocol to enable the *Blackboard* and agents to communicate. Instead of making the communication protocol part of the agent's internal code, we create a *Relay* object. It plays the role of an intermediary between the agent and *Blackboard*, and contains all the necessary information to establish a connection between the agent and the *Blackboard*. Each agent contains a reference to this object. If an agent requires any service from the *Blackboard*, it invokes a function call on the *Relay*'s internal object. *Relay* starts communication with the *Blackboard*, gets all necessary information, and returns the required information to the agent. Then, at the end of the operation, it terminates the connection with the *Blackboard*. With this design, we can easily change implementation of the *Blackboard*. For example, we could use a RMI server or CORBA instead of multi-threaded server without any change in the agents implementation. Since the agent does not depend on any internal details of the *Blackboard*, only the *Relay* object has to be changed. The sequence of operations for setting up communication between an agent and the *Blackboard* through the *Relay* object is shown in Figure 4.1.

4.2.2 Whiteboard

The implementation of the *Whiteboard* is based on the Remote Method Invocation (RMI) technology. Unlike *Blackboard*, *Whiteboard* does not require an application-level protocol. In RMI, all information about a server's service is provided in the remote interface definition in the form of method signature. Clients should be able to tell what methods the server provides and how to invoke them simply by looking at the remote interface. Clients obtain reference to the remote object through the RMI registry, which is a simple naming service. Figure 4.2 shows how an agent obtains a

Figure 4.1: Communication Between Agent and Blackboard

reference to the remote *Whiteboard* using RMI server.

As shown in Figure 4.2, when we create a *Whiteboard*, we register it with the RMI registry and an agent then gets the reference to the *Whiteboard* by querying the RMI registry. When an agent gets the reference of the *Whiteboard* from RMI server, it will invoke a message on the *Whiteboard* as it's a local object.

Below is the summary of how we create a *Whiteboard* component using RMI technology and how we startup the RMI registry server to help find a *Whiteboard* component [18]:

1. Define a remote interface which contains the list of services the *Whiteboard* provides.

Figure 4.2: Agent Method Invocation

2. Implement the *Whiteboard* interface or the server (i.e the *Whiteboard* implementation).
3. Develop a client (i.e an agent) that uses the *Whiteboard* interface. We embed all necessary code inside the agent to make it capable of obtaining a reference to the *Whiteboard* implementation from the RMI server.
4. Generate stubs and skeletons.
5. Start the RMI registry.
6. Run the *Whiteboard* implementation and agent.

4.2.3 Router

A *Router* component is implemented as a set of (RMI) objects, each associated with a node in the network. When an agent arrives to the site, it will contact the *Router* asking about the next set of immediate neighbors for that particular site. The *Router* will provide then the set of next immediate neighbors. Depending on the number of agents used for the search, dealing with the newly gathered information will be different. In the case of a *Single agent* search, the agent attempts to travel to one of

the nodes and inserts the rest into the internal list of the agent. This follows any of the traversal algorithms (i.e., BFS or DFS). In the *multiple agent search*, the agent will duplicate as many as the number of neighbors and attempt to travel to all of the new neighbors.

The *Router* provides a way to recover from the node's failure. At each *Router* object, we can choose not only to save the information about local neighbors for that node, but also information about all other nodes, we call this the *Global* information. The agent will contact a *Router* again to find the set of neighbors for a failed node when an agent fails to dispatch to a particular node. By doing so, the agent guarantees that only the faulty node will be skipped and that the sub-tree below will be explored.

4.2.4 Log

The *Log* implementation is similar to the *Whiteboard* implementation and based on RMI technology. However the difference between the *Whiteboard* and *Log* components is not in the implementation but in the set of services they provide and the location they occupy in the whole system structure of our distributed search. Originally, we intended to use this component only when we performed distributed search with what we call a *Regular flooding* algorithm. The purpose was to terminate the flooding algorithm without using the *Blackboard* component (as is the case when using the *Local* and *Remote flooding* algorithm). We later added another duty to the *Log* component by using it for the holding of information on the overhead time an agent spends in contacting the *Blackboard*, *Whiteboard* and *Router* components. Hence, its use is not limited to the *Regular flooding* algorithm. It is used with all different searches we run, but with different duties.

In a *Single agent* search, the *Remote* and *Local flooding* algorithm plays the role of the storage that holds the overhead time resulted from the agents contact with the

other component. In *Regular flooding* search, however, it is used as a storage and helps in terminating the search algorithm.

The use of the *Log* component with the *Regular flooding* algorithm is different from its use with the single and two other flooding algorithms. In *Regular flooding* algorithm, it is used as the main component for the terminating algorithm. The *Log* component keeps track of the number of neighbors a node has and the number of those neighbors visited by an agent. Every time an agent returns from a neighboring node, the counter of the number of neighboring nodes is reduced by one until this number becomes zero. The associated node will send back an agent to its ancestor when the number of neighboring nodes becomes zero. This process is applied at each node in the network until finally we reach the initiator node and then the algorithm terminates. We also use the *Log* component for keeping all information about the time an agent spends at each node to contact *Blackboard*, *Whiteboard* and *Router*. This helps us to determine how much search time is coming from contacting other components.

4.3 Algorithm Implementation

We start by constructing a logical spanning tree on the top of the physical network. After the spanning tree has been constructed, the search problem reduces to the problem of tree traversal. The search time of the mobile agent does not include the spanning tree construction time, as we assume that the spanning tree has been constructed in the previous steps and is updated every time we change the spanning tree.

In order to traverse the spanning tree, the agent may start from any node in the network, reading its neighboring information and traveling to the first host on the list. It will change its internal container after the agent gets the neighboring information.

The implementation of adding method of the neighboring information into the internal container in *Single agent* search, leads to different spanning tree traversal such as the BFS or DFS.

The agent interacts with local resources at each node, opens a file at each site it visits; and carries along the content of the file. It will return to the initiator node to report the search result after all the nodes in the network have been visited.

We use the *Itinerary* pattern from [17] to implement the single and multiple agent search. This pattern objectifies the AGLET'S *Itinerary* and their navigation among the multiple destinations. The key idea of this pattern is to shift the responsibility for navigation from the Agent to its associate *Itinerary* object.

For both single and multiple agent searches, we created two Java classes called the *Itinerary* class and *Aglet* class. The *Itinerary* provides an interface to maintain or modify the AGLET'S *Itinerary* and dispatches the AGLET to new destinations.

An *Aglet* and *Itinerary* are connected as follows: the *Aglet* creates the *Itinerary* object initialized with a reference to the *Aglet*. Then the *Aglet* uses its own method *Go* to dispatch itself to the next available destination in its *Itinerary*. Figure 4.3 represents the collaboration between agent and itinerary.

Similar to [26], the interaction between the agents and stationary components are represented in the algorithms by calling supporting functions. The interaction between the agent and *Router* are encapsulated in the functions GET_LOCAL_NEIGHBORS and GET_GLOBAL_NEIGHBORS. The GET_LOCAL_NEIGHBORS sends a message with the name of current host as a parameter to the Java RMI server which is running on the same host and holds the information about the local neighbors of that particular host. The RMI server returns the list of local neighbors. In a similar manner, the agent can call GET_GLOBAL_NEIGHBORS to contact the local or global RMI server to get the list of immediate local neighbors for a particular host.

Figure 4.3: Collaboration in the Itinerary Pattern

The communication between the *Blackboard* component and the agent is done by invoking functions of the *bBoard* object: `Add_Agent` and `Delete_ID`. The *bBoard* object is a *relay* object between the multi-thread server *Blackboard* and the agents. Both functions establish TCP/IP connections with *Blackboard* in order to register and remove agents ID from the *Blackboard*. Similarly, the communication between the *Whiteboard* and *Log* components with agents is done through invoking `GET_AGENT_ID`, `SET_AGENT_ID` and `READ_NUMBER_OF_NEIGHBORS`, `WRITE_NUMBER_OF_NEIGHBORS`, `READ_TIME`, and `WRITE_TIME` respectively.

4.3.1 The Single Traveling Agent Procedure

Similar to the [26], the agent is implemented as a simple finite state machine with three states: `START`, `SEARCHING` and `END`. When the agent is created, it is in the `START` state. When the agent starts searching the hosts, its states changes to

SEARCHING. During the search, the agent's state stays in the SEARCHING state. At the end of the search, the agent's state changes to END. The result is reported and the search terminates.

At each host the agent searches the host and carries the result along. After the agent reaches the initiator host and it is in the END state, it reports the result and the search terminates.

At each host, the agent tries to get the list of the next host in which it has to visit. This is done by contacting the *Router* component through the function call GET_LOCAL_NEIGHBORS. The *Router* returns the list of immediate neighbors of the host to the agent. Before the obtained list from the *Router* is added to the internal container in which the agent carries along and specifies the *Itinerary*, the agent checks to see if the returned list contains the initiator host or the host in which the agent has just visited. In another words, the agent checks to see if a parent of the current host or initiator host is in the list. If the list contains the parent or initiator node, the agent removes them to avoid returning to the host that it has just came from or returning to the initiator node before completing the search. It adds the remaining hosts (if the are not null) to the internal container for visiting. The internal container or NEIGHBORING_LIST denoted as a "List" in Figures 4.5 and 4.4 is defined as follows:

$$\text{List} = \{ \text{nodes to be visited}, \{\}, \text{node already visited} \}.$$

As mentioned earlier, the different implementations of the ADDING function results in different network traversals. We implement two different ADDING methods: BFS and DFS. For the BFS traversal, every time we get a set of new neighbors from the *Router*, we add it to the end of the internal container (i.e enqueue the internal queue). This results in the BFS traversal as illustrated in Figure 4.4.

In case of the DFS, we add a new set of neighbors to the beginning of the internal

Figure 4.4: The BFS Search

list (i.e the top of the stack). This results in the DFS traversal illustrates in Figure 4.5

Note that when the internal container (NEIGHBOR_LIST) ends with an empty host name, it means that the search is completed and there are no more hosts to visit. Hence, the agent has to go back to *Home* or to the initiator node to report the result. By putting the visited host name at the end of the NEIGHBOR_LIST, we also keep the path of visited hosts.

If the attempt by an agent to travel to a host fails because the node is temporarily down, the agent contacts the *Global Router*. When the *Global Router* is contacted, it will return the list of immediate neighbors of the faulty node. By applying this technique, the algorithm guarantees that only the faulty nodes are not visited and the sub-tree below these nodes will be explored.

We also measure the time an agent spends to get the list of immediate neighbors

Figure 4.5: The DFS Search

from the *Router*. This is done by starting the timer before the agent contacts the *Router*, stopping the timer after the agent returns the list of neighbors if any exist, and subtracting the start time from the end time. This information helps to better understanding the performance of the mobile agent, and the overhead which is associated with contacting the *Router*.

4.3.2 The Single Traveling Agent Algorithm

procedure *Single Traveling Agent*

Begin

if STATE $\neq$ END **then**

 Perform_Local_Search

 Remove_Current_Host /* move the current host from the top to the end of NEIGHBORS_LIST */

if (Top_Host_Name = "") **then** /* the name of first host in NEIGHBORS_LIST is empty string */

 STATE = END

```

    DISPATCH TO (HOME)
else
    Begin
        STATE = Searching
        Start timer
        GET_LOCAL_NEIGHBORS (Router)
        Stop timer
        Partial_OverheadTime = (Stop_Timer - Start_Timer)
        ADD (NEIGHBORS_LIST)
        DISPATCH TO TOP (NEIGHBOR_LIST)
        if (Can Not Travel to Top Host) then
            Begin
                Start timer
                Get_Global_Neighbors (host, Router)
                Stop timer
                Partial_OverheadTime = (Stop_Timer - Start_Timer)
                ADD (NEIGHBORS_LIST)
                DISPATCH TO TOP (NEIGHBOR_LIST)
            endif
        /* if the state = end */
    else
        Report The Result
        Report The Search Time
        Agent Disposed
    End Single Traveling Agent Procedure

```

4.3.3 Flooding Agent Procedure

In *Remote flooding*, the agent contacts the *Blackboard* for removing of its ID once it is at the end of the search path and just before moving back to the initiator node. In this way, the agent removes its ID while being the host on remote node, and only after its ID has been removed from the *Blackboard* it returns to the initiator node. By looking at the *Remote* and *Local* algorithms, we can clearly see the difference between these two approaches. To be able to remove IDs from the *Blackboard* remotely in the *Remote* algorithm, we have one extra function called the *ON_DISPATCHING*. This function takes care of removing agent IDs remotely. In the case of the *Local flooding* algorithm, we do not have this function, where removing the IDs happens inside the *ON_ARRIVAL* function. This function is executed after the agent reaches the host. Figure 4.6 show steps of *Remote flooding* algorithm.

Figure 4.6: Remote flooding algorithm

In the case of *Local flooding*, the agent and *Blackboard* communicate locally on the initiator node. This contact happens just before the agent reports its partial result

and after it returns back to the initiator node. The agent contacts the *Blackboard* to remove its ID and reports the partial result to the initiator node after its ID is removed. Figure 4.7 show steps of *Local flooding* algorithm.

Figure 4.7: Local flooding algorithm

In the case of *Regular flooding*, agents start moving from the initiator node all the way down to the end of their search path. After they reach the end of the search path, each agent reports its partial result to its immediate parent (i.e. the node they came from). This will continue until the final result reaches the initiator. Once the last agent returns back to initiator, the algorithm terminates.

In this approach, we do not have any *Blackboard* component to facilitate termination as it is the case with two previous algorithms. At each node we keep a *Log* (i.e. local *Blackboard*) for holding information about that node. The information we keep track of is the number of neighbors or children each node has as well as, how many of those children have reported back their partial result. Figure 4.8 show steps of *Regular flooding* algorithm.

Figure 4.8: Regular flooding algorithm

The procedure for traversing the spanning tree with the multiple agent starts by launching an agent on the initiator node. The agent searches the host first and carries the result along. The agent contacts the *Router* for the list of immediate neighbors, and after it receives the list of its neighbors it does the following two things:

1. It checks to see if the returned host list from the *Router* is NULL. In other words it checks to see if the current host is a leaf without any children. If the current host is a leaf, the agent travels to the initiator node to remove its ID from *Blackboard* and reports its partial result.
2. If the returned list from the *Router* is not NULL and the current host is not a leaf, it will check each of the returned hosts by contacting the *Whiteboard* component to see if any of the hosts have already been visited by other agents. If any of the hosts have been visited already, it will be removed from the list.

After the above checks, the agent clones itself as many times as the number of neighbors left in the returned list and registers these agents with the *Blackboard*. Each of the cloned agents and the original agent will travel to a different neighbor. When an agent arrives at the neighboring host, it will repeat the above procedure.

Each time an agent reaches the initiator node, it removes its ID from the *Blackboard* and reports its partial result. The total time of the search is reported, and the search terminates after the last agent removes its ID from the *Blackboard*.

Each time we contact the *Blackboard*, *Whiteboard* or *Router*, we calculate the required time to do this communication and we register these times in the *Log* component. At the end of the search, we not only know the search time but also the overhead time which occurred as a result of contacting those components as well as the separate overhead time for each component.

A few things to note about the flooding algorithm:

1. We depend on the *Blackboard* component for terminating the search algorithm.

The agent terminates the search when the *Blackboard* declares it is empty. This could lead to some errors. For example, There is a possibility that some agents complete their search path, return to the initiator node and remove their IDs from the *Blackboard*. As the result of the removal, this makes the *Blackboard* empty, before other agents start registering with the *Blackboard*. This leads the algorithm to be terminated, where it thinks that the search is completed where in fact it has not. To avoid this situation, each agent not only registers its ID but also registers the ID of all the agents which are cloned after contacting the *Router* and are ready to dispatch to different hosts. Since each agent only removes its ID the *Blackboard* will be prevented from becoming empty until the

last agent arrives. One thing we have to be careful about here is to not register the same agent more than once with the *Blackboard*. Before registering any agent with the *Blackboard*, we have to ensure that the agent has not been registered before nor that it has been removed from the *Blackboard* after completing its search path.

2. An agent contacts the *Router* to get the list of immediate neighbors for a certain host. Once it has the list, the agent removes the initiator and the immediate parent nodes from the list (if they are in the list). In this way, we eliminate contacting the *Whiteboard* used for checking to see if those have been visited or not, since we already know that they have been visited.

4.3.4 Local Flooding Algorithm

procedure Go (destinations)

Begin

for k = 0 **to** dest.size

 begin

 nodes = nodes + destinations.elementAt (k)

endfor

 bBoard.operation (register, nodes)

for j=0 **to** dest.size -1

 begin

 aHost = dest.elementAt(j)

 cloneProxy = aglet.clone ()

 cloneProxy.Dispatch (aHost)

endfor

 aglet.Dispatch (aHost)

```

    if ( Dispatch failed )
        neighbors = obj.getGlobalNeighbours (host)
        Go (neighbors)
    endif
End Go Procedure

procedure onArrival(MobilityEvent ev)
Begin
if State= END then
    RESULT = bBoard.operation (delete, host)
    Print ( Partial Result)
    aglet.dispose ()
    if ( RESULT=0 ) /* black board is empty */
        Print ( Search Time )
    else if State  $\neq$  END
Begin
    Perform the local search ()
    Neighbors = getLocalNeighbors (Router)
    if ( Neighbors.contains ( initial_host ) )
        Neighbors.remove (initial_host)
    if (Neighbors.contains ( immediateParent ) )
        Neighbors.remove (immediateParent)
    if ( Neighbors are empty ) /* become empty after removing */
        State = END
        DISPATCH (home)
    else

```

```

    for i =0 to Neighbors.size ()
        if (! Whiteboard.isVisited (Neighbors(i))) /* host not been visited */
            Whiteboard.setVisited(Neighbors(i))
        else
            Neighbors.removeElementAt (i)
    end for
    if (All children been visited)
        State = END
        aglet.Dispatch ( aglet.home) /* go home */
    Go (Neighbors)
End On_Arrival

```

4.3.5 Remote Flooding Algorithm

```

procedure onArrival(MobilityEvent ev)
if State= END then
    print ( aglet.partial result)
    if ( RESULT=0 ) /* black board is empty */
        aglet.dispose ()
        Print ( search time )
    else if State  $\neq$  END
    Begin
        Perform the search ()
        neighbors = getLocalNeighbors (Router)
        if ( Neighbors.contains ( initial_host ) )
            Neighbors.remove (initial_host)
        if (Neighbors.contains ( immediateParent ) )

```

```

        Neighbors.remove (immediateParent)
    if ( Neighbors are empty ) /* become empty after removing */
        State = END
        DISPATCH (home)
    else
        for i =0 ; to Neighbors.size ( )
            if (! Whiteboard.isVisited (Neighbors(i))) /* host not been visited */
                Whiteboard.setVisited(Neighbors(i))
            else
                Neighbors.removeElementAt ( i)
        enf for
        if ( All children been visited )
            State = END
            aglet.Dispatch ( aglet.home) /* go home */
        Go ( Neighbors )
    End On_Arriaval

```

```

procedure Go (destinations)
Begin
    for k = 0 to dest.size
        begin
            nodes = nodes + dest.elementAt (k)
        endfor
        bBoard.operation (register, nodes)
        for j=0 to dest.size -1
            begin

```

```

        aHost = dest.elementAt(j)
        cloneProxy = aglet.clone ()
        cloneProxy.Dispatch ( aHost )
    endfor
    aglet.Dispatch (aHost)
    if ( Dispatch failed )
        neighbors = getGlobalNeighbours (host )
        Go (neighbors)
    endif
End Go procedure

```

```

procedure onDispatching ( MobilityEvent ev )
Begin
    if ( STATE = END )
        RESULT = bBoard.operation ( delete , host )
    endif
End onDispatching

```

4.3.6 Regular Flooding Algorithm

```

procedure Go (destinations)
Begin
    for j=0 to dest.size -1
    begin
        aHost = destinations.elementAt(j)
        AgletProxy cloneProxy = (AgletProxy) aglet.clone ()

```



```

        cloneProxy.getAglet().stack.push ( host )
        cloneProxy.Dispatch ( host )
    endfor
    aglet.stack.push ( originHost )
    aglet.Dispatch (originHost )
    if( Dispatch failed )
        Neighbors = getGlobalNeighbors (host )
        Go( Neighbors )
End

procedure onArrival(MobilityEvent ev)
Begin
    if State = END then
        readLog= log.readPartialResult ()
        num_child = readLog.nextToken ()
        previous_result = redLog.nextToken ()
    end if
    num_child = num_child -1
    log.writePartialResult (num_child ,previous_result + partialresult , contact_log_time )
    if ( ! last_child )
        aglet.dispose ()
    else if (last_child )
        aglet.partialresult = previous_result + aglet.partialresult
        if ( ! Initiator )
            nextChild = (String)aglet.stack.pop()
            Dispatch ( nextChild )

```

```

    else
        print ( search result)
        print ( total search time )
    if State  $\neq$  END
Begin
    Perform the search ()
    neighbors = getLocalNeighbors (Router)
    if ( Neighbors.contains ( initial_host ) )
        Neighbors.remove (initial_host)
    if (Neighbors.contains ( immediateParent ) )
        Neighbors.remove (immediateParent)
    if (Neighbors = null OR Neighbors is empty) /* become empty after removing */
        State = END
    if ( ! aglet.stack.isEmpty () ) /* get the parent and dispatch to it */
        hostName = aglet.stack.pop()
        log.writeTime ( contactTime)
        getAgletProxy ().Dispatch (hostName)
    else
        for i =0 to Neighbors.size
            /* talking to Whiteboard(host not been visited */
            if ( ! wboard.getAgentID ( aglet.id ) )
                wboard.setAgentID ( aglet.id )
            else
                Neighbors.removeElementAt (i)
        end for
        if ( all children been visted )
            State = END

```

```
        aglet.Dispatch (home) ) /* go home */  
log.writeNumberOfNeighbors ( Neighbors.size () )  
Go ( Neighbors )  
End
```

Chapter 5

Implementation Issues and Experimental Results

5.1 Introduction

In this chapter, we will discuss the implementation of the prototypes based on the algorithms described in the previous chapters. The strategy followed for testing the prototypes consists of a set of test cases aimed to evaluate the impact of certain variables on the overall performance. The prototype for the mobile agent search is tested under a variable network size, network topology and number of agents. The objective is to obtain sufficient information to compare the performance between the AGLET mobile agent and the TACOMA mobile agent. In the experiments, the agents are injected from any node of the network to perform a distributed search. The launching application or initiator measures the time in milliseconds before injecting the first agent and after the last agent has arrived. The difference between these two time measurements gives the execution time for the entire search. All the experiments were carried out in the absence of simulated failure and under normal circumstances.

The computers were used simultaneously with other users and applications and no special care was taken to guarantee an exclusive access to computer resources or network during these experiments. A computer listing of the execution time for each experiment can be found in the Appendices.

5.2 Test Environment

All the experiments presented herein were developed in the Graduate Lab of the School of Computer Science on 1000 MHz Pentium with 500 MB RAM machines. The agent platform used was the AGLET-2.0.1 compiled and run on the Sigma Network (Linux Mandrake 7.1 machines) and the Ultra Network (SunOs 5.7 Generic_106541-06 sun4u sparc SUNW, Ultra5-10). The *Blackboard* associated with each node was implemented as a multi-threaded application using the Java JDK1.2.2. the *Router* and *Whiteboard* were implemented in Java JDK1.2.2 and used a RMI server. The size of information carried by the agent was the same for the all algorithms. The agent performed a dummy search at each site consisting of opening a file and reading the content. Since all the tests were carried out in an actual network environment, the limitations on the number of machines involved in the experiments were determined by the number of nodes available at the Graduate Lab of the School of Computer Science, which is 28 nodes.

For a single agent search similarly to [26], we apply the idea of [19] and [5] to our logical spanning tree in order to implement the traversal algorithm. It is important to note that since we are running our experiment on the LAN, the effect of the latencies between nodes is negligible. We also take advantage of the caching mechanism

provided by the AGLET. Using the caching mechanism, the agents do not have to transfer classes every time they move to the server side, they transfer it once and keep it at the server side for subsequent use. The tests are conducted with security checks turned off.

5.3 Test Results

In this section, we present the tests and their results. In order to enhance the analysis of the tests, we have measured the *Router*, *Blackboard* and *Whiteboard* contact time as well. We also measured search time without contacting time and we show the percentage of search time resulting from contacting other components. A complete listing for the search time and overhead time for each test can be found in the Appendices. We run each test thirty times and determined the average search time and variance of each test.

5.3.1 Variable Network Size

In this test, we compared the search time of running single and multiple agents on binary trees of different sizes. We start our testing with three nodes. In subsequent tests, we increased the size of the tree by two nodes at each step. We continue increasing the size of the tree until it reached 28 nodes. We obtained the following test results:

1. As the number of nodes in network increases, the total time of search also increases for both systems, both in single and multiple agent searches. The result is as expected, since it is clear that once the network size increases, we have more nodes to visit, hence the total time of the search increases.

2. The AGLET agent performs better than the TACOMA Agent. Searching with a single AGLET agent is almost ten times faster than searching with a single TACOMA agent. In multiple agent search, AGLET agent performed almost seven and a half times better than the TACOMA agent. This difference is due to the different architectures mentioned in the previous chapters.
3. Single agents of both the AGLET and the TACOMA platforms perform better than multiple agents for a small size network. Conversely, multiple agents perform better than single agents when the network size becomes large. The only difference here is in the size of the network in which the multiple agent's performance overcomes that of the single agent. In the TACOMA platform, the single agent performs better than multiple agent for a network size up to 18 nodes. However, in the AGLET platform, a network size of only 11 nodes is sufficient for the multiple agents to outperform a single agent. This result suggests that we can run multiple AGLET agents on a binary tree (i.e. take advantage of parallelism) on networks with a size of as little as eight nodes, while we cannot do that with TACOMA agents. In order to take advantage of parallelism with TACOMA agents, we should have a network (i.e. binary tree) size of at least eighteen nodes.

The last point of result shows that the single agent search performance is better than the multiple agents for a small network size. This result can be understood in the following ways:

- The search with multiple agents requires registering and removing the agent ID with the *Blackboard* for termination, and contacting the *Whiteboard* to avoid multiple visits to the same host. This extra work wastes some time and increases the overall search time.

- In a single agent search, hosts are visited sequentially. Hence, there is no need for having or contacting the *Blackboard* as it is the case with the multiple agent search. Also, there is only one agent traversing the entire structure of the network, where it only visits each node once. There is no need for contacting the *Whiteboard* to check prior visits. In other words, with the single agent search we avoid the overhead time associated with multiple agents.

The above explanation leads us to conclude that for small network sizes the advantages we gain from parallelism by using multiple agents are not enough to overcome the overhead associated with it. In a network of large size with multiple agents traversing at the same time, the advantages of parallelism overcome the above mentioned overhead and as a result we get better search performance with multiple agents than single agent search.

Another aspect of the third point of the result is that the AGLET system requires a smaller network size than the TACOMA system for multiple agent searches to perform better than the single agent search. This is an expected result since from the second point we noted that the AGLET agent performs much better than the TACOMA agent. Hence, it could overcome the overhead associated with multiple agent much sooner than the TACOMA agent.

The average execution time for a single and multiple traveling agent for both the AGLET and TACOMA System are shown in Figure 5.1 and Figure 5.2 and Tables in (Appendix A, B,C, D, E and F) and the comparison of AGLET Performance Vs. TACOMA Performance is shown in 5.3. For further performance investigation return to Table 5.1. The variance percentage for all tests in different search algorithm is $\leq$ %4 of the total search time.

Figure 5.1: Average Execution Time for Single and Multiple AGELT Agent

Figure 5.2: Average Execution Time for Single and Multiple TACOMA Agent

Figure 5.3: AGLET VS. TACOMA with Variable Network Sizes

5.3.2 Variable Number of Neighbors

In this test, we compare the results of running a single and a multiple agent of both TACOMA and AGLET Systems on various trees with a constant size. The trees are different in the number of neighbors. We start with a binary tree and increase the number of neighbors for each node at each step by two nodes until we end up with a star structure.

The test result shows that in the case of the single agent search, the changing of the number of neighbors, while keeping tree size constant, does not have serious effect on search time in both the systems. This result is predictable such that, by increasing the number of neighbors in single agent search, the change occurs only in the visiting sequence of hosts. This change in the visiting sequence does not have any effect on the search.

Aglet				Tacoma		
Network Size	Flooding	DFS	BFS	Flooding	DFS	BFS
3	389.70	289.67	289.03	2157.17	1266.70	1261.47
9	999.60	932.30	899.43	4937.07	3806.90	3828.83
15	1128.73	1446.63	1441.40	7410.33	6669.60	6878.10
21	1319.57	1901.53	1911.50	10117	10925.90	10990.10
27	1385.03	2102.00	2051.60	11618.63	19456.50	20115.20

Table 5.1: Performance of Single and Multiple AGLET and TACOMA Agents on Different Networks

For both systems, the multiple agent searches performed better than the single agent search when the number of neighbors was low. As the number of neighbors increases and the search becomes closer to a sequential search, the single agent search performs better than the multiple agent due to the overhead associated with the multiple agent search.

As such, the only difference between the AGLET and the TACOMA agents is at the turning point when the single agent search starts to become more efficient than the multiple agent search. In the TACOMA system, the multiple agent search performs better than the single agent search only when the structure is a binary tree. Alternatively, in the AGLET system, the multiple agent search performs better than a single agent search in a tree with up to an eight node neighbor ($k=8$) and the best performance comes from a tree where each node has four neighbors ($k=4$). This difference in the turning point is due to better performance of the AGLET agent than that of the TACOMA agent. The AGLET agent could handle and overcome the overhead associated with the multiple agents more efficiently.

The result of this test is consistent with the previous test. In both cases, the multiple agent search performs better than a single agent search when we have a binary tree

with twenty nodes. Both tests show that the AGLET agent is more efficient than the TACOMA agent.

Figure 5.4, Figure 5.5 and the tables in Appendix G show the average execution time for a single and a multiple traveling agent for both the AGLET and the TACOMA System once the size of the network is constant ($N = 20$) and the structure of nodes starts with a binary tree and ends with a star ($k = 2, 4, 6 \dots 19$). Figure 5.6 shows the AGLET Performance Vs. TACOMA Performance for this. For further performance investigation refer to Table 5.2. The variance percentage for all tests in different search algorithm is $\leq 5\%$ of the total search time.

Figure 5.4: Single and Multiple Mobile Agents with Variable Number of Neighbors (AGLET)

5.3.3 Different Topologies

In this test, we compare the results of running a single and multiple agents of both the TACOMA and AGLET systems on various topologies with a constant network size. The network for this test consists of sixteen nodes. The topologies used for this

Figure 5.5: Single and Multiple Mobile Agents with Variable Number of Neighbors (TACOMA)

Figure 5.6: AGLET Vs. TACOMA with Variable Number of Neighbors

Aglet			Tacoma	
Network Size	Flooding	BFS	Flooding	BFS
2	1358.63	1956.52	8313.10	8445.30
4	1188.57	1778.20	10167.27	8513.53
6	1695.17	1887.11	10542.77	8344.93
8	2112.70	1887.31	11225.33	8647.93
10	2412.87	1768.97	11350.17	8653.63
12	2495.53	1781.86	11674.93	9002.60
14	2422.77	1806.46	11678.07	9053.00
16	2480.20	1817.74	11570.67	11751.57
19	2494.57	1864.76	12029.27	11736.37

Table 5.2: Single and Multiple AGLET and TACOMA Agent Performances with Variable Number of Neighbors

test are the unidirectional ring, the bidirectional ring, the hypercube, and the binary tree. There were topologies considered by [26]; so we chose them to compare our results with that of [26]. However, it is possible to choose other topologies such as mesh, other variant of ring, etc. Besides comparing the search results of the AGLET and the TACOMA, we also compare the search results of a single and multiple agent for the AGLET system as well.

The following results from the test were obtained:

1. Single Agent Traversal:
 - a. The best performance occurs with a bidirectional ring and the worse performance occurs with a hypercube for both the AGLET and the TACOMA systems. Hence, the behavior of both systems is *similar* on different topologies when a *single agent* is used.
 - b. In the best and the worst scenarios as mentioned earlier, the performance of the agent on all of the topologies is almost the same for both systems.

This result is not surprising since traversing the network with the single agent

is only matter of finding the next set of nodes, which should be added to the list of nodes that the agent already carries and should be visited sequentially. Finding the next set of nodes requires contacting the *Router*. We contact the *Router* a similar number of times for all topologies, hence we have a similar performance of all topologies.

2. Multiple Agent Traversal:

- a. The *best* performance of the TACOMA agent occurs with the bidirectional ring, where it is similar to the performance of a binary tree. On the other hand, the best performance for the AGLET agent occurs with a hypercube. The *differences* in the best suitable topology for both the AGLET and TACOMA come from the efficiency with which they can overcome the overhead associated with the flooding algorithm. It is shown from the first test that the AGLET agents are more efficient than the TACOMA agents. Hence, the AGLET agents are more capable of taking advantage of parallelism. The second test shows that the best performance of the AGLET agents comes from a *tree with four neighbors* for each node or hypercube.
- b. The *worse case* for the AGLET agent comes from the unidirectional ring while the worse case of the TACOMA system comes from hypercube. This result is expected and we can explained as follows::
 - Even though the best performance of the AGLET agent comes from deploying multiple agents, deploying multiple AGLET agents on a unidirectional ring is irrelevant. It adds a lot of unnecessary overhead to the timing of the search result associated with contacting other components.
 - It has been shown in the first and the second tests that, TACOMA's

best performance in networks of twenty nodes or less comes with a single agent or with multiple agents in a binary tree structure. Hence, deploying multiple agents on hypercube structure with a size of 16 nodes, does not add any efficiency to the search result. In contrast, it adds more overhead compared to running a single agent. This explains why the worse case with the TACOMA case is with a hypercube and not a unidirectional ring.

each of the systems. As shown in the second test, the AGLET agent performs the best when we deploy multiple agents on a tree with four neighbors for each node. In a similar manner, the TACOMA agent performs best with a binary tree topology.

3. Single and Multiple AGLET agent performance:

The result shows that the AGLET agent performs better with flooding rather than a single agent for a binary tree and a hypercube, and is almost same for the bidirectional ring. This result is consistent with the result of the first test which shows that the AGLET agent performs well with flooding when the size of the network exceeds 8 nodes. In the case of a bidirectional ring, the single agent performs similar or slightly better than flooding. This is because of the overhead associated with flooding. If we consider the extra overhead associated with flooding, the bidirectional ring performs better than the single agent search. For more information about this result refer to Appendix H (tables H1 and H4). The comparison of single and multiple agent performance is shown in Figure 5.13. Figures 5.7, 5.8, 5.9, 5.10, 5.11, 5.12 and tables in Appendix H show the test results for all cases. The variance percentage for all tests in different search algorithm is $\leq 8\%$ of the total search time.

Figure 5.7: Execution Time of Single Agents on Different Topologies (AGLET)

Figure 5.8: Execution Time of Single Agents on Different Topologies (TACOMA)

Figure 5.9: AGLET VS. TACOMA with Different Topology (Single Agent)

Figure 5.10: Execution Time for Multiple Agents on Different Topologies (AGLET)

Figure 5.11: Execution Time for Multiple Agents on Different Topologies (TACOMA)

Figure 5.12: AGLET Vs. TACOMA with Different Topologies(Flooding)

Figure 5.13: Single vs. Multiple with Different Topologies (AGLET)

5.3.4 Three Different Flooding Algorithm

In this test, we investigate the effects of three different implementations of the flooding algorithm on binary trees of different sizes. We start our tests on a binary tree with three nodes. In subsequent tests, we increase the size of the tree by two nodes at each step. We continue until the size of the tree has reached 28 nodes. The differences between different algorithms which have been investigated is in the way each algorithm *terminates*.

For the first and the second algorithms we have a *Blackboard* component as a separate component running on the initiator node. Each agent registers its ID with the *Blackboard* after the search starts and removes its ID from the *Blackboard* when the search is completed. The search terminates when all the agents remove their ID from the *Blackboard*. The difference between the first and the second algorithm is the locations where the agent tries to remove its ID from the *Blackboard*.

In the first approach, which we called *Local flooding*, the communication between the agent and the *Blackboard* is done locally at the initiator node. This contact happens just before the agent reports its partial result and removes its ID from the *Blackboard* after completing its search path and returning to the initiator.

In the second approach, which we called *Remote flooding*, the communication between the agent and the *Blackboard* is done remotely. Here, the agent contacts the *Blackboard* to remove its ID once it is at the end of the search path just before moving back to the initiator. In this way, the agent removes its ID while being hosted on a remote node. Only after its ID has been removed from the *Blackboard*, it returns to the initiator of the algorithm.

There is another difference in addition to the way in which *Remote* and the *Local flooding* terminate. Even though in both algorithms the agents have to return to the initiator node to report their partial results, they differ in the amount of time they stay at the initiator node. In *Remote flooding*, once an agent arrives at the initiator node, it stays for a very short time. It does not have to contact the *Blackboard* to remove its ID. It reports its partial result and dies. In the case of *Local flooding* however, the agent has to contact the *Blackboard* to remove its ID. Once its ID has been removed, it reports its partial result. This results in living for a longer period than the agents in *Remote flooding*. This situation has an effect on the number of threads the system has to create at a time, and ultimately it has an effect on the performance of the agents.

In the third approach, which we called *Regular flooding*, the agents start moving from the initiator node all the way down to the end of their search path. Each agent

reports its partial result to its immediate parent (i.e. the node they came from) when they reach the end of the search path. This continues until the final result reaches the initiator. The algorithm terminates after the last agent returns to the initiator. In this approach, we do not have any *Blackboard* component for the agent to contact for the termination as it was the case with the other two algorithms. At each node, we keep a *Log* (i.e. local *Blackboard*) for holding information about it. With this information, we keep track of the number of neighbors (or children) each node has, how many of those children reported back their partial results, and the partial result before the last child arrives. the following results were obtained from the test :

1. There is very little difference between all three approaches if the network size is small.
2. Once the network size exceeds nineteen nodes, *Remote flooding* performs better than the other two algorithms. This result looks strange at the first glance since we expect that communication on the same host (i.e local communication) with the *Blackboard* would perform better than communication on two different hosts (i.e remote communication) with the *Blackboard*, since the latest involves contacting *Blackboard* through the network, which is time consuming.

This occurs because once we have *Local flooding* there is almost always a possibility that several agents will complete their search path and return to the initiator at same time, or atleast before the other agents running on the initiator node, are able to complete their work. This creates a situation where we have several agents on the same node at one time. Each of them is executing independently of the other agents residing within the same place. Hence, the agent execution environment has to create new threads to run each of those agents independently each time one of them arrives. This operation is time consuming. The tests result shows that once the network size increases the cost

of creating a thread is more than the cost of contacting the *Blackboard* remotely. This explains why the *Remote flooding* performs better than the *Local flooding* when the network size exceeds 19 nodes. This is also an example of when the computation on the server side or waiting for a clogged CPU has a greater effect than the actual latency on the search time

3. The *Remote flooding* performs better than the *Regular flooding* even though we do not contact the *Blackboard* remotely or have multiple agents running at the same place at one time. The overhead time in this case comes from two other places.
 - a. As agents move from the top of the tree towards the leaves, we have to keep some information about each of the nodes in a *Log*. This information will be used later to check if the last child for a particular node has arrived before that node reports back to its parent that its search has been completed.
 - b. Once an agent completes its search, it returns to its parent or predecessor. At that moment, we have to check if the arrived agent is the last child of that node. If the arrived agent is not the last child, its partial results are written to the *Log* and the agent is destroyed. We continue to wait for the last agent to arrive. If the arrived agent is the last agent, it adds its partial result to the previous partial results, which are collected and saved on the *Log*. It then moves to a higher level of the tree. The same process continues until the last agent reports the final result to the initiator node. The test result shows that contacting the *Log* once an agent heads towards a child or leaf of the tree, plus collecting partial results from the arrived agents plus waiting for last agent to arrive at each node, is more expensive (i.e time consuming) than registering and removing the ID with the

Blackboard, which is the case with *Remote flooding*.

Figure 5.14 shows the execution time for all three algorithms. For further performance investigation return to table 5.3.

Figure 5.14: Execution of Remote Flooding, Local Flooding and Regular Flooding for Various Network Size

5.3.5 Platform Overhead

In this test, we compare the time required for moving a single agent between two nodes for both the AGLET and the TACOMA agents. In other words, comparing the time required to deploy an agent to a node and pulling back this agent to the source for both systems. The test results show that the AGLET mobile agent is almost ten times faster than the TACOMA agent. The difference in performance of the two systems is due to their differences in architecture. In the AGLET mobile agent, we compile the agent code to byte code. When we deploy an agent we serialize the byte code and send it over the socket to be executed on the remote place. While

Network size	Remote Flooding	Local Flooding	Regular Flooding
3	389.70	385.80	359.60
5	718.67	718.27	729.23
7	705.73	756.40	699.40
9	999.60	1022.70	1074.80
11	991.40	994.87	1037.93
13	1028.70	1051.33	1095.83
15	1128.73	1107.37	1056.97
17	1362.93	1288.00	1467.39
19	1310.48	1319.93	1402.10
21	1319.57	1342.27	1482.30
23	1250.60	1440.27	1446.33
25	1395.20	1512.53	1471.83
27	1385.03	1655.33	1513.97

Table 5.3: Performance Comparison of Three Flooding Algorithms

in the TACOMA system, when we deploy the agent we send a code. This code should be compiled and linked on the fly to be ready for execution on a remote place. This requires more time than having byte code executed on a remote site. For more information about TACOMA performance return to the [13]. The results of this test are presented in Appendix I.

Figure 5.15 shows the execution time of the single agent for both the AGLET and TACOMA. The variance percentage for all tests on AGLET platform is $\leq 1\%$ and for TACOMA platform is $\leq 9\%$ of execution time.

5.3.6 Cost of Overhead Associated with the Search

In our tests, we measure the time required for the agent to contact the stationary components *Blackboard*, *Whiteboard* and *Router*. Contact time, which is required by the agent to contact these components, is measured by taking the difference between the time before and after the contact. These measurements lead to a better understanding of the test results, agent's performance, and the possibility of how to improve

Figure 5.15: Execution Time for AGLET and TACOMA

the performance by eliminating or reducing the effect of some of the components.

Implementing stationary components as HTTP or RMI server creates some overhead time. For example once using RMI server, every time we invoke a function call on the server, it redirects the call to the actual object, which implements the functionality. Similarly, the reply from the object to the agent has to go through RMI server.

It is possible to use the *Aglet context* instead of HTTP and RMI server for implementing stationary components. Using Aglet context is not free of overhead either. For an agent to contact other agents running in different context requires knowledge about the agent's id and its location. Our choice of HTTP and RMI server is based

on trying a different approach than the one used by [26].

Comparing search time with and without overhead shows that although the overhead linked to using RMI and HTTP server has an effect on the total search time, it does not affect the search performance behavior. In other words, the time required for moving agents in the network (i.e. performing search) is independent of how the stationary components are implemented. This becomes clearer if the agent knows its itinerary ahead of time and does not require contacting any stationary components.

Figure 5.16 and 5.17 show the search time with and without overhead for the single agent and multiple agent search respectively. They also verify that the time required for moving agent in network is independent of how stationary components are implemented.

Figure 5.16: Execution Time of the Single Agent With and Without Overhead

Figure 5.17: Execution time of the Multiple Agents With and Without Overhead

Figure 5.18 shows the overhead time associated with contacting the *Router* for both the single and multiple agents search. From figure 5.18 we observed the following :

1. As the network size increases, the overhead resulted from contacting the *Router* also increases in both the single and multiple agent search. This result is not surprising since, as the network size becomes larger, there are more contacts to the *Router* components and consequently the overhead time increases as well.
2. The overhead associated with the single agent search is more than the multiple agent's search. This difference is due to the difference in the nature of the two searches. In the single agent search, the agent contacts the *Router* on each node of the network (i.e. number of contacts is n) during the search. It then computes the *Router* contact time at each node, adds the time to the previous *Router* contact time, carries the result and finally reports the result once the search terminates.

In the multiple agent search, since parallelism is involved several, contacts happen at the same time (depending on the number of children that each node has). Here, the *Router* contact time is the largest one. In order to find the largest contact time in the multiple agent's search, the agent contacts the *Log* component and writes its contact time into it to be used later for finding the largest *Router* contact time. Each contact to the *Router* component is actually a contact to the *Router* and the *Log* component. The number of times the agent contacts the *Router* is equal to two times the height of the tree (i.e. $2h$) if we assume that contacting the *Router* and *Log* component will take the same time. Once the Network size increases, the difference between the network size (n) and the height of the tree (h) also increases. Hence, the difference in overhead associated with contacting the *Router* in both the single and multiple agents' search increases as well.

Figure 5.18: The Overhead Time Produced from Contacting the Router for Single and Multiple Agents

5.4 Summary of Experimental Result

In all the tests we conducted, the single agent performs better than the multiple agents where the network size is small. For the larger network sizes, the multiple agents overcome the overhead associated with it and outperform the single mobile agent. All of the flooding algorithms have similar performances on the small sized networks. *Remote flooding* outperforms the *Local* and the *Regular flooding* on the large sized network. Among all the topologies we tested, the best performance was obtained from hypercube. Comparing the test results that we obtained from using the AGLET platform to similar tests performed on the TACOMA platform, it suggests that the overall behaviors of the mobile agents are platform independent in all test cases except the hypercube, whereas its performance is platform dependent.

Chapter 6

Conclusions

6.1 Thesis Summary

In this research, we have examined the performance of a mobile agent in a distributed search. Several research papers related to this topic and their solutions were surveyed. The agent's definition and some of its characteristics (i.e autonomous behavior, migration process, proactive and reactive behaviors) as well as some current mobile agent platforms have also been discussed.

We have presented three different distributed search algorithms which enable the designer to be more aware of the pros and cons associated with the distributed search using multiple mobile agents. It also helps designers to choose an algorithm and a platform that suits their application and leads to the best search performance.

We have presented a model to implement an agent-based distributed search as an alternative to the message passing approaches. The model is structurally similar to the model presented by [26], however, it differs in its implementation.

The model consists of several components, which makes navigation and coordination of the agents possible. The components include a *Blackboard*, *Whiteboard*, *Router* and *Log*.

Contrary to [26], *Whiteboard* and *Blackboard* components are not stationary agents, but are RMI and HTTP simple servers respectively. By implementing the *Router* and *Whiteboard* as a simple RMI server, the agent sends a message to the *Router* and *Whiteboard* every time it needs to contact those components.

Several experiments were carried out using the AGLET mobile agent platform for a single traveling agent as well as for multiple agents. In all the tests we conducted, the single agent performs better than multiple agents when the network size is small. For the larger network size, multiple agents overcome the overhead associated with it, and outperform the single mobile agent. The same result is true for the TACOMA platform. The only difference between the AGLET and the TACOMA is in the size of the network in which the multiple agent performance overcomes the single agent performance. In the TACOMA platform, the single agent performs better than the multiple agent for a network size of up to eighteen nodes. In the AGLET platform network, a size of only eight nodes is sufficient for the multiple agent to beat a single agent. We identify the crossover points for all of our tests. This could be used to switch between different structures for performance critical applications.

The AGLET agent performs better than the TACOMA Agent. Searching with a single AGLET agent is almost ten times faster than searching with a single TACOMA agent. In the multiple agent search, the AGLET agent performed almost seven and a half times better than the TACOMA's agent.

We found that changing the number of neighbors, while keeping tree size constant in a single agent search, does not have a serious effect on the search time for both systems. The effects can be seen by using the multiple agent search. For both systems, the multiple agent searches performed better than the single agent search when the number of neighbors were low. As the number of neighbors increases, the single agent performs better than the multiple agent search. The difference between the AGLET and the TACOMA agents occur at the turning point when multiple agent search becomes better than single agent search. Furthermore, it was also found that the best topology for the AGLET agent is the hypercube.

All three flooding algorithms implemented have similar performance on the small sized network. However, *Remote flooding* outperforms the *Local* and the *Regular flooding* on the large sized network.

A comparison of the overall test results obtained using the AGLET platform to similar tests performed on the TACOMA platform by [26] suggests that the behavior of the mobile agents are platform independent in all tests except the hypercube topology, whereas its performance is platform dependent.

6.2 Future Work

One place where there could be further examination is with the actual model which we used to implement a distributed search. For example, in our model we use the RMI server for implementing a *Whiteboard* and *Router* component. These components could be implemented using CORBA. Replacing the RMI server with the CORBA

server makes the *Whiteboard* and *Router* components independent of the agent's implementation language. A similar thing could be said about the *Blackboard* component which is a multi-threaded HTTP server. We could also implement the *Whiteboard*, *Router* and *Blackboard* using stationary agents instead of the RMI or multi-threaded HTTP server, and investigate the affects of these changes on the performance.

Different mobile agents could be used for the experiments or changes could made to the mobile agents we used for the experiment. In the case of the AGLET agent, this change could be in agent mobility. The AGLET agent migration could be changed from weak mobility to strong mobility and the effect of these changes on the performance of mobile agents could be examined.

The creation of a component responsible for sensing the network connections, CPU loads, and dynamically reassigning the local and global routing tables is a desirable improvement. Further modifications might explore the flooding algorithm to more efficiently broadcast the changes on the global routing tables and creating the new tables dynamically.

The combination of a mobile agent and a message passing to perform the distributed search could be considered as a viable improvement of the distributed paradigm. For example, in places where an agent has to return back to the initiator node to report back the total or partial result, it could just send a message with a result to the initiator node and terminate itself on the remote nodes. This replaces the transferring of the agent code with a simple message call improving the overall performance if the size of the transferable data is small relative to the size of the agent.

In our experiment, we focused on the search algorithm rather than the search criteria. The agent interacts with local resources at each node and carries the content of a predefined file. The focus of searching the algorithm could be changed to put on the search criteria, and then examine the effect of this change on the performances.

When we studied the effect of the topology on the search time, we found that the best topology for the AGLET agent is hypercube for a network of 16 nodes. We also found that a tree with four neighbors for each node is the best topology for a network of 20 nodes. This test suggests that there is some relation between hypercube and the tree of four neighbors. Further tests could be done on the bigger network size to determine the best topology for the AGLET agents.

In cases where we have to search the network more than once and changes in the network does not occur very often, we can save the path we obtained in prior search and reuse it in subsequent searches. This would allow us to eliminate contacting the *Router* and consequently give a better performance.

The search time can be affected considerably by the size of an agent. This is especially true if the agent size is a substantial percentage of total data which have to be transferred over the network. Further study to determine the accepted size of an agent is required in the distributed search.

Another area of research is to study the effect of a query size on the search time using mobile agents. The research can cover the effect of the request and/or the result of the query.

In our study we focused on the search algorithm and traversed the entire network. In real life, we do not have to traverse the entire network. Studying the effect of number of trips performed by the agent in order to find a key target is a viable research area.

Bibliography

- [1] J. Baek, G. Kim and H. Yeom. Cost-Effective Planning of Timed Mobile Agents. *Proceedings of the 5th international conference on Autonomous agents*, Pages 120-121, 2001.
- [2] J. Baek, J. Yeo, G. Kim, and H. Yeom. Cost Effective Mobile Agent Planning for Distributed Information Retrieval. In *Proceedings of 21st Conference on Distributed Systems*, pages 65-72, 2001.
- [3] J. Baek, J. Yeo and H. Yeom. Agent Chaining: An Approach to Dynamic Mobile Agent Planning. *Proceedings 22nd International Conference on Distributed Computing Systems*, pages 579-589, 2002.
- [4] L. F. Bic, M. Fukuda, M.B.Dillencourt. Distributed Computing using Autonomous Objects. In *IEEE Computer*, 29(8):55-61, 1996.
- [5] B. Brewington, R. Gray and K. Moizumi, D. Kotz, G. Cybenko and D. Rus. Mobile agents in distributed information retrieval. In *Intelligent Information*

Agents, pages 355-395, 1999.

- [6] A. Castillo, M. Kawaguchi, N. Paciorek, and D. Wong. Concordiatm as Enabling Technology for Cooperative Information Gathering. In *Proceedings of the Japanese Society for Artificial Intelligence Conference*, pages 280-283 1998.
- [7] R. S. Gray, D. Kotz, R. A. Peterson, J. Barton, D. Chacn, P. Gerken, M. Hofmann, J. Bradshaw, M. Breedy, R. Jeffers, and N. Suri. Mobile-Agent versus Client/Server Performance: Scalability in an Information-Retrieval Task. In *Proceedings 5th International Conference on Mobile Agents*, pages 229-243 , 2002.
- [8] Robert S. Gray, George Cybenko, David Kotz, Ronald A. Peterson and Daniela Rus Thaye. D'Agents: Applications and Performance of a Mobile-Agent System. In *Software Practice& Experience*, 32(6):543-573, 2002.
- [9] S. Green, L. Hurst, B. Nangle, P. Cunningham, F. Somers and R. Evans. Software Agents: A review. http://www.cs.tcd.ie/research_groups/aig/iag/toplevel2.html, 1997.
- [10] L. Ismail and D. Hagimont. A Performance Evaluation of the Mobile Agent Paradigm. *Conference on Object-Oriented*, pages 306-313, 1999.
- [11] R. Jain, F. Anjum, A. Umar. A Comparison of mobile agent and client-server paradigms for information retrieval tasks in virtual enterprises. In *IEEE*

- Academia/Industry Working Conference on Research Challenges* , pages 209-215, 2000.
- [12] R. Jha, and S. Iyer Performance Evaluation of Mobile Agents for E-Commerce Application. In *Proceedings International Conference on High Performance Computing*, pages 331-341, 2001.
- [13] D. Johansen, N. P. Sudmann and R. v. Renesse. Performance Issues in TACOMA. In *Proceedings of the 3rd Workshop on Mobile Object Systems, 11th European Conference on Object-Oriented Programming*, 1997
- [14] D. Johansen, R. V. Renesse and F. Schneider. An Introduction to the TACOMA Distributed System Version 1.0. *Technical Report 95-23, Department of Computer Science, University of Troms, Norway*, June 1995
- [15] D. Lange and M. Oshima. Mobile Agents with Java, The Aglet API. In *World Wide Web*, 1(3):111-121, 1998
- [16] D. B. Lange and M. Oshima. Seven good reasons for mobile agents. In *Communications of the ACM*, 42(3): 88-89, 1999
- [17] D. B. Lange and M. Oshima. Programming and Deploying Java Mobile Agents with Aglets. In *Addison Wesley publication*, 2nd, 1998.

- [18] Q. H. Mahmoud Distributed Programing with Java. *Manning publication*, 2000.
- [19] K. Moizumi, and G. Cybenko. The Traveling Agent Problem. *Mathematics of Control, Signals and Systems*, pages 213-232, 1998.
- [20] R. Morawek. Threshold Route Optimization Algorithm for Information Retrieving Mobile Agents. In *Proceedings 6th International Workshop on Cooperative Information Agents*, pages 312-320, 2002.
- [21] A. Puliafito, S. Riccobene, M. Scarpa. Which paradigm should I use? An analytical comparison of the client-server, remote evaluation and mobile agent paradigms. *Concurrency and Computation: Practice and Experience* 13(1): 71-94, 2001 .
- [22] C. Raibulet and C. Demartini. Mobile Agent Technology for the Management of Distributed Systems - a case study. In *Computer Networkes*, 34(6):823-830, 2000.
- [23] M.G. Rubinstein, O. Carlos, M. B. Duarte and G. Pujolle. Scalability of a Network Management Application Based on Mobile Agents In *Proceedings Second International IFIP-TC6 Networking Conference*, pages 515-526, 2002 .
- [24] L. Silva, G. Soares, P. Martins, V. Batista and L. Santos. Comparing the Performance of Mobile Agent Systems: A Study of Benchmarking. *First International Symposium on Agent Systems and Applications Third International Symposium*

on Mobile Agents, pages 270-272, 1999.

- [25] M. Straer, M. Schwehm. A Performance Model for Mobile Agent Systems. In *Proceedings Int. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA '97)*, Vol II, pages 1132-1140, 1997.

- [26] E. Velazquez. Mobile Agents In Distributed Search: A Comparative Study. *Master thesis, M.cs, School of Computer Science, Carleton University*, 2001

- [27] D. Wong, N. Paciorek, T. Walsh, J. DiCelie, M. Young, and B. Peet. Concordia: An infrastructure for collaborating mobile agents. In *Proceedings of the First International Workshop on Mobile Agents*, pages 86-97, 1997.

Appendix A

Remote Flooding With Variable Network Size

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	368.00	12.00	7.00	76.00	95.00	0.26	273.00
2	341.00	13.00	6.00	75.00	94.00	0.28	247.00
3	374.00	12.00	7.00	75.00	94.00	0.25	280.00
4	387.00	12.00	6.00	120.00	138.00	0.36	249.00
5	405.00	12.00	6.00	92.00	110.00	0.27	295.00
6	390.00	13.00	8.00	120.00	141.00	0.36	249.00
7	367.00	13.00	6.00	103.00	122.00	0.33	245.00
8	417.00	13.00	6.00	128.00	147.00	0.35	270.00
9	431.00	13.00	7.00	118.00	138.00	0.32	293.00
10	398.00	12.00	7.00	101.00	120.00	0.30	278.00
11	450.00	13.00	6.00	125.00	144.00	0.32	306.00
12	364.00	13.00	6.00	92.00	111.00	0.30	253.00
13	377.00	12.00	6.00	67.00	85.00	0.23	292.00
14	401.00	16.00	6.00	131.00	153.00	0.38	248.00
15	369.00	13.00	9.00	67.00	89.00	0.24	280.00
16	406.00	13.00	6.00	69.00	88.00	0.22	318.00
17	419.00	15.00	6.00	117.00	138.00	0.33	281.00
18	403.00	12.00	9.00	68.00	89.00	0.22	314.00
19	426.00	13.00	6.00	111.00	130.00	0.31	296.00
20	414.00	13.00	6.00	121.00	140.00	0.34	274.00
21	389.00	13.00	7.00	81.00	101.00	0.26	288.00
22	391.00	14.00	6.00	81.00	101.00	0.26	290.00
23	385.00	13.00	6.00	78.00	97.00	0.25	288.00
24	412.00	13.00	6.00	112.00	131.00	0.32	281.00
25	438.00	13.00	7.00	113.00	133.00	0.30	305.00
26	344.00	13.00	6.00	77.00	96.00	0.28	248.00
27	308.00	12.00	7.00	94.00	113.00	0.37	195.00
28	373.00	12.00	7.00	76.00	95.00	0.25	278.00
29	397.00	14.00	6.00	85.00	105.00	0.26	292.00
30	347.00	13.00	8.00	72.00	93.00	0.27	254.00
Average	389.70	12.93	6.60	94.83	114.37	0.29	275.33

Table A.1: Remote Flooding (N =3)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	674.00	19.00	15.00	151.00	185.00	0.27	489.00
2	681.00	19.00	15.00	195.00	229.00	0.34	452.00
3	693.00	23.00	14.00	219.00	256.00	0.37	437.00
4	745.00	24.00	16.00	190.00	230.00	0.31	515.00
5	736.00	25.00	14.00	184.00	223.00	0.30	513.00
6	667.00	19.00	16.00	183.00	218.00	0.33	449.00
7	680.00	25.00	15.00	179.00	219.00	0.32	461.00
8	690.00	24.00	15.00	183.00	222.00	0.32	468.00
9	751.00	19.00	17.00	235.00	271.00	0.36	480.00
10	720.00	26.00	16.00	166.00	208.00	0.29	512.00
11	779.00	24.00	15.00	223.00	262.00	0.34	517.00
12	773.00	24.00	16.00	216.00	256.00	0.33	517.00
13	700.00	20.00	15.00	165.00	200.00	0.29	500.00
14	687.00	18.00	16.00	149.00	183.00	0.27	504.00
15	725.00	18.00	29.00	210.00	257.00	0.35	468.00
16	757.00	23.00	15.00	218.00	256.00	0.34	501.00
17	713.00	25.00	16.00	210.00	251.00	0.35	462.00
18	688.00	23.00	17.00	185.00	225.00	0.33	463.00
19	677.00	20.00	14.00	182.00	216.00	0.32	461.00
20	682.00	18.00	15.00	151.00	184.00	0.27	498.00
21	676.00	23.00	15.00	178.00	216.00	0.32	460.00
22	872.00	19.00	18.00	174.00	211.00	0.24	661.00
23	723.00	19.00	16.00	220.00	255.00	0.35	468.00
24	711.00	19.00	14.00	175.00	208.00	0.29	503.00
25	750.00	20.00	21.00	167.00	208.00	0.28	542.00
26	750.00	25.00	16.00	200.00	241.00	0.32	509.00
27	672.00	19.00	15.00	215.00	249.00	0.37	423.00
28	734.00	25.00	19.00	206.00	250.00	0.34	484.00
29	737.00	25.00	13.00	208.00	246.00	0.33	491.00
30	717.00	23.00	15.00	218.00	256.00	0.36	461.00
Average	718.67	21.77	16.10	191.83	229.70	0.32	488.97

Table A.2: Remote Flooding(N =5)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	689.00	19.00	50.00	163.00	232.00	0.34	457.00
2	706.00	24.00	22.00	218.00	264.00	0.37	442.00
3	810.00	24.00	24.00	249.00	297.00	0.37	513.00
4	788.00	24.00	23.00	214.00	261.00	0.33	527.00
5	710.00	27.00	36.00	162.00	225.00	0.32	485.00
6	714.00	19.00	22.00	204.00	245.00	0.34	469.00
7	728.00	26.00	30.00	174.00	230.00	0.32	498.00
8	1045.00	53.00	22.00	470.00	545.00	0.52	500.00
9	676.00	19.00	21.00	141.00	181.00	0.27	495.00
10	676.00	19.00	24.00	146.00	189.00	0.28	487.00
11	625.00	25.00	21.00	136.00	182.00	0.29	443.00
12	634.00	25.00	26.00	156.00	207.00	0.33	427.00
13	605.00	19.00	23.00	141.00	183.00	0.30	422.00
14	652.00	24.00	23.00	152.00	199.00	0.31	453.00
15	675.00	18.00	25.00	205.00	248.00	0.37	427.00
16	655.00	20.00	27.00	138.00	185.00	0.28	470.00
17	739.00	20.00	22.00	229.00	271.00	0.37	468.00
18	662.00	19.00	23.00	221.00	263.00	0.40	399.00
19	721.00	25.00	21.00	230.00	276.00	0.38	445.00
20	744.00	25.00	24.00	208.00	257.00	0.35	487.00
21	656.00	20.00	22.00	180.00	222.00	0.34	434.00
22	621.00	23.00	24.00	148.00	195.00	0.31	426.00
23	633.00	23.00	20.00	171.00	214.00	0.34	419.00
24	699.00	22.00	25.00	152.00	199.00	0.28	500.00
25	681.00	25.00	20.00	163.00	208.00	0.31	473.00
26	667.00	24.00	21.00	160.00	205.00	0.31	462.00
27	752.00	19.00	22.00	191.00	232.00	0.31	520.00
28	707.00	25.00	27.00	199.00	251.00	0.36	456.00
29	761.00	24.00	21.00	199.00	244.00	0.32	517.00
30	741.00	26.00	25.00	194.00	245.00	0.33	496.00
Average	705.73	23.50	24.53	190.47	238.50	0.34	467.23

Table A.3: Remote Flooding(N =7)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	955.00	57.00	23.00	193.00	273.00	0.29	682.00
2	913.00	36.00	31.00	192.00	259.00	0.28	654.00
3	965.00	32.00	23.00	255.00	310.00	0.32	655.00
4	896.00	30.00	26.00	190.00	246.00	0.27	650.00
5	904.00	31.00	29.00	203.00	263.00	0.29	641.00
6	872.00	30.00	23.00	236.00	289.00	0.33	583.00
7	916.00	32.00	25.00	229.00	286.00	0.31	630.00
8	913.00	32.00	28.00	223.00	283.00	0.31	630.00
9	970.00	25.00	24.00	275.00	324.00	0.33	646.00
10	957.00	29.00	30.00	257.00	316.00	0.33	641.00
11	1015.00	25.00	25.00	307.00	357.00	0.35	658.00
12	951.00	32.00	36.00	247.00	315.00	0.33	636.00
13	1059.0	31.00	27.00	262.00	320.00	0.30	739.00
14	1005.00	31.00	26.00	276.00	333.00	0.33	672.00
15	1106.00	31.00	25.00	305.00	361.00	0.33	745.00
16	1011.0	26.00	26.00	243.00	295.00	0.29	716.00
17	902.00	26.00	28.00	237.00	291.00	0.32	611.00
18	968.00	30.00	24.00	263.00	317.00	0.33	651.00
19	1025.00	25.00	29.00	302.00	356.00	0.35	669.00
20	1097.00	27.00	26.00	310.00	363.00	0.33	734.00
21	1109.00	25.00	25.00	290.00	340.00	0.31	769.00
22	1126.00	31.00	27.00	344.00	402.00	0.36	724.00
23	1193.00	31.00	26.00	329.00	386.00	0.32	807.00
24	961.00	24.00	29.00	258.00	311.00	0.32	650.00
25	1018.00	29.00	25.00	249.00	303.00	0.30	715.00
26	1085.00	26.00	26.00	324.00	376.00	0.35	709.00
27	1001.00	26.00	28.00	315.00	369.00	0.37	632.00
28	1082.0	31.00	29.00	322.00	382.00	0.35	700.00
29	1080.00	31.00	43.00	325.00	399.00	0.37	681.00
30	933.00	29.00	28.00	275.00	332.00	0.36	601.00
Average	999.60	30.03	27.33	267.87	325.23	0.32	674.37

Table A.4: Remote Flooding(N =9)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	979.00	31.00	29.00	208.00	268.00	0.27	711.00
2	962.00	34.00	34.00	245.00	313.00	0.33	649.00
3	991.00	30.00	27.00	220.00	277.00	0.28	714.00
4	847.00	31.00	31.00	225.00	287.00	0.34	560.00
5	902.00	33.00	30.00	224.00	287.00	0.32	615.00
6	930.00	38.00	27.00	235.00	300.00	0.32	630.00
7	873.00	26.00	36.00	228.00	290.00	0.33	583.00
8	960.00	29.00	28.00	232.00	289.00	0.30	671.00
9	943.00	31.00	40.00	245.00	316.00	0.34	627.00
10	1045.00	29.00	30.00	288.00	347.00	0.33	698.00
11	1201.00	38.00	40.00	270.00	348.00	0.29	853.00
12	894.00	31.00	31.00	241.00	303.00	0.34	591.00
13	956.00	30.00	41.00	260.00	331.00	0.35	625.00
14	928.00	28.00	29.00	232.00	289.00	0.31	639.00
15	1009.00	25.00	28.00	298.00	351.00	0.35	658.00
16	1025.00	30.00	32.00	267.00	329.00	0.32	696.00
17	1048.00	31.00	28.00	315.00	374.00	0.36	674.00
18	922.00	26.00	31.00	222.00	279.00	0.30	643.00
19	1032.00	27.00	30.00	246.00	303.00	0.29	729.00
20	1065.00	26.00	32.00	314.00	372.00	0.35	693.00
21	1115.00	31.00	41.00	306.00	378.00	0.34	737.00
22	1053.00	30.00	37.00	292.00	359.00	0.34	694.00
23	1007.00	26.00	29.00	263.00	318.00	0.32	689.00
24	1091.00	27.00	34.00	326.00	387.00	0.35	704.00
25	967.00	30.00	30.00	288.00	348.00	0.36	619.00
26	954.00	29.00	37.00	287.00	353.00	0.37	601.00
27	999.00	28.00	34.00	245.00	307.00	0.31	692.00
28	1000.00	30.00	29.00	286.00	345.00	0.35	655.00
29	1045.00	34.00	39.00	255.00	328.00	0.31	717.00
30	999.00	31.00	32.00	317.00	380.00	0.38	619.00
Average	991.40	30.00	32.53	262.67	325.20	0.33	666.20

Table A.5: Remote Flooding(N =11)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	969.00	29.00	31.00	278.00	338.00	0.35	631.00
2	964.00	33.00	36.00	209.00	278.00	0.29	686.00
3	883.00	30.00	32.00	206.00	268.00	0.30	615.00
4	915.00	33.00	35.00	229.00	297.00	0.32	618.00
5	943.00	31.00	34.00	204.00	269.00	0.29	674.00
6	1088.00	35.00	39.00	299.00	373.00	0.34	715.00
7	1017.00	31.00	33.00	280.00	344.00	0.34	673.00
8	1033.00	31.00	32.00	310.00	373.00	0.36	660.00
9	926.00	32.00	36.00	241.00	309.00	0.33	617.00
10	933.00	32.00	31.00	263.00	326.00	0.35	607.00
11	1091.00	31.00	36.00	303.00	370.00	0.34	721.00
12	1081.00	31.00	34.00	303.00	368.00	0.34	713.00
13	1072.00	33.00	43.00	276.00	352.00	0.33	720.00
14	947.00	30.00	36.00	219.00	285.00	0.30	662.00
15	1050.00	30.00	35.00	299.00	364.00	0.35	686.00
16	1083.00	31.00	36.00	288.00	355.00	0.33	728.00
17	954.00	31.00	34.00	238.00	303.00	0.32	651.00
18	1055.00	30.00	38.00	302.00	370.00	0.35	685.00
19	1088.00	30.00	35.00	325.00	390.00	0.36	698.00
20	1123.00	82.00	35.00	285.00	402.00	0.36	721.00
21	1039.00	27.00	35.00	270.00	332.00	0.32	707.00
22	1073.00	26.00	41.00	339.00	406.00	0.38	667.00
23	1108.00	30.00	35.00	384.00	449.00	0.41	659.00
24	1242.00	121.00	71.00	323.00	515.00	0.41	727.00
25	1108.00	31.00	33.00	333.00	397.00	0.36	711.00
26	1126.00	31.00	46.00	336.00	413.00	0.37	713.00
27	1113.00	31.00	40.00	368.00	439.00	0.39	674.00
28	889.00	29.00	35.00	276.00	340.00	0.38	549.00
29	936.00	29.00	35.00	272.00	336.00	0.36	600.00
30	1012.00	82.00	44.00	272.00	398.00	0.39	614.00
Average	1028.70	37.10	37.20	284.33	358.63	0.35	670.07

Table A.6: Remote Flooding(N =13)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	929.00	31.00	34.00	214.00	279.00	0.30	650.00
2	974.00	33.00	62.00	235.00	330.00	0.34	644.00
3	1002.00	33.00	37.00	268.00	338.00	0.34	664.00
4	1046.00	31.00	33.00	275.00	339.00	0.32	707.00
5	903.00	31.00	40.00	225.00	296.00	0.33	607.00
6	975.00	34.00	38.00	235.00	307.00	0.31	668.00
7	894.00	30.00	34.00	217.00	281.00	0.31	613.00
8	1036.00	32.00	40.00	288.00	360.00	0.35	676.00
9	857.00	31.00	33.00	213.00	277.00	0.32	580.00
10	1040.00	28.00	144.00	239.00	411.00	0.40	629.00
11	1122.00	31.00	35.00	320.00	386.00	0.34	736.00
12	920.00	30.00	36.00	239.00	305.00	0.33	615.00
13	989.00	30.00	39.00	282.00	351.00	0.35	638.00
14	1119.00	30.00	42.00	364.00	436.00	0.39	683.00
15	3963.00	31.00	38.00	595.00	664.00	0.17	3299.00
16	937.00	30.00	37.00	236.00	303.00	0.32	634.00
17	1048.00	27.00	35.00	255.00	317.00	0.30	731.00
18	990.00	30.00	48.00	273.00	351.00	0.35	639.00
19	1121.00	32.00	40.00	295.00	367.00	0.33	754.00
20	1063.00	44.00	37.00	314.00	395.00	0.37	668.00
21	944.00	29.00	42.00	227.00	298.00	0.32	646.00
22	1075.00	26.00	38.00	235.00	299.00	0.28	776.00
23	1028.00	25.00	39.00	222.00	286.00	0.28	742.00
24	1071.00	25.00	37.00	308.00	370.00	0.35	701.00
25	1139.00	31.00	37.00	364.00	432.00	0.38	707.00
26	1085.00	31.00	38.00	313.00	382.00	0.35	703.00
27	1309.00	30.00	37.00	614.00	681.00	0.52	628.00
28	1042.00	26.00	41.00	245.00	312.00	0.30	730.00
29	1170.00	25.00	38.00	346.00	409.00	0.35	761.00
30	1071.00	31.00	41.00	311.00	383.00	0.36	688.00
Average	1128.73	30.27	42.33	292.23	364.83	0.34	763.90

Table A.7: Remote Flooding(N =15)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1331.00	37.00	44.00	354.00	435.00	0.33	896.00
2	1478.00	37.00	42.00	333.00	412.00	0.28	1066.00
3	1366.00	38.00	43.00	351.00	432.00	0.32	934.00
4	1237.00	37.00	42.00	322.00	401.00	0.32	836.00
5	1370.00	40.00	42.00	365.00	447.00	0.33	923.00
6	1319.00	37.00	42.00	349.00	428.00	0.32	891.00
7	1412.00	37.00	43.00	367.00	447.00	0.32	965.00
8	1413.00	36.00	47.00	439.00	522.00	0.37	891.00
9	1425.00	37.00	42.00	353.00	432.00	0.30	993.00
10	1323.00	37.00	43.00	364.00	444.00	0.34	879.00
11	1369.00	38.00	43.00	349.00	430.00	0.31	939.00
12	1323.00	37.00	43.00	341.00	421.00	0.32	902.00
13	1304.00	37.00	41.00	376.00	454.00	0.35	850.00
14	1382.00	36.00	43.00	382.00	461.00	0.33	921.00
15	1397.00	38.00	43.00	347.00	428.00	0.31	969.00
16	1462.00	37.00	40.00	414.00	491.00	0.34	971.00
17	1501.00	36.00	44.00	424.00	504.00	0.34	997.00
18	1276.00	36.00	40.00	340.00	416.00	0.33	860.00
19	1427.00	38.00	65.00	354.00	457.00	0.32	970.00
20	1242.00	37.00	39.00	335.00	411.00	0.33	831.00
21	1385.00	36.00	68.00	331.00	435.00	0.31	950.00
22	1369.00	37.00	38.00	454.00	529.00	0.39	840.00
23	1341.00	38.00	83.00	318.00	439.00	0.33	902.00
24	1362.00	91.00	40.00	341.00	472.00	0.35	890.00
25	1507.00	38.00	46.00	380.00	464.00	0.31	1043.00
26	1476.00	37.00	45.00	445.00	527.00	0.36	949.00
27	1186.00	39.00	44.00	325.00	408.00	0.34	778.00
28	1397.00	37.00	50.00	454.00	541.00	0.39	856.00
29	1155.00	32.00	41.00	368.00	441.00	0.38	714.00
30	1353.00	37.00	45.00	315.00	397.00	0.29	956.00
Average	1362.93	38.83	45.70	366.33	450.87	0.33	912.07

Table A.8: Remote Flooding(N=17)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1245.00	42.00	46.00	362.00	450.00	0.36	795.00
2	1245.00	36.00	36.00	379.00	451.00	0.36	794.00
3	1231.00	37.00	47.00	270.00	354.00	0.29	877.00
4	1341.00	42.00	45.00	392.00	479.00	0.36	862.00
5	1113.00	36.00	45.00	270.00	351.00	0.32	762.00
6	1290.00	37.00	42.00	304.00	383.00	0.30	907.00
7	1305.00	37.00	45.00	374.00	456.00	0.35	849.00
8	1198.00	38.00	46.00	274.00	358.00	0.30	840.00
9	1222.00	37.00	44.00	290.00	371.00	0.30	851.00
10	1208.00	37.00	45.00	350.00	432.00	0.36	776.00
11	1231.00	36.00	57.00	302.00	395.00	0.32	836.00
12	1225.00	37.00	47.00	340.00	424.00	0.35	801.00
13	1388.00	37.00	45.00	404.00	486.00	0.35	902.00
14	1426.00	36.00	45.00	409.00	490.00	0.34	936.00
15	1506.00	39.00	48.00	396.00	483.00	0.32	1023.00
16	1214.00	36.00	45.00	335.00	416.00	0.34	798.00
17	1420.00	36.00	142.00	336.00	514.00	0.36	906.00
18	1341.00	37.00	45.00	319.00	401.00	0.30	940.00
19	1589.00	37.00	52.00	420.00	509.00	0.32	1080.00
20	1293.00	35.00	45.00	328.00	408.00	0.32	885.00
21	1502.00	37.00	45.00	502.00	584.00	0.39	918.00
22	1444.00	37.00	46.00	406.00	489.00	0.34	955.00
23	1221.00	40.00	47.00	330.00	417.00	0.34	804.00
24	1182.00	39.00	44.00	317.00	400.00	0.34	782.00
25	1246.00	38.00	46.00	314.00	398.00	0.32	848.00
26	1498.00	36.00	51.00	500.00	587.00	0.39	911.00
27	1305.00	38.00	55.00	326.00	419.00	0.32	886.00
28	1428.00	39.00	46.00	326.00	411.00	0.29	1017.00
29	1198.00	36.00	45.00	292.00	373.00	0.31	825.00
30	1194.00	41.00	47.00	307.00	395.00	0.33	799.00
Average	1310.48	37.38	49.59	348.69	435.66	0.33	874.83

Table A.9: Remote Flooding(N=19)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1235.00	36.00	61.00	258.00	355.00	0.29	880.00
2	1333.00	46.00	57.00	276.00	379.00	0.28	954.00
3	1230.00	37.00	48.00	289.00	374.00	0.30	856.00
4	1276.00	38.00	51.00	290.00	379.00	0.30	897.00
5	1323.00	37.00	94.00	339.00	470.00	0.36	853.00
6	1247.00	37.00	52.00	309.00	398.00	0.32	849.00
7	1285.00	86.00	109.00	304.00	499.00	0.39	786.00
8	1280.00	80.00	48.00	303.00	431.00	0.34	849.00
9	1379.00	42.00	64.00	326.00	432.00	0.31	947.00
10	1172.00	40.00	49.00	277.00	366.00	0.31	806.00
11	1219.00	38.00	47.00	275.00	360.00	0.30	859.00
12	1131.00	39.00	48.00	281.00	368.00	0.33	763.00
13	1181.00	39.00	47.00	274.00	360.00	0.30	821.00
14	1305.00	38.00	44.00	404.00	486.00	0.37	819.00
15	1386.00	36.00	43.00	343.00	422.00	0.30	964.00
16	1258.00	42.00	50.00	379.00	471.00	0.37	787.00
17	1227.00	36.00	48.00	275.00	359.00	0.29	868.00
18	1219.00	38.00	44.00	371.00	453.00	0.37	766.00
19	1366.00	42.00	44.00	307.00	393.00	0.29	973.00
20	1251.00	35.00	43.00	294.00	372.00	0.30	879.00
21	1230.00	36.00	46.00	284.00	366.00	0.30	864.00
22	1151.00	38.00	46.00	297.00	381.00	0.33	770.00
23	1389.00	32.00	42.00	286.00	360.00	0.26	1029.00
24	1226.00	35.00	44.00	284.00	363.00	0.30	863.00
25	1467.00	36.00	53.00	499.00	588.00	0.40	879.00
26	1492.00	37.00	45.00	447.00	529.00	0.35	963.00
27	1156.00	35.00	49.00	321.00	405.00	0.35	751.00
28	1819.00	31.00	130.00	284.00	445.00	0.24	1374.00
29	1862.00	31.00	138.00	378.00	547.00	0.29	1315.00
30	1492.00	38.00	41.00	494.00	573.00	0.38	919.00
Average	1319.57	40.37	57.50	324.93	422.80	0.32	896.77

Table A.10: Remote Flooding(N =21)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1167.00	36.00	41.00	245.00	322.00	0.28	845.00
2	1276.00	45.00	60.00	264.00	369.00	0.29	907.00
3	1232.00	35.00	42.00	323.00	400.00	0.32	832.00
4	1312.00	37.00	41.00	272.00	350.00	0.27	962.00
5	1192.00	37.00	57.00	253.00	347.00	0.29	845.00
6	1143.00	38.00	47.00	266.00	351.00	0.31	792.00
7	1230.00	36.00	52.00	273.00	361.00	0.29	869.00
8	1109.00	35.00	48.00	265.00	348.00	0.31	761.00
9	1300.00	36.00	42.00	268.00	346.00	0.27	954.00
10	1113.00	38.00	63.00	257.00	358.00	0.32	755.00
11	1111.00	40.00	47.00	255.00	342.00	0.31	769.00
12	1327.00	36.00	47.00	285.00	368.00	0.28	959.00
13	1314.00	39.00	122.00	261.00	422.00	0.32	892.00
14	1143.00	37.00	45.00	316.00	398.00	0.35	745.00
15	1231.00	34.00	61.00	260.00	355.00	0.29	876.00
16	1310.00	35.00	48.00	282.00	365.00	0.28	945.00
17	1128.00	40.00	44.00	275.00	359.00	0.32	769.00
18	1248.00	38.00	47.00	286.00	371.00	0.30	877.00
19	1145.00	39.00	62.00	274.00	375.00	0.33	770.00
20	1302.00	38.00	39.00	383.00	460.00	0.35	842.00
21	1242.00	35.00	52.00	321.00	408.00	0.33	834.00
22	1189.00	37.00	48.00	331.00	416.00	0.35	773.00
23	1292.00	36.00	50.00	315.00	401.00	0.31	891.00
24	1276.00	37.00	50.00	316.00	403.00	0.32	873.00
25	1361.00	39.00	53.00	415.00	507.00	0.37	854.00
26	1364.00	39.00	53.00	386.00	478.00	0.35	886.00
27	1391.00	35.00	49.00	393.00	477.00	0.34	914.00
28	1350.00	35.00	55.00	324.00	414.00	0.31	936.00
29	1306.00	37.00	49.00	330.00	416.00	0.32	890.00
30	1414.00	36.00	56.00	328.00	420.00	0.30	994.00
Average	1250.60	37.17	52.33	300.73	390.23	0.31	860.37

Table A.11: Remote Flooding(N =23)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1141.00	38.00	50.00	243.00	331.00	0.29	810.00
2	1546.00	43.00	48.00	350.00	441.00	0.29	1105.00
3	1153.00	42.00	47.00	249.00	338.00	0.29	815.00
4	1269.00	43.00	48.00	262.00	353.00	0.28	916.00
5	1357.00	38.00	49.00	236.00	323.00	0.24	1034.00
6	1133.00	36.00	52.00	268.00	356.00	0.31	777.00
7	1282.00	40.00	64.00	295.00	399.00	0.31	883.00
8	1264.00	37.00	51.00	240.00	328.00	0.26	936.00
9	1297.00	36.00	51.00	392.00	479.00	0.37	818.00
10	1243.00	35.00	44.00	304.00	383.00	0.31	860.00
11	1300.00	36.00	50.00	238.00	324.00	0.25	976.00
12	1227.00	36.00	54.00	256.00	346.00	0.28	881.00
13	1431.00	38.00	54.00	428.00	520.00	0.36	911.00
14	2033.00	35.00	152.00	439.00	626.00	0.31	1407.00
15	1326.00	37.00	54.00	272.00	363.00	0.27	963.00
16	1292.00	38.00	55.00	266.00	359.00	0.28	933.00
17	1325.00	39.00	54.00	276.00	369.00	0.28	956.00
18	1424.00	36.00	52.00	290.00	378.00	0.27	1046.00
19	1546.00	37.00	52.00	441.00	530.00	0.34	1016.00
20	1499.00	37.00	52.00	425.00	514.00	0.34	985.00
21	1234.00	37.00	54.00	283.00	374.00	0.30	860.00
22	1477.00	36.00	50.00	421.00	507.00	0.34	970.00
23	1455.00	43.00	52.00	385.00	480.00	0.33	975.00
24	1510.00	38.00	52.00	561.00	651.00	0.43	859.00
25	1374.00	39.00	50.00	302.00	391.00	0.28	983.00
26	1414.00	35.00	54.00	313.00	402.00	0.28	1012.00
27	1389.00	32.00	50.00	294.00	376.00	0.27	1013.00
28	1809.00	32.00	53.00	754.00	839.00	0.46	970.00
29	1738.00	38.00	53.00	308.00	399.00	0.23	1339.00
30	1368.00	37.00	53.00	291.00	381.00	0.28	987.00
Average	1395.20	37.47	55.13	336.07	428.67	0.30	966.53

Table A.12: Remote Flooding(N =25)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1214.00	38.00	53.00	319.00	410.00	0.34	804.00
2	1316.00	37.00	52.00	308.00	397.00	0.30	919.00
3	1325.00	35.00	53.00	303.00	391.00	0.30	934.00
4	1236.00	41.00	55.00	264.00	360.00	0.29	876.00
5	1260.00	40.00	52.00	302.00	394.00	0.31	866.00
6	1436.00	35.00	52.00	395.00	482.00	0.34	954.00
7	1239.00	35.00	52.00	279.00	366.00	0.30	873.00
8	1472.00	33.00	104.00	306.00	443.00	0.30	1029.00
9	1488.00	38.00	54.00	381.00	473.00	0.32	1015.00
10	1504.00	35.00	52.00	403.00	490.00	0.33	1014.00
11	1599.00	43.00	53.00	436.00	532.00	0.33	1067.00
12	1448.00	36.00	52.00	338.00	426.00	0.29	1022.00
13	1441.00	35.00	52.00	332.00	419.00	0.29	1022.00
14	1370.00	36.00	53.00	318.00	407.00	0.30	963.00
15	1301.00	36.00	53.00	308.00	397.00	0.31	904.00
16	1356.00	35.00	57.00	332.00	424.00	0.31	932.00
17	1418.00	36.00	62.00	326.00	424.00	0.30	994.00
18	1387.00	35.00	50.00	277.00	362.00	0.26	1025.00
19	1260.00	40.00	86.00	266.00	392.00	0.31	868.00
20	1487.00	36.00	114.00	467.00	617.00	0.41	870.00
21	1746.00	37.00	69.00	629.00	735.00	0.42	1011.00
22	1416.00	37.00	70.00	250.00	357.00	0.25	1059.00
23	1666.00	40.00	58.00	388.00	486.00	0.29	1180.00
24	1273.00	37.00	50.00	253.00	340.00	0.27	933.00
25	1369.00	43.00	55.00	269.00	367.00	0.27	1002.00
26	1545.00	37.00	48.00	388.00	473.00	0.31	1072.00
27	1274.00	40.00	45.00	216.00	301.00	0.24	973.00
28	1237.00	43.00	51.00	303.00	397.00	0.32	840.00
29	1266.00	35.00	46.00	307.00	388.00	0.31	878.00
30	1202.00	34.00	54.00	289.00	377.00	0.31	825.00
Average	1385.03	37.27	58.57	331.73	427.57	0.31	957.47

Table A.13: Remote Flooding(N =27)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1357.00	40.00	47.00	280.00	367.00	0.27	990.00
2	1276.00	42.00	56.00	270.00	368.00	0.29	908.00
3	1269.00	37.00	47.00	264.00	348.00	0.27	921.00
4	1305.00	39.00	55.00	273.00	367.00	0.28	938.00
5	1444.00	35.00	48.00	299.00	382.00	0.26	1062.00
6	1219.00	36.00	52.00	294.00	382.00	0.31	837.00
7	1458.00	37.00	53.00	406.00	496.00	0.34	962.00
8	1482.00	35.00	78.00	282.00	395.00	0.27	1087.00
9	1326.00	38.00	55.00	236.00	329.00	0.25	997.00
10	1213.00	39.00	50.00	265.00	354.00	0.29	859.00
11	1229.00	37.00	50.00	273.00	360.00	0.29	869.00
12	1333.00	35.00	50.00	252.00	337.00	0.25	996.00
13	1244.00	35.00	51.00	275.00	361.00	0.29	883.00
14	1260.00	37.00	48.00	285.00	370.00	0.29	890.00
15	1244.00	36.00	57.00	283.00	376.00	0.30	868.00
16	1386.00	35.00	53.00	364.00	452.00	0.33	934.00
17	1263.00	37.00	51.00	276.00	364.00	0.29	899.00
18	1427.00	38.00	52.00	288.00	378.00	0.26	1049.00
19	1240.00	35.00	57.00	299.00	391.00	0.32	849.00
20	1431.00	38.00	54.00	385.00	477.00	0.33	954.00
21	1325.00	38.00	58.00	302.00	398.00	0.30	927.00
22	1421.00	35.00	52.00	305.00	392.00	0.28	1029.00
23	1169.00	38.00	52.00	279.00	369.00	0.32	800.00
24	1490.00	55.00	85.00	261.00	401.00	0.27	1089.00
25	1183.00	35.00	46.00	252.00	333.00	0.28	850.00
26	1200.00	39.00	51.00	232.00	322.00	0.27	878.00
27	1337.00	29.00	77.00	228.00	334.00	0.25	1003.00
28	1183.00	35.00	46.00	252.00	333.00	0.28	850.00
29	1200.00	39.00	51.00	232.00	322.00	0.27	878.00
30	1337.00	29.00	77.00	228.00	334.00	0.25	1003.00
Average	1308.37	37.10	55.30	280.67	373.07	0.29	935.30

Table A.14: Remote Flooding(N =28)

Appendix B

Local Flooding With Variable Network Size

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	370.00	12.00	6.00	56.00	74.00	0.20	296.00
2	371.00	12.00	7.00	91.00	110.00	0.30	261.00
3	335.00	12.00	7.00	54.00	73.00	0.22	262.00
4	398.00	12.00	6.00	59.00	77.00	0.19	321.00
5	411.00	12.00	7.00	87.00	106.00	0.26	305.00
6	353.00	13.00	6.00	67.00	86.00	0.24	267.00
7	406.00	13.00	6.00	71.00	90.00	0.22	316.00
8	378.00	13.00	6.00	44.00	63.00	0.17	315.00
9	359.00	13.00	6.00	50.00	69.00	0.19	290.00
10	441.00	12.00	7.00	88.00	107.00	0.24	334.00
11	413.00	13.00	6.00	93.00	112.00	0.27	301.00
12	446.00	13.00	6.00	81.00	100.00	0.22	346.00
13	408.00	13.00	7.00	81.00	101.00	0.25	307.00
14	431.00	13.00	11.00	90.00	114.00	0.26	317.00
15	408.00	13.00	7.00	81.00	101.00	0.25	307.00
16	431.00	13.00	6.00	89.00	108.00	0.25	323.00
17	407.00	14.00	6.00	91.00	111.00	0.27	296.00
18	439.00	12.00	8.00	104.00	124.00	0.28	315.00
19	401.00	13.00	6.00	82.00	101.00	0.25	300.00
20	412.00	13.00	7.00	124.00	144.00	0.35	268.00
21	325.00	12.00	7.00	28.00	47.00	0.14	278.00
22	353.00	12.00	6.00	66.00	84.00	0.24	269.00
23	344.00	12.00	7.00	57.00	76.00	0.22	268.00
24	347.00	12.00	6.00	50.00	68.00	0.20	279.00
25	386.00	13.00	6.00	98.00	117.00	0.30	269.00
26	414.00	12.00	6.00	96.00	114.00	0.28	300.00
27	354.00	12.00	6.00	38.00	56.00	0.16	298.00
28	382.00	14.00	6.00	31.00	51.00	0.13	331.00
29	302.00	13.00	6.00	64.00	83.00	0.27	219.00
30	349.00	13.00	6.00	48.00	67.00	0.19	282.00
Average	385.80	12.63	6.53	71.97	91.13	0.23	294.67

Table B.1: Local Flooding (N =3)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	665.00	33.00	17.00	81.00	131.00	0.20	534.00
2	677.00	18.00	15.00	106.00	139.00	0.21	538.00
3	642.00	19.00	15.00	86.00	120.00	0.19	522.00
4	671.00	20.00	16.00	127.00	163.00	0.24	508.00
5	685.00	27.00	16.00	99.00	142.00	0.21	543.00
6	691.00	19.00	16.00	162.00	197.00	0.29	494.00
7	702.00	23.00	13.00	94.00	130.00	0.19	572.00
8	678.00	18.00	17.00	120.00	155.00	0.23	523.00
9	759.00	21.00	17.00	121.00	159.00	0.21	600.00
10	677.00	18.00	18.00	135.00	171.00	0.25	506.00
11	733.00	18.00	17.00	123.00	158.00	0.22	575.00
12	714.00	23.00	17.00	123.00	163.00	0.23	551.00
13	715.00	24.00	17.00	125.00	166.00	0.23	549.00
14	780.00	23.00	17.00	123.00	163.00	0.21	617.00
15	735.00	24.00	18.00	152.00	194.00	0.26	541.00
16	743.00	20.00	13.00	140.00	173.00	0.23	570.00
17	721.00	25.00	16.00	122.00	163.00	0.23	558.00
18	815.00	25.00	60.00	139.00	224.00	0.27	591.00
19	740.00	24.00	22.00	117.00	163.00	0.22	577.00
20	776.00	24.00	18.00	173.00	215.00	0.28	561.00
21	667.00	35.00	18.00	99.00	152.00	0.23	515.00
22	665.00	23.00	17.00	121.00	161.00	0.24	504.00
23	655.00	23.00	17.00	99.00	139.00	0.21	516.00
24	731.00	24.00	19.00	106.00	149.00	0.20	582.00
25	695.00	19.00	16.00	129.00	164.00	0.24	531.00
26	748.00	24.00	16.00	98.00	138.00	0.18	610.00
27	748.00	21.00	18.00	133.00	172.00	0.23	576.00
28	694.00	22.00	19.00	138.00	179.00	0.26	515.00
29	844.00	23.00	19.00	144.00	186.00	0.22	658.00
30	782.00	24.00	17.00	130.00	171.00	0.22	611.00
Average	718.27	22.80	18.37	122.17	163.33	0.23	554.93

Table B.2: Local Flooding (N =5)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	712.00	20.00	19.00	182.00	221.00	0.31	491.00
2	697.00	20.00	20.00	141.00	181.00	0.26	516.00
3	713.00	19.00	20.00	181.00	220.00	0.31	493.00
4	712.00	51.00	23.00	144.00	218.00	0.31	494.00
5	687.00	19.00	25.00	177.00	221.00	0.32	466.00
6	710.00	24.00	24.00	153.00	201.00	0.28	509.00
7	769.00	24.00	24.00	198.00	246.00	0.32	523.00
8	752.00	24.00	23.00	237.00	284.00	0.38	468.00
9	796.00	25.00	19.00	209.00	253.00	0.32	543.00
10	792.00	24.00	22.00	222.00	268.00	0.34	524.00
11	782.00	23.00	24.00	199.00	246.00	0.31	536.00
12	639.00	24.00	25.00	111.00	160.00	0.25	479.00
13	699.00	21.00	25.00	156.00	202.00	0.29	497.00
14	765.00	18.00	30.00	191.00	239.00	0.31	526.00
15	749.00	24.00	27.00	153.00	204.00	0.27	545.00
16	775.00	18.00	28.00	201.00	247.00	0.32	528.00
17	782.00	19.00	30.00	199.00	248.00	0.32	534.00
18	790.00	18.00	28.00	198.00	244.00	0.31	546.00
19	828.00	18.00	29.00	213.00	260.00	0.31	568.00
20	791.00	24.00	23.00	232.00	279.00	0.35	512.00
21	824.00	24.00	27.00	247.00	298.00	0.36	526.00
22	758.00	42.00	56.00	169.00	267.00	0.35	491.00
23	766.00	49.00	19.00	174.00	242.00	0.32	524.00
24	802.00	26.00	24.00	165.00	215.00	0.27	587.00
25	759.00	25.00	28.00	152.00	205.00	0.27	554.00
26	767.00	23.00	23.00	171.00	217.00	0.28	550.00
27	768.00	25.00	23.00	202.00	250.00	0.33	518.00
28	857.00	25.00	27.00	206.00	258.00	0.30	599.00
29	752.00	24.00	24.00	249.00	297.00	0.39	455.00
30	699.00	27.00	26.00	130.00	183.00	0.26	516.00
Average	756.40	24.90	25.50	185.40	235.80	0.31	520.60

Table B.3: Local Flooding ($N = 7$)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	980.00	30.00	26.00	210.00	266.00	0.27	714.00
2	976.00	33.00	29.00	217.00	279.00	0.29	697.00
3	981.00	33.00	28.00	185.00	246.00	0.25	735.00
4	955.00	30.00	27.00	188.00	245.00	0.26	710.00
5	1012.00	31.00	32.00	183.00	246.00	0.24	766.00
6	969.00	29.00	23.00	181.00	233.00	0.24	736.00
7	1031.00	31.00	43.00	162.00	236.00	0.23	795.00
8	1017.0	30.00	24.00	216.00	270.00	0.27	747.00
9	944.00	30.00	25.00	148.00	203.00	0.22	741.00
10	1007.00	30.00	33.00	202.00	265.00	0.26	742.00
11	987.00	31.00	25.00	201.00	257.00	0.26	730.00
12	986.00	29.00	31.00	210.00	270.00	0.27	716.00
13	996.00	31.00	26.00	148.00	205.00	0.21	791.00
14	1013.00	29.00	28.00	202.00	259.00	0.26	754.00
15	1041.00	30.00	24.00	205.00	259.00	0.25	782.00
16	1020.00	30.00	32.00	222.00	284.00	0.28	736.00
17	1099.00	31.00	24.00	202.00	257.00	0.23	842.00
18	1088.00	32.00	30.00	263.00	325.00	0.30	763.00
19	1140.00	33.00	29.00	195.00	257.00	0.23	883.00
20	1052.00	32.00	30.00	225.00	287.00	0.27	765.00
21	1023.00	30.00	25.00	179.00	234.00	0.23	789.00
22	1032.00	30.00	29.00	213.00	272.00	0.26	760.00
23	1048.00	31.00	24.00	162.00	217.00	0.21	831.00
24	1064.00	31.00	28.00	183.00	242.00	0.23	822.00
25	981.00	30.00	27.00	210.00	267.00	0.27	714.00
26	1111.00	30.00	25.00	259.00	314.00	0.28	797.00
27	1097.00	85.00	28.00	155.00	268.00	0.24	829.00
28	1025.00	30.00	27.00	273.00	330.00	0.32	695.00
29	1015.00	30.00	23.00	157.00	210.00	0.21	805.00
30	991.00	31.00	29.00	172.00	232.00	0.23	759.00
Average	1022.70	32.43	27.80	197.60	257.83	0.25	764.87

Table B.4: Local Flooding (N = 9)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	956.00	33.00	29.00	177.00	239.00	0.25	717.00
2	1061.00	57.00	48.00	236.00	341.00	0.32	720.00
3	956.00	31.00	26.00	215.00	272.00	0.28	684.00
4	1074.00	30.00	27.00	219.00	276.00	0.26	798.00
5	1046.00	32.00	33.00	247.00	312.00	0.30	734.00
6	927.00	30.00	31.00	224.00	285.00	0.31	642.00
7	972.00	30.00	30.00	222.00	282.00	0.29	690.00
8	910.00	31.00	26.00	192.00	249.00	0.27	661.00
9	1052.00	30.00	37.00	260.00	327.00	0.31	725.00
10	988.00	31.00	25.00	210.00	266.00	0.27	722.00
11	1027.00	30.00	27.00	240.00	297.00	0.29	730.00
12	999.00	32.00	32.00	202.00	266.00	0.27	733.00
13	1043.00	31.00	28.00	237.00	296.00	0.28	747.00
14	976.00	29.00	38.00	208.00	275.00	0.28	701.00
15	1093.00	28.00	29.00	280.00	337.00	0.31	756.00
16	1075.00	86.00	40.00	233.00	359.00	0.33	716.00
17	981.00	29.00	30.00	236.00	295.00	0.30	686.00
18	1067.00	29.00	32.00	252.00	313.00	0.29	754.00
19	1080.00	62.00	26.00	282.00	370.00	0.34	710.00
20	1075.00	30.00	30.00	212.00	272.00	0.25	803.00
21	935.00	30.00	33.00	183.00	246.00	0.26	689.00
22	965.00	31.00	26.00	175.00	232.00	0.24	733.00
23	892.00	29.00	30.00	138.00	197.00	0.22	695.00
24	1003.00	30.00	31.00	175.00	236.00	0.24	767.00
25	894.00	29.00	33.00	157.00	219.00	0.24	675.00
26	920.00	31.00	30.00	168.00	229.00	0.25	691.00
27	986.00	25.00	38.00	209.00	272.00	0.28	714.00
28	961.00	30.00	30.00	220.00	280.00	0.29	681.00
29	945.00	29.00	30.00	190.00	249.00	0.26	696.00
30	987.00	30.00	37.00	191.00	258.00	0.26	729.00
Average	994.87	33.83	31.40	213.00	278.23	0.28	716.63

Table B.5: Local Flooding (N = 11)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1063.00	29.00	30.00	222.00	281.00	0.26	782.00
2	1027.00	31.00	42.00	312.00	385.00	0.37	642.00
3	1046.00	31.00	34.00	256.00	321.00	0.31	725.00
4	1122.00	64.00	37.00	272.00	373.00	0.33	749.00
5	1109.00	29.00	37.00	316.00	382.00	0.34	727.00
6	1022.00	28.00	42.00	244.00	314.00	0.31	708.00
7	1057.00	30.00	42.00	295.00	367.00	0.35	690.00
8	1072.00	31.00	36.00	288.00	355.00	0.33	717.00
9	1085.00	29.00	31.00	330.00	390.00	0.36	695.00
10	1079.00	31.00	37.00	302.00	370.00	0.34	709.00
11	1047.00	29.00	38.00	306.00	373.00	0.36	674.00
12	1078.00	30.00	34.00	311.00	375.00	0.35	703.00
13	1109.00	29.00	35.00	328.00	392.00	0.35	717.00
14	1022.00	29.00	42.00	251.00	322.00	0.32	700.00
15	944.00	29.00	35.00	190.00	254.00	0.27	690.00
16	980.00	30.00	38.00	198.00	266.00	0.27	714.00
17	985.00	31.00	41.00	244.00	316.00	0.32	669.00
18	1018.00	31.00	32.00	266.00	329.00	0.32	689.00
19	1017.00	30.00	31.00	261.00	322.00	0.32	695.00
20	981.00	29.00	57.00	210.00	296.00	0.30	685.00
21	964.00	30.00	34.00	213.00	277.00	0.29	687.00
22	1058.00	28.00	34.00	285.00	347.00	0.33	711.00
23	1063.00	29.00	30.00	312.00	371.00	0.35	692.00
24	1070.00	28.00	31.00	267.00	326.00	0.30	744.00
25	1051.00	30.00	31.00	262.00	323.00	0.31	728.00
26	1130.00	29.00	33.00	314.00	376.00	0.33	754.00
27	1100.00	32.00	36.00	335.00	403.00	0.37	697.00
28	1034.00	31.00	32.00	280.00	343.00	0.33	691.00
29	1087.00	30.00	34.00	247.00	311.00	0.29	776.00
30	1120.00	30.00	34.00	277.00	341.00	0.30	779.00
Average	1051.33	30.90	36.00	273.13	340.03	0.32	711.30

Table B.6: Local Flooding (N = 13)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1133.00	30.00	33.00	390.00	453.00	0.40	680.00
2	1030.00	31.00	41.00	311.00	383.00	0.37	647.00
3	1112.00	30.00	36.00	320.00	386.00	0.35	726.00
4	1194.00	34.00	39.00	357.00	430.00	0.36	764.00
5	1132.00	103.00	38.00	315.00	456.00	0.40	676.00
6	1129.00	31.00	36.00	257.00	324.00	0.29	805.00
7	1291.00	31.00	46.00	547.00	624.00	0.48	667.00
8	1002.00	33.00	34.00	283.00	350.00	0.35	652.00
9	1019.00	30.00	36.00	243.00	309.00	0.30	710.00
10	1063.00	28.00	77.00	295.00	400.00	0.38	663.00
11	1056.00	32.00	69.00	303.00	404.00	0.38	652.00
12	971.00	30.00	38.00	228.00	296.00	0.30	675.00
13	998.00	30.00	48.00	279.00	357.00	0.36	641.00
14	1114.00	30.00	70.00	298.00	398.00	0.36	716.00
15	1063.00	31.00	35.00	302.00	368.00	0.35	695.00
16	1065.00	27.00	45.00	257.00	329.00	0.31	736.00
17	1136.00	99.00	38.00	313.00	450.00	0.40	686.00
18	1084.00	61.00	37.00	279.00	377.00	0.35	707.00
19	1047.00	29.00	39.00	281.00	349.00	0.33	698.00
20	1024.00	30.00	40.00	237.00	307.00	0.30	717.00
21	1091.00	31.00	44.00	269.00	344.00	0.32	747.00
22	1073.00	30.00	35.00	255.00	320.00	0.30	753.00
23	1124.00	62.00	42.00	346.00	450.00	0.40	674.00
24	1112.00	29.00	67.00	249.00	345.00	0.31	767.00
25	1142.00	29.00	42.00	356.00	427.00	0.37	715.00
26	1357.00	29.00	38.00	529.00	596.00	0.44	761.00
27	1207.00	31.00	45.00	434.00	510.00	0.42	697.00
28	1370.00	30.00	47.00	527.00	604.00	0.44	766.00
29	1028.00	30.00	43.00	309.00	382.00	0.37	646.00
30	1054.00	30.00	46.00	242.00	318.00	0.30	736.00
Average	1107.37	37.03	44.13	320.37	401.53	0.36	705.83

Table B.7: Local Flooding (N = 15)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1265.00	37.00	34.00	309.00	380.00	0.30	885.00
2	1214.00	37.00	39.00	305.00	381.00	0.31	833.00
3	1216.00	37.00	37.00	271.00	345.00	0.28	871.00
4	1222.00	36.00	35.00	273.00	344.00	0.28	878.00
5	1288.00	36.00	38.00	224.00	298.00	0.23	990.00
6	1277.00	100.00	42.00	272.00	414.00	0.32	863.00
7	1217.00	37.00	39.00	264.00	340.00	0.28	877.00
8	1261.00	36.00	34.00	252.00	322.00	0.26	939.00
9	1296.00	36.00	45.00	248.00	329.00	0.25	967.00
10	1287.00	38.00	35.00	284.00	357.00	0.28	930.00
11	1246.00	36.00	40.00	265.00	341.00	0.27	905.00
12	1268.00	38.00	41.00	314.00	393.00	0.31	875.00
13	1242.00	36.00	40.00	330.00	406.00	0.33	836.00
14	1269.00	37.00	40.00	343.00	420.00	0.33	849.00
15	1346.00	37.00	39.00	319.00	395.00	0.29	951.00
16	1194.00	36.00	36.00	280.00	352.00	0.29	842.00
17	1333.00	36.00	36.00	326.00	398.00	0.30	935.00
18	1257.00	37.00	34.00	323.00	394.00	0.31	863.00
19	1281.00	34.00	35.00	255.00	324.00	0.25	957.00
20	1533.00	36.00	54.00	323.00	413.00	0.27	1120.00
21	1317.00	36.00	37.00	339.00	412.00	0.31	905.00
22	1386.00	39.00	42.00	281.00	362.00	0.26	1024.00
23	1308.00	37.00	42.00	296.00	375.00	0.29	933.00
24	1264.00	36.00	42.00	296.00	374.00	0.30	890.00
25	1337.00	40.00	39.00	309.00	388.00	0.29	949.00
26	1268.00	37.00	43.00	369.00	449.00	0.35	819.00
27	1330.00	38.00	42.00	313.00	393.00	0.30	937.00
28	1291.00	36.00	41.00	329.00	406.00	0.31	885.00
29	1304.00	36.00	42.00	194.00	272.00	0.21	1032.00
30	1323.00	68.00	48.00	280.00	396.00	0.30	927.00
Average	1288.00	39.87	39.70	292.87	372.43	0.29	915.57

Table B.8: Local Flooding (N = 17)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1353.00	48.00	69.00	215.00	332.00	0.25	1021.00
2	1220.00	50.00	37.00	311.00	398.00	0.33	822.00
3	1128.00	36.00	45.00	258.00	339.00	0.30	789.00
4	1283.00	37.00	38.00	386.00	461.00	0.36	822.00
5	1341.00	40.00	58.00	268.00	366.00	0.27	975.00
6	1220.00	41.00	39.00	261.00	341.00	0.28	879.00
7	1333.00	37.00	41.00	281.00	359.00	0.27	974.00
8	1196.00	37.00	39.00	300.00	376.00	0.31	820.00
9	1325.00	44.00	37.00	256.00	337.00	0.25	988.00
10	1147.00	37.00	44.00	265.00	346.00	0.30	801.00
11	1265.00	36.00	40.00	264.00	340.00	0.27	925.00
12	1165.00	36.00	39.00	270.00	345.00	0.30	820.00
13	1260.00	37.00	35.00	311.00	383.00	0.30	877.00
14	1205.00	36.00	40.00	245.00	321.00	0.27	884.00
15	1236.00	37.00	43.00	331.00	411.00	0.33	825.00
16	1301.00	38.00	37.00	401.00	476.00	0.37	825.00
17	1460.00	36.00	41.00	446.00	523.00	0.36	937.00
18	1516.00	38.00	40.00	409.00	487.00	0.32	1029.00
19	1343.00	37.00	45.00	292.00	374.00	0.28	969.00
20	1446.00	38.00	39.00	374.00	451.00	0.31	995.00
21	1498.00	43.00	49.00	403.00	495.00	0.33	1003.00
22	1692.00	36.00	38.00	657.00	731.00	0.43	961.00
23	1285.00	37.00	58.00	336.00	431.00	0.34	854.00
24	1298.00	36.00	40.00	288.00	364.00	0.28	934.00
25	1284.00	37.00	42.00	263.00	342.00	0.27	942.00
26	1326.00	37.00	49.00	268.00	354.00	0.27	972.00
27	1236.00	37.00	49.00	250.00	336.00	0.27	900.00
28	1313.00	36.00	40.00	297.00	373.00	0.28	940.00
29	1321.00	37.00	48.00	311.00	396.00	0.30	925.00
30	1602.00	37.00	42.00	545.00	624.00	0.39	978.00
Average	1319.93	38.30	43.37	325.40	407.07	0.31	912.87

Table B.9: Local Flooding (N = 19)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1376.00	37.00	42.00	393.00	472.00	0.34	904.00
2	1426.00	39.00	41.00	391.00	471.00	0.33	955.00
3	1473.00	37.00	44.00	336.00	417.00	0.28	1056.00
4	1528.00	38.00	39.00	447.00	524.00	0.34	1004.00
5	1291.00	37.00	43.00	395.00	475.00	0.37	816.00
6	1243.00	37.00	39.00	261.00	337.00	0.27	906.00
7	1208.00	39.00	42.00	260.00	341.00	0.28	867.00
8	1276.00	35.00	42.00	269.00	346.00	0.27	930.00
9	1454.00	35.00	58.00	397.00	490.00	0.34	964.00
10	1272.00	39.00	61.00	248.00	348.00	0.27	924.00
11	1238.00	42.00	44.00	310.00	396.00	0.32	842.00
12	1434.00	70.00	49.00	384.00	503.00	0.35	931.00
13	1402.00	35.00	40.00	301.00	376.00	0.27	1026.00
14	1175.00	36.00	46.00	297.00	379.00	0.32	796.00
15	1217.00	37.00	40.00	269.00	346.00	0.28	871.00
16	1261.00	36.00	42.00	239.00	317.00	0.25	944.00
17	1384.00	36.00	41.00	304.00	381.00	0.28	1003.00
18	1369.00	39.00	43.00	294.00	376.00	0.27	993.00
19	1343.00	35.00	38.00	326.00	399.00	0.30	944.00
20	1394.00	38.00	42.00	253.00	333.00	0.24	1061.00
21	1200.00	36.00	45.00	248.00	329.00	0.27	871.00
22	1300.00	117.00	43.00	250.00	410.00	0.32	890.00
23	1329.00	34.00	63.00	275.00	372.00	0.28	957.00
24	1420.00	41.00	50.00	276.00	367.00	0.26	1053.00
25	1417.00	36.00	44.00	325.00	405.00	0.29	1012.00
26	1448.00	37.00	61.00	284.00	382.00	0.26	1066.00
27	1421.00	35.00	49.00	358.00	442.00	0.31	979.00
28	1320.00	42.00	43.00	325.00	410.00	0.31	910.00
29	1336.00	36.00	48.00	257.00	341.00	0.26	995.00
30	1313.00	37.00	43.00	263.00	343.00	0.26	970.00
Average	1342.27	40.93	45.50	307.83	394.27	0.29	948.00

Table B.10: Local Flooding (N = 21)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1496.00	40.00	59.00	320.00	419.00	0.28	1077.00
2	1345.00	37.00	55.00	265.00	357.00	0.27	988.00
3	1449.00	39.00	60.00	366.00	465.00	0.32	984.00
4	1415.00	38.00	58.00	308.00	404.00	0.29	1011.00
5	1373.00	35.00	58.00	333.00	426.00	0.31	947.00
6	1357.00	37.00	60.00	358.00	455.00	0.34	902.00
7	1364.00	36.00	59.00	333.00	428.00	0.31	936.00
8	1540.00	34.00	54.00	336.00	424.00	0.28	1116.00
9	1403.00	36.00	76.00	327.00	439.00	0.31	964.00
10	1640.00	37.00	59.00	496.00	592.00	0.36	1048.00
11	1371.00	36.00	57.00	336.00	429.00	0.31	942.00
12	1358.00	38.00	57.00	346.00	441.00	0.32	917.00
13	1499.00	35.00	56.00	304.00	395.00	0.26	1104.00
14	1397.00	37.00	59.00	310.00	406.00	0.29	991.00
15	1372.00	36.00	53.00	349.00	438.00	0.32	934.00
16	1455.00	38.00	54.00	311.00	403.00	0.28	1052.00
17	1447.00	38.00	51.00	508.00	597.00	0.41	850.00
18	1371.00	36.00	68.00	288.00	392.00	0.29	979.00
19	1374.00	31.00	60.00	276.00	367.00	0.27	1007.00
20	1678.00	31.00	51.00	532.00	614.00	0.37	1064.00
21	1649.00	30.00	56.00	582.00	668.00	0.41	981.00
22	1348.00	30.00	46.00	351.00	427.00	0.32	921.00
23	1401.00	30.00	80.00	267.00	377.00	0.27	1024.00
24	1417.00	35.00	52.00	293.00	380.00	0.27	1037.00
25	1422.00	34.00	55.00	314.00	403.00	0.28	1019.00
26	1513.00	34.00	54.00	376.00	464.00	0.31	1049.00
27	1362.00	31.00	59.00	230.00	320.00	0.23	1042.00
28	1378.00	31.00	51.00	425.00	507.00	0.37	871.00
29	1436.00	30.00	50.00	382.00	462.00	0.32	974.00
30	1578.00	30.00	49.00	551.00	630.00	0.40	948.00
Average	1440.27	34.67	57.20	359.10	450.97	0.31	989.30

Table B.11: Local Flooding (N = 23)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1,517.00	35.00	40.00	552.00	627.00	0.41	890.00
2	1,491.00	35.00	55.00	544.00	634.00	0.43	857.00
3	1,524.00	35.00	57.00	446.00	538.00	0.35	986.00
4	1,604.00	39.00	51.00	485.00	575.00	0.36	1,029.00
5	1,597.00	42.00	49.00	523.00	614.00	0.38	983.00
6	1,843.00	36.00	52.00	681.00	769.00	0.42	1,074.00
7	1,461.00	37.00	67.00	385.00	489.00	0.33	972.00
8	1,322.00	37.00	62.00	369.00	468.00	0.35	854.00
9	1,421.00	34.00	50.00	427.00	511.00	0.36	910.00
10	1,491.00	36.00	51.00	476.00	563.00	0.38	928.00
11	1,368.00	36.00	49.00	406.00	491.00	0.36	877.00
12	1,426.00	34.00	47.00	472.00	553.00	0.39	873.00
13	1,539.00	35.00	50.00	505.00	590.00	0.38	949.00
14	1,640.00	37.00	80.00	490.00	607.00	0.37	1,033.00
15	1,772.00	37.00	61.00	518.00	616.00	0.35	1,156.00
16	1,436.00	36.00	63.00	478.00	577.00	0.40	859.00
17	1,372.00	36.00	61.00	363.00	460.00	0.34	912.00
18	1,494.00	35.00	62.00	467.00	564.00	0.38	930.00
19	1,553.00	36.00	60.00	475.00	571.00	0.37	982.00
20	1,672.00	37.00	60.00	612.00	709.00	0.42	963.00
21	1,536.00	38.00	64.00	367.00	469.00	0.31	1,067.00
22	1,401.00	37.00	58.00	367.00	462.00	0.33	939.00
23	1,462.00	39.00	55.00	434.00	528.00	0.36	934.00
24	1,352.00	36.00	46.00	385.00	467.00	0.35	885.00
25	1,619.00	38.00	57.00	509.00	604.00	0.37	1,015.00
26	1,494.00	41.00	77.00	411.00	529.00	0.35	965.00
27	1,344.00	35.00	49.00	366.00	450.00	0.33	894.00
28	1,429.00	36.00	55.00	461.00	552.00	0.39	877.00
29	1,523.00	34.00	81.00	339.00	454.00	0.30	1,069.00
30	1,673.00	34.00	58.00	669.00	761.00	0.45	912.00
Average	1,512.53	36.43	57.57	466.07	560.07	0.37	952.47

Table B.12: Local Flooding (N = 25)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1464.00	53.00	87.00	452.00	592.00	0.40	872.00
2	1485.00	38.00	41.00	496.00	575.00	0.39	910.00
3	1521.00	52.00	70.00	635.00	757.00	0.50	764.00
4	1685.00	35.00	42.00	482.00	559.00	0.33	1126.00
5	1523.00	34.00	62.00	548.00	644.00	0.42	879.00
6	1688.00	42.00	60.00	478.00	580.00	0.34	1108.00
7	1411.00	35.00	44.00	423.00	502.00	0.36	909.00
8	1401.00	35.00	48.00	425.00	508.00	0.36	893.00
9	1663.00	34.00	51.00	582.00	667.00	0.40	996.00
10	1523.00	35.00	62.00	501.00	598.00	0.39	925.00
11	1674.00	34.00	57.00	687.00	778.00	0.46	896.00
12	2142.00	35.00	51.00	1086.00	1172.00	0.55	970.00
13	1414.00	35.00	49.00	455.00	539.00	0.38	875.00
14	1637.00	38.00	58.00	551.00	647.00	0.40	990.00
15	1815.00	35.00	55.00	710.00	800.00	0.44	1015.00
16	2339.00	40.00	63.00	1010.00	1113.00	0.48	1226.00
17	1515.00	36.00	51.00	483.00	570.00	0.38	945.00
18	1781.00	34.00	81.00	609.00	724.00	0.41	1057.00
19	1453.00	36.00	53.00	454.00	543.00	0.37	910.00
20	1605.00	34.00	57.00	554.00	645.00	0.40	960.00
21	1674.00	34.00	56.00	484.00	574.00	0.34	1100.00
22	1740.00	36.00	55.00	692.00	783.00	0.45	957.00
23	1421.00	36.00	57.00	414.00	507.00	0.36	914.00
24	1875.00	37.00	55.00	646.00	738.00	0.39	1137.00
25	1498.00	36.00	59.00	479.00	574.00	0.38	924.00
26	1509.00	36.00	63.00	493.00	592.00	0.39	917.00
27	1811.00	39.00	66.00	659.00	764.00	0.42	1047.00
28	1496.00	34.00	56.00	502.00	592.00	0.40	904.00
29	1503.00	38.00	62.00	445.00	545.00	0.36	958.00
30	2394.00	34.00	64.00	1081.00	1179.00	0.49	1215.00
Average	1655.33	37.00	57.83	583.87	678.70	0.40	976.63

Table B.13: Local Flooding (N = 27)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1732.00	36.00	61.00	714.00	811.00	0.47	921.00
2	1871.00	35.00	57.00	754.00	846.00	0.45	1025.00
3	1639.00	40.00	67.00	555.00	662.00	0.40	977.00
4	1749.00	36.00	65.00	675.00	776.00	0.44	973.00
5	1819.00	36.00	65.00	756.00	857.00	0.47	962.00
6	1911.00	38.00	62.00	749.00	849.00	0.44	1062.00
7	1590.00	37.00	63.00	542.00	642.00	0.40	948.00
8	1698.00	38.00	77.00	553.00	668.00	0.39	1030.00
9	1554.00	34.00	64.00	465.00	563.00	0.36	991.00
10	1633.00	35.00	64.00	582.00	681.00	0.42	952.00
11	1788.00	37.00	63.00	688.00	788.00	0.44	1000.00
12	1615.00	36.00	58.00	580.00	674.00	0.42	941.00
13	1638.00	39.00	57.00	506.00	602.00	0.37	1036.00
14	1794.00	36.00	57.00	703.00	796.00	0.44	998.00
15	2574.00	34.00	69.00	1418.00	1521.00	0.59	1053.00
16	1605.00	35.00	49.00	654.00	738.00	0.46	867.00
17	1622.00	35.00	57.00	608.00	700.00	0.43	922.00
18	1642.00	39.00	47.00	562.00	648.00	0.39	994.00
19	1557.00	35.00	53.00	497.00	585.00	0.38	972.00
20	1582.00	35.00	53.00	586.00	674.00	0.43	908.00
21	1638.00	34.00	54.00	630.00	718.00	0.44	920.00
22	1628.00	36.00	59.00	586.00	681.00	0.42	947.00
23	1953.00	24.00	48.00	589.00	661.00	0.34	1292.00
24	1747.00	35.00	56.00	702.00	793.00	0.45	954.00
25	1723.00	25.00	51.00	634.00	710.00	0.41	1013.00
26	1566.00	36.00	65.00	480.00	581.00	0.37	985.00
27	1578.00	36.00	65.00	480.00	581.00	0.37	997.00
28	1589.00	36.00	65.00	480.00	581.00	0.37	1008.00
29	1546.00	36.00	65.00	480.00	581.00	0.38	965.00
30	1476.00	36.00	65.00	480.00	581.00	0.39	895.00
Average	1701.90	35.33	60.03	622.93	718.30	0.42	983.60

Table B.14: Local Flooding (N = 28)

Appendix C

Regular Flooding With Variable Network Size

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	372.00	15.00	6.00	12.00	33.00	0.09	339.00
2	369.00	11.50	7.00	12.00	30.50	0.08	338.50
3	369.00	12.00	7.00	12.00	31.00	0.08	338.00
4	394.00	24.00	6.00	11.00	41.00	0.10	353.00
5	332.00	12.00	6.00	46.00	64.00	0.19	268.00
6	323.00	12.00	6.00	12.00	30.00	0.09	293.00
7	299.00	13.00	7.00	11.00	31.00	0.10	268.00
8	397.00	14.50	6.00	14.00	34.50	0.09	362.50
9	326.00	13.00	7.00	12.00	32.00	0.10	294.00
10	357.00	12.00	6.00	17.00	35.00	0.10	322.00
11	306.00	14.00	7.00	14.00	35.00	0.11	271.00
12	327.00	12.00	6.00	12.00	30.00	0.09	297.00
13	354.00	12.00	6.00	12.00	30.00	0.08	324.00
14	365.00	15.00	6.00	13.00	34.00	0.09	331.00
15	374.00	12.00	7.00	11.00	30.00	0.08	344.00
16	354.00	13.00	6.00	11.00	30.00	0.08	324.00
17	369.00	14.00	6.00	11.00	31.00	0.08	338.00
18	359.00	13.00	7.00	12.00	32.00	0.09	327.00
19	371.00	15.00	7.00	14.00	36.00	0.10	335.00
20	380.00	24.00	7.00	13.00	44.00	0.12	336.00
21	363.00	12.50	6.00	15.00	33.50	0.09	329.50
22	364.00	13.00	6.00	13.00	32.00	0.09	332.00
23	363.00	22.00	6.00	12.00	40.00	0.11	323.00
24	367.00	15.50	6.00	13.00	34.50	0.09	332.50
25	367.00	13.50	6.00	11.00	30.50	0.08	336.50
26	371.00	13.50	6.00	11.00	30.50	0.08	340.50
27	404.00	28.00	6.00	12.00	46.00	0.11	358.00
28	351.00	12.00	7.00	11.00	30.00	0.09	321.00
29	369.00	13.00	6.00	48.00	67.00	0.18	302.00
30	372.00	12.00	6.00	12.00	30.00	0.08	342.00
Average	359.60	14.60	6.33	14.67	35.60	0.10	324.00

Table C.1: Regular Flooding ($N = 3$)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	748.00	27.50	17.00	16.50	61.00	0.08	687.00
2	751.00	25.50	16.00	19.50	61.00	0.08	690.00
3	704.00	25.00	16.50	27.50	69.00	0.10	635.00
4	680.00	23.50	15.50	14.00	53.00	0.08	627.00
5	684.00	26.00	16.00	21.00	63.00	0.09	621.00
6	613.00	20.50	16.00	14.50	51.00	0.08	562.00
7	722.00	25.00	17.50	16.00	58.50	0.08	663.50
8	702.00	20.50	16.00	16.00	52.50	0.07	649.50
9	693.00	25.00	16.50	26.50	68.00	0.10	625.00
10	685.00	22.00	18.00	15.00	55.00	0.08	630.00
11	765.00	23.00	17.50	15.50	56.00	0.07	709.00
12	739.00	26.00	16.50	24.00	66.50	0.09	672.50
13	701.00	21.50	17.00	15.50	54.00	0.08	647.00
14	666.00	20.50	14.50	15.00	50.00	0.08	616.00
15	715.00	26.50	19.00	16.50	62.00	0.09	653.00
16	661.00	27.00	16.50	22.00	65.50	0.10	595.50
17	737.00	21.50	20.00	17.50	59.00	0.08	678.00
18	686.00	20.50	17.50	16.50	54.50	0.08	631.50
19	737.00	25.50	15.50	16.00	57.00	0.08	680.00
20	718.00	24.50	23.00	19.00	66.50	0.09	651.50
21	747.00	20.50	16.00	17.00	53.50	0.07	693.50
22	699.00	26.00	17.50	10.50	54.00	0.08	645.00
23	775.00	20.00	16.50	18.50	55.00	0.07	720.00
24	764.00	25.00	29.00	23.00	77.00	0.10	687.00
25	766.00	21.00	18.50	16.50	56.00	0.07	710.00
26	781.00	25.50	24.50	20.00	70.00	0.09	711.00
27	771.00	20.50	16.50	19.50	56.50	0.07	714.50
28	764.00	25.00	17.00	22.00	64.00	0.08	700.00
29	816.00	21.50	23.00	16.50	61.00	0.07	755.00
30	887.00	24.50	17.50	18.00	60.00	0.07	827.00
Average	729.23	23.55	17.93	18.18	59.67	0.08	669.57

Table C.2: Regular Flooding (N = 5)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	653.00	28.25	28.00	21.00	77.25	0.12	575.75
2	673.00	27.75	26.50	20.50	74.75	0.11	598.25
3	646.00	24.25	55.50	22.00	101.75	0.16	544.25
4	890.00	25.50	55.50	22.00	103.00	0.12	787.00
5	651.00	20.50	24.00	22.00	66.50	0.10	584.50
6	671.00	28.00	24.50	24.00	76.50	0.11	594.50
7	691.00	42.25	28.00	51.00	121.25	0.18	569.75
8	644.00	28.00	27.50	25.50	81.00	0.13	563.00
9	660.00	21.25	23.00	20.00	64.25	0.10	595.75
10	728.00	23.50	26.00	20.50	70.00	0.10	658.00
11	693.00	26.50	23.00	21.00	70.50	0.10	622.50
12	686.00	27.00	25.50	21.00	73.50	0.11	612.50
13	724.00	23.50	23.00	22.00	68.50	0.09	655.50
14	682.00	26.75	25.00	22.00	73.75	0.11	608.25
15	703.00	27.50	24.00	33.50	85.00	0.12	618.00
16	708.00	26.75	26.00	19.50	72.25	0.10	635.75
17	741.00	30.25	37.50	21.00	88.75	0.12	652.25
18	660.00	27.50	28.50	22.50	78.50	0.12	581.50
19	719.00	27.50	27.00	24.00	78.50	0.11	640.50
20	667.00	27.50	29.00	22.00	78.50	0.12	588.50
21	693.00	25.00	26.50	22.00	73.50	0.11	619.50
22	685.00	27.25	26.00	21.00	74.25	0.11	610.75
23	681.00	27.25	27.00	22.50	76.75	0.11	604.25
24	714.00	26.75	24.50	19.50	70.75	0.10	643.25
25	722.00	28.50	26.50	22.00	77.00	0.11	645.00
26	681.00	26.75	27.00	21.50	75.25	0.11	605.75
27	740.00	22.25	55.50	23.50	101.25	0.14	638.75
28	698.00	25.00	23.00	20.00	68.00	0.10	630.00
29	749.00	22.75	25.00	27.50	75.25	0.10	673.75
30	729.00	22.50	27.00	21.00	70.50	0.10	658.50
Average	699.40	26.47	29.17	23.25	78.88	0.11	620.52

Table C.3: Regular Flooding ($N = 7$)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1067.00	35.75	32.25	25.25	93.25	0.09	973.75
2	1110.00	38.25	32.00	24.50	94.75	0.09	1015.25
3	1256.00	38.75	35.25	24.00	98.00	0.08	1158.00
4	1049.00	29.75	30.75	23.50	84.00	0.08	965.00
5	997.00	59.75	28.50	23.50	111.75	0.11	885.25
6	1057.00	49.00	31.50	29.00	109.50	0.10	947.50
7	1077.00	35.00	30.50	22.75	88.25	0.08	988.75
8	1052.00	31.00	34.75	23.25	89.00	0.08	963.00
9	1117.00	35.50	26.75	22.50	84.75	0.08	1032.25
10	1080.00	34.75	31.75	23.75	90.25	0.08	989.75
11	1069.00	33.75	27.00	25.00	85.75	0.08	983.25
12	1167.00	33.00	29.00	22.25	84.25	0.07	1082.75
13	1170.00	36.75	31.25	31.50	99.50	0.09	1070.50
14	1172.00	35.75	29.75	25.00	90.50	0.08	1081.50
15	1245.00	34.50	36.00	28.75	99.25	0.08	1145.75
16	1184.00	34.75	32.75	24.50	92.00	0.08	1092.00
17	1182.00	36.00	28.75	23.75	88.50	0.07	1093.50
18	1029.00	34.75	30.50	23.75	89.00	0.09	940.00
19	950.00	34.25	30.25	21.50	86.00	0.09	864.00
20	987.00	32.50	26.00	25.00	83.50	0.08	903.50
21	1023.00	30.75	30.00	23.00	83.75	0.08	939.25
22	962.00	35.50	28.25	23.25	87.00	0.09	875.00
23	969.00	34.25	30.50	21.75	86.50	0.09	882.50
24	1030.00	28.75	57.75	22.00	108.50	0.11	921.50
25	1040.00	31.00	28.50	21.75	81.25	0.08	958.75
26	1040.00	34.50	37.00	28.00	99.50	0.10	940.50
27	1000.00	34.25	30.25	21.75	86.25	0.09	913.75
28	1061.00	29.75	52.00	24.00	105.75	0.10	955.25
29	1051.00	33.00	32.25	23.25	88.50	0.08	962.50
30	1051.00	34.25	32.00	31.50	97.75	0.09	953.25
Average	1074.80	35.32	32.46	24.44	92.22	0.09	982.58

Table C.4: Regular Flooding ($N = 9$)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	958.00	36.50	35.00	26.50	98.00	0.10	860.00
2	1017.00	36.75	31.50	31.75	100.00	0.10	917.00
3	940.00	34.75	35.25	26.75	96.75	0.10	843.25
4	1163.00	36.75	32.00	26.25	95.00	0.08	1068.00
5	970.00	35.50	35.00	25.00	95.50	0.10	874.50
6	1028.00	32.75	31.00	27.50	91.25	0.09	936.75
7	1063.00	38.50	34.25	25.25	98.00	0.09	965.00
8	1105.00	61.25	30.75	24.50	116.50	0.11	988.50
9	1020.00	36.75	61.75	42.25	140.75	0.14	879.25
10	1144.00	32.75	31.00	27.50	91.25	0.08	1052.75
11	1021.00	37.00	35.00	26.75	98.75	0.10	922.25
12	1057.00	29.25	33.75	54.00	117.00	0.11	940.00
13	1035.00	36.50	35.00	25.00	96.50	0.09	938.50
14	1105.00	30.75	59.50	25.00	115.25	0.10	989.75
15	1050.00	36.00	33.75	29.25	99.00	0.09	951.00
16	1103.00	32.00	68.50	29.00	129.50	0.12	973.50
17	1119.00	29.50	30.00	53.25	112.75	0.10	1006.25
18	1070.00	30.75	34.50	63.00	128.25	0.12	941.75
19	1031.00	35.50	40.75	30.00	106.25	0.10	924.75
20	991.00	35.75	37.50	28.25	101.50	0.10	889.50
21	1047.00	31.25	28.50	27.25	87.00	0.08	960.00
22	905.00	37.50	31.25	25.50	94.25	0.10	810.75
23	1035.00	30.00	30.75	24.50	85.25	0.08	949.75
24	1026.00	33.50	31.00	26.25	90.75	0.09	935.25
25	1003.00	29.50	29.75	28.00	87.25	0.09	915.75
26	989.00	32.25	33.50	26.50	92.25	0.09	896.75
27	1060.00	62.50	36.00	27.50	126.00	0.12	934.00
28	1039.00	31.75	33.00	29.75	94.50	0.09	944.50
29	1021.00	29.75	32.75	28.50	91.00	0.09	930.00
30	1023.00	30.50	32.50	25.00	88.00	0.09	935.00
Average	1037.93	35.46	36.16	30.52	102.13	0.10	935.80

Table C.5: Regular Flooding ($N = 11$)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1099.00	39.00	46.00	30.00	115.00	0.10	984.00
2	1135.00	38.00	45.00	25.00	108.00	0.10	1027.00
3	1098.00	37.00	42.00	30.00	109.00	0.10	989.00
4	1116.00	36.00	46.00	26.00	108.00	0.10	1008.00
5	1152.00	38.00	45.00	30.00	113.00	0.10	1039.00
6	1173.00	39.00	44.00	27.00	110.00	0.09	1063.00
7	1171.00	37.00	45.00	30.00	112.00	0.10	1059.00
8	1117.00	37.00	41.00	34.00	112.00	0.10	1005.00
9	1154.00	37.00	42.00	29.00	108.00	0.09	1046.00
10	1023.00	37.00	40.00	30.00	107.00	0.10	916.00
11	1099.00	37.00	39.00	31.00	107.00	0.10	992.00
12	1028.00	38.00	36.00	29.00	103.00	0.10	925.00
13	1057.00	39.00	40.00	28.00	107.00	0.10	950.00
14	991.00	37.00	41.00	27.00	105.00	0.11	886.00
15	1054.00	43.00	39.00	34.00	116.00	0.11	938.00
16	1083.00	37.00	41.00	31.00	109.00	0.10	974.00
17	999.00	50.00	42.00	29.00	121.00	0.12	878.00
18	1073.00	38.00	38.00	33.00	109.00	0.10	964.00
19	1051.00	38.00	49.00	31.00	118.00	0.11	933.00
20	1081.00	38.00	41.00	29.00	108.00	0.10	973.00
21	1065.00	36.00	44.00	29.00	109.00	0.10	956.00
22	1086.00	38.00	43.00	32.00	113.00	0.10	973.00
23	1103.00	37.00	45.00	29.00	111.00	0.10	992.00
24	1096.00	37.00	43.00	29.00	109.00	0.10	987.00
25	1152.00	37.00	43.00	28.00	108.00	0.09	1044.00
26	1107.00	38.00	41.00	33.00	112.00	0.10	995.00
27	1086.00	37.00	43.00	29.00	109.00	0.10	977.00
28	1072.00	38.00	41.00	30.00	109.00	0.10	963.00
29	1131.00	37.00	41.00	27.00	105.00	0.09	1026.00
30	1223.00	38.00	85.00	34.00	157.00	0.13	1066.00
Average	1095.83	38.10	43.70	29.77	111.57	0.10	984.27

Table C.6: Regular Flooding (N = 13)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1037.00	39.00	41.00	32.00	112.00	0.11	925.00
2	1003.00	39.00	47.00	30.00	116.00	0.12	887.00
3	1097.00	40.00	75.00	33.00	148.00	0.13	949.00
4	1281.00	41.00	49.00	32.00	122.00	0.10	1159.00
5	1153.00	38.00	50.00	31.00	119.00	0.10	1034.00
6	1131.00	38.00	48.00	35.00	121.00	0.11	1010.00
7	1134.00	41.00	49.00	31.00	121.00	0.11	1013.00
8	1148.00	68.00	51.00	32.00	151.00	0.13	997.00
9	1152.00	41.00	50.00	26.00	117.00	0.10	1035.00
10	1089.00	39.00	93.00	40.00	172.00	0.16	917.00
11	945.00	38.00	45.00	42.00	125.00	0.13	820.00
12	1022.00	41.00	46.00	33.00	120.00	0.12	902.00
13	1070.00	40.00	58.00	45.00	143.00	0.13	927.00
14	915.00	36.00	49.00	35.00	120.00	0.13	795.00
15	1012.00	112.00	58.00	55.00	225.00	0.22	787.00
16	959.00	36.00	40.00	32.00	108.00	0.11	851.00
17	1047.00	52.00	44.00	34.00	130.00	0.12	917.00
18	1006.00	36.00	40.00	31.00	107.00	0.11	899.00
19	961.00	39.00	44.00	30.00	113.00	0.12	848.00
20	978.00	37.00	44.00	30.00	111.00	0.11	867.00
21	973.00	38.00	44.00	31.00	113.00	0.12	860.00
22	1020.00	38.00	43.00	33.00	114.00	0.11	906.00
23	1003.00	39.00	44.00	31.00	114.00	0.11	889.00
24	1015.00	37.00	41.00	36.00	114.00	0.11	901.00
25	1104.00	39.00	42.00	66.00	147.00	0.13	957.00
26	1019.00	41.00	45.00	33.00	119.00	0.12	900.00
27	1090.00	40.00	48.00	41.00	129.00	0.12	961.00
28	1067.00	39.00	44.00	31.00	114.00	0.11	953.00
29	1161.00	42.00	46.00	86.00	174.00	0.15	987.00
30	1117.00	41.00	45.00	32.00	118.00	0.11	999.00
Average	1056.97	42.83	48.77	36.97	128.57	0.12	928.40

Table C.7: Regular Flooding (N = 15)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1423.00	44.00	48.00	76.00	168.00	0.12	1255.00
2	1282.00	58.00	48.00	32.00	138.00	0.11	1144.00
3	1359.00	40.00	42.00	33.00	115.00	0.08	1244.00
4	1411.00	43.00	41.00	33.00	117.00	0.08	1294.00
5	1414.00	46.00	45.00	33.00	124.00	0.09	1290.00
6	1398.00	44.00	50.00	39.00	133.00	0.10	1265.00
7	1453.00	45.00	48.00	31.00	124.00	0.09	1329.00
8	1460.00	47.00	48.00	31.00	126.00	0.09	1334.00
9	1430.00	48.00	49.00	30.00	127.00	0.09	1303.00
10	1358.00	46.00	48.00	33.00	127.00	0.09	1231.00
11	1422.00	47.00	49.00	33.00	129.00	0.09	1293.00
12	1407.00	46.00	51.00	39.00	136.00	0.10	1271.00
13	1409.00	47.00	48.00	32.00	127.00	0.09	1282.00
14	1441.00	48.00	48.00	43.00	139.00	0.10	1302.00
15	1693.00	91.00	75.00	72.00	238.00	0.14	1455.00
16	1419.00	47.00	47.00	31.00	125.00	0.09	1294.00
17	1446.00	46.00	52.00	35.00	133.00	0.09	1313.00
18	1458.00	48.00	52.00	34.00	134.00	0.09	1324.00
19	1437.00	45.00	58.00	34.00	137.00	0.10	1300.00
20	1554.00	47.00	49.00	34.00	130.00	0.08	1424.00
21	1519.00	46.00	53.00	30.00	129.00	0.08	1390.00
22	1466.00	46.00	51.00	30.00	127.00	0.09	1339.00
23	1500.00	47.00	51.00	33.00	131.00	0.09	1369.00
24	1542.00	46.00	52.00	36.00	134.00	0.09	1408.00
25	1626.00	44.00	65.00	49.00	158.00	0.10	1468.00
26	1658.00	44.00	52.00	41.00	137.00	0.08	1521.00
27	1462.00	46.00	51.00	37.00	134.00	0.09	1328.00
28	1476.00	42.00	59.00	31.00	132.00	0.09	1344.00
29	1506.00	46.00	47.00	33.00	126.00	0.08	1380.00
30	1492.00	45.00	50.00	33.00	128.00	0.09	1364.00
31	1568.00	45.00	50.00	32.00	127.00	0.08	1441.00
Average	1467.39	47.42	50.87	36.87	135.16	0.09	1332.23

Table C.8: Regular Flooding (N = 17)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1334.00	46.00	52.00	35.00	133.00	0.10	1201.00
2	1382.00	46.00	52.00	42.00	140.00	0.10	1242.00
3	1342.00	46.00	49.00	37.00	132.00	0.10	1210.00
4	1598.00	48.00	51.00	35.00	134.00	0.08	1464.00
5	1337.00	46.00	52.00	34.00	132.00	0.10	1205.00
6	1482.00	46.00	64.00	34.00	144.00	0.10	1338.00
7	1436.00	47.00	49.00	37.00	133.00	0.09	1303.00
8	1489.00	47.00	54.00	35.00	136.00	0.09	1353.00
9	1431.00	46.00	53.00	33.00	132.00	0.09	1299.00
10	1321.00	49.00	50.00	38.00	137.00	0.10	1184.00
11	1275.00	43.00	50.00	34.00	127.00	0.10	1148.00
12	1414.00	44.00	78.00	34.00	156.00	0.11	1258.00
13	1372.00	41.00	45.00	35.00	121.00	0.09	1251.00
14	1309.00	46.00	47.00	36.00	129.00	0.10	1180.00
15	1397.00	43.00	49.00	35.00	127.00	0.09	1270.00
16	1343.00	44.00	45.00	40.00	129.00	0.10	1214.00
17	1248.00	43.00	49.00	34.00	126.00	0.10	1122.00
18	1360.00	43.00	44.00	36.00	123.00	0.09	1237.00
19	1382.00	45.00	48.00	35.00	128.00	0.09	1254.00
20	1415.00	133.00	55.00	40.00	228.00	0.16	1187.00
21	1402.00	45.00	48.00	34.00	127.00	0.09	1275.00
22	1381.00	46.00	53.00	33.00	132.00	0.10	1249.00
23	1440.00	43.00	45.00	34.00	122.00	0.08	1318.00
24	1449.00	44.00	49.00	33.00	126.00	0.09	1323.00
25	1429.00	51.00	46.00	36.00	133.00	0.09	1296.00
26	1380.00	46.00	57.00	31.00	134.00	0.10	1246.00
27	1423.00	47.00	48.00	34.00	129.00	0.09	1294.00
28	1488.00	45.00	46.00	35.00	126.00	0.08	1362.00
29	1519.00	48.00	50.00	36.00	134.00	0.09	1385.00
30	1485.00	47.00	53.00	38.00	138.00	0.09	1347.00
Average	1402.10	48.47	51.03	35.43	134.93	0.10	1267.17

Table C.9: Regular Flooding (N = 19)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1326.00	47.00	56.00	35.00	138.00	0.10	1188.00
2	1540.00	47.00	51.00	34.00	132.00	0.09	1408.00
3	1430.00	44.00	53.00	33.00	130.00	0.09	1300.00
4	1531.00	47.00	53.00	32.00	132.00	0.09	1399.00
5	1509.00	48.00	56.00	35.00	139.00	0.09	1370.00
6	1502.00	49.00	64.00	34.00	147.00	0.10	1355.00
7	1458.00	45.00	54.00	32.00	131.00	0.09	1327.00
8	1498.00	45.00	55.00	33.00	133.00	0.09	1365.00
9	1564.00	47.00	59.00	36.00	142.00	0.09	1422.00
10	1585.00	43.00	54.00	31.00	128.00	0.08	1457.00
11	1514.00	77.00	53.00	37.00	167.00	0.11	1347.00
12	1527.00	43.00	52.00	33.00	128.00	0.08	1399.00
13	1505.00	48.00	55.00	41.00	144.00	0.10	1361.00
14	1556.00	47.00	51.00	36.00	134.00	0.09	1422.00
15	1553.00	43.00	66.00	35.00	144.00	0.09	1409.00
16	1611.00	41.00	46.00	37.00	124.00	0.08	1487.00
17	1444.00	45.00	49.00	42.00	136.00	0.09	1308.00
18	1530.00	40.00	42.00	71.00	153.00	0.10	1377.00
19	1430.00	42.00	45.00	29.00	116.00	0.08	1314.00
20	1369.00	45.00	46.00	35.00	126.00	0.09	1243.00
21	1526.00	49.00	49.00	86.00	184.00	0.12	1342.00
22	1389.00	43.00	49.00	36.00	128.00	0.09	1261.00
23	1460.00	44.00	46.00	34.00	124.00	0.08	1336.00
24	1451.00	49.00	56.00	34.00	139.00	0.10	1312.00
25	1615.00	43.00	43.00	91.00	177.00	0.11	1438.00
26	1406.00	46.00	52.00	34.00	132.00	0.09	1274.00
27	1393.00	48.00	78.00	36.00	162.00	0.12	1231.00
28	1330.00	43.00	45.00	35.00	123.00	0.09	1207.00
29	1447.00	46.00	49.00	33.00	128.00	0.09	1319.00
30	1470.00	44.00	50.00	35.00	129.00	0.09	1341.00
Average	1482.30	46.27	52.57	39.50	138.33	0.09	1343.97

Table C.10: Regular Flooding (N = 21)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1430.00	47.00	103.00	42.00	192.00	0.13	1238.00
2	1263.00	48.00	58.00	37.00	143.00	0.11	1120.00
3	1375.00	47.00	58.00	35.00	140.00	0.10	1235.00
4	1458.00	48.00	50.00	35.00	133.00	0.09	1325.00
5	1398.00	51.00	55.00	43.00	149.00	0.11	1249.00
6	1392.00	48.00	53.00	36.00	137.00	0.10	1255.00
7	1380.00	45.00	56.00	45.00	146.00	0.11	1234.00
8	1380.00	50.00	61.00	36.00	147.00	0.11	1233.00
9	1412.00	45.00	60.00	37.00	142.00	0.10	1270.00
10	1497.00	47.00	56.00	35.00	138.00	0.09	1359.00
11	1427.00	47.00	63.00	36.00	146.00	0.10	1281.00
12	1598.00	49.00	62.00	86.00	197.00	0.12	1401.00
13	1474.00	54.00	59.00	38.00	151.00	0.10	1323.00
14	1502.00	50.00	61.00	40.00	151.00	0.10	1351.00
15	1575.00	46.00	62.00	34.00	142.00	0.09	1433.00
16	1433.00	43.00	59.00	35.00	137.00	0.10	1296.00
17	1454.00	49.00	55.00	37.00	141.00	0.10	1313.00
18	1457.00	46.00	57.00	38.00	141.00	0.10	1316.00
19	1328.00	46.00	60.00	42.00	148.00	0.11	1180.00
20	1386.00	48.00	58.00	39.00	145.00	0.10	1241.00
21	1471.00	49.00	56.00	38.00	143.00	0.10	1328.00
22	1374.00	46.00	52.00	35.00	133.00	0.10	1241.00
23	1476.00	48.00	58.00	35.00	141.00	0.10	1335.00
24	1343.00	46.00	60.00	38.00	144.00	0.11	1199.00
25	1524.00	46.00	56.00	38.00	140.00	0.09	1384.00
26	1447.00	47.00	58.00	37.00	142.00	0.10	1305.00
27	1510.00	47.00	69.00	35.00	151.00	0.10	1359.00
28	1450.00	48.00	60.00	35.00	143.00	0.10	1307.00
29	1639.00	49.00	72.00	38.00	159.00	0.10	1480.00
30	1537.00	87.00	100.00	42.00	229.00	0.15	1308.00
Average	1446.33	48.90	61.57	39.23	149.70	0.10	1296.63

Table C.11: Regular Flooding (N = 23)

Test Numb	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1278.00	46.00	54.00	47.00	147.00	0.12	1131.00
2	1333.00	47.00	71.00	55.00	173.00	0.13	1160.00
3	1282.00	41.00	48.00	38.00	127.00	0.10	1155.00
4	1362.00	51.00	53.00	41.00	145.00	0.11	1217.00
5	1331.00	45.00	55.00	37.00	137.00	0.10	1194.00
6	1330.00	46.00	57.00	42.00	145.00	0.11	1185.00
7	1329.00	44.00	50.00	39.00	133.00	0.10	1196.00
8	1351.00	48.00	55.00	35.00	138.00	0.10	1213.00
9	1393.00	48.00	55.00	36.00	139.00	0.10	1254.00
10	1425.00	46.00	54.00	38.00	138.00	0.10	1287.00
11	1558.00	46.00	61.00	38.00	145.00	0.09	1413.00
12	1462.00	44.00	58.00	38.00	140.00	0.10	1322.00
13	1371.00	46.00	58.00	38.00	142.00	0.10	1229.00
14	1369.00	49.00	57.00	36.00	142.00	0.10	1227.00
15	1464.00	46.00	57.00	35.00	138.00	0.09	1326.00
16	1424.00	48.00	58.00	40.00	146.00	0.10	1278.00
17	1457.00	44.00	56.00	37.00	137.00	0.09	1320.00
18	1580.00	49.00	57.00	36.00	142.00	0.09	1438.00
19	1479.00	46.00	56.00	38.00	140.00	0.09	1339.00
20	1520.00	47.00	58.00	36.00	141.00	0.09	1379.00
21	1475.00	47.00	58.00	36.00	141.00	0.10	1334.00
22	1418.00	51.00	57.00	38.00	146.00	0.10	1272.00
23	1577.00	49.00	59.00	36.00	144.00	0.09	1433.00
24	1526.00	45.00	59.00	36.00	140.00	0.09	1386.00
25	1738.00	49.00	60.00	41.00	150.00	0.09	1588.00
26	1431.00	51.00	69.00	37.00	157.00	0.11	1274.00
27	1535.00	43.00	57.00	36.00	136.00	0.09	1399.00
28	1653.00	46.00	61.00	37.00	144.00	0.09	1509.00
29	1986.00	72.00	83.00	36.00	191.00	0.10	1795.00
30	1718.00	88.00	57.00	41.00	186.00	0.11	1532.00
Average	1471.83	48.93	58.27	38.47	145.67	0.10	1326.17

Table C.12: Regular Flooding ($N = 25$)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1426.00	49.00	73.00	40.00	162.00	0.11	1264.00
2	1498.00	47.00	56.00	43.00	146.00	0.10	1352.00
3	1325.00	46.00	56.00	39.00	141.00	0.11	1184.00
4	1408.00	44.00	55.00	38.00	137.00	0.10	1271.00
5	1412.00	46.00	55.00	39.00	140.00	0.10	1272.00
6	1435.00	47.00	58.00	37.00	142.00	0.10	1293.00
7	1438.00	50.00	57.00	38.00	145.00	0.10	1293.00
8	1485.00	48.00	58.00	38.00	144.00	0.10	1341.00
9	1611.00	215.00	64.00	44.00	323.00	0.20	1288.00
10	1474.00	50.00	63.00	40.00	153.00	0.10	1321.00
11	1480.00	48.00	75.00	37.00	160.00	0.11	1320.00
12	1496.00	50.00	57.00	41.00	148.00	0.10	1348.00
13	1489.00	47.00	66.00	40.00	153.00	0.10	1336.00
14	1565.00	48.00	58.00	38.00	144.00	0.09	1421.00
15	1623.00	48.00	61.00	36.00	145.00	0.09	1478.00
16	1518.00	51.00	61.00	39.00	151.00	0.10	1367.00
17	1702.00	49.00	65.00	38.00	152.00	0.09	1550.00
18	1550.00	48.00	62.00	38.00	148.00	0.10	1402.00
19	1680.00	46.00	62.00	37.00	145.00	0.09	1535.00
20	1596.00	47.00	63.00	44.00	154.00	0.10	1442.00
21	1651.00	46.00	70.00	45.00	161.00	0.10	1490.00
22	1619.00	54.00	81.00	38.00	173.00	0.11	1446.00
23	1623.00	54.00	62.00	40.00	156.00	0.10	1467.00
24	1584.00	54.00	74.00	43.00	171.00	0.11	1413.00
25	1622.00	51.00	59.00	42.00	152.00	0.09	1470.00
26	1392.00	48.00	59.00	43.00	150.00	0.11	1242.00
27	1499.00	43.00	68.00	18.00	129.00	0.09	1370.00
28	1347.00	46.00	55.00	39.00	140.00	0.10	1207.00
29	1426.00	46.00	58.00	44.00	148.00	0.10	1278.00
30	1445.00	49.00	56.00	36.00	141.00	0.10	1304.00
Average	1513.97	53.83	62.23	39.07	155.13	0.10	1358.83

Table C.13: Regular Flooding (N = 27)

Appendix D

Single Agent With Variable Network Size (BFS)

Search Time	router	overhead %	search without overhead
254.00	19.00	0.07	235.00
318.00	20.00	0.06	298.00
242.00	20.00	0.08	222.00
297.00	20.00	0.07	277.00
290.00	19.00	0.07	271.00
268.00	20.00	0.07	248.00
247.00	20.00	0.08	227.00
319.00	20.00	0.06	299.00
274.00	20.00	0.07	254.00
271.00	21.00	0.08	250.00
318.00	19.00	0.06	299.00
272.00	22.00	0.08	250.00
317.00	20.00	0.06	297.00
273.00	19.00	0.07	254.00
319.00	20.00	0.06	299.00
272.00	19.00	0.07	253.00
293.00	19.00	0.06	274.00
284.00	19.00	0.07	265.00
313.00	18.00	0.06	295.00
273.00	19.00	0.07	254.00
315.00	20.00	0.06	295.00
270.00	20.00	0.07	250.00
313.00	19.00	0.06	294.00
269.00	19.00	0.07	250.00
314.00	19.00	0.06	295.00
272.00	18.00	0.07	254.00
313.00	19.00	0.06	294.00
284.00	21.00	0.07	263.00
319.00	21.00	0.07	298.00
288.00	20.00	0.07	268.00
289.03	19.63	0.07	269.40

Search Time	router	overhead %	search without overhead
467.00	39.00	0.08	428.00
476.00	30.00	0.06	446.00
407.00	32.00	0.08	375.00
455.00	31.00	0.07	424.00
482.00	36.00	0.07	446.00
452.00	34.00	0.08	418.00
406.00	31.00	0.08	375.00
483.00	33.00	0.07	450.00
453.00	33.00	0.07	420.00
475.00	34.00	0.07	441.00
411.00	33.00	0.08	378.00
474.00	31.00	0.07	443.00
482.00	31.00	0.06	451.00
475.00	32.00	0.07	443.00
404.00	31.00	0.08	373.00
495.00	32.00	0.06	463.00
468.00	35.00	0.07	433.00
478.00	31.00	0.06	447.00
437.00	31.00	0.07	406.00
539.00	39.00	0.07	500.00
431.00	31.00	0.07	400.00
479.00	33.00	0.07	446.00
506.00	31.00	0.06	475.00
570.00	33.00	0.06	537.00
476.00	33.00	0.07	443.00
522.00	32.00	0.06	490.00
474.00	32.00	0.07	442.00
489.00	32.00	0.07	457.00
542.00	32.00	0.06	510.00
476.00	31.00	0.07	445.00
472.80	32.63	0.07	440.17

Table D.1: Single Agent (BFS)(N = 3) Table D.2: Single Agent (BFS)(N = 5)

Search Time	router	overhead %	search without overhead
589.00	46.00	0.08	543.00
663.00	44.00	0.07	619.00
650.00	56.00	0.09	594.00
627.00	47.00	0.07	580.00
605.00	44.00	0.07	561.00
657.00	48.00	0.07	609.00
658.00	43.00	0.07	615.00
623.00	45.00	0.07	578.00
652.00	48.00	0.07	604.00
677.00	45.00	0.07	632.00
633.00	45.00	0.07	588.00
645.00	50.00	0.08	595.00
677.00	45.00	0.07	632.00
675.00	71.00	0.11	604.00
605.00	44.00	0.07	561.00
715.00	50.00	0.07	665.00
646.00	45.00	0.07	601.00
597.00	45.00	0.08	552.00
681.00	45.00	0.07	636.00
647.00	42.00	0.06	605.00
661.00	43.00	0.07	618.00
672.00	43.00	0.06	629.00
662.00	44.00	0.07	618.00
649.00	45.00	0.07	604.00
686.00	45.00	0.07	641.00
680.00	43.00	0.06	637.00
707.00	44.00	0.06	663.00
623.00	46.00	0.07	577.00
699.00	47.00	0.07	652.00
648.00	44.00	0.07	604.00
653.63	46.40	0.07	607.23

Table D.3: Single Agent (BFS)(N = 7)

Search Time	router	overhead %	search without overhead
820.00	56.00	0.07	764.00
814.00	57.00	0.07	757.00
946.00	65.00	0.07	881.00
847.00	64.00	0.08	783.00
811.00	58.00	0.07	753.00
879.00	58.00	0.07	821.00
891.00	57.00	0.06	834.00
833.00	60.00	0.07	773.00
896.00	55.00	0.06	841.00
843.00	63.00	0.07	780.00
836.00	58.00	0.07	778.00
886.00	56.00	0.06	830.00
900.00	55.00	0.06	845.00
839.00	57.00	0.07	782.00
905.00	60.00	0.07	845.00
925.00	59.00	0.06	866.00
889.00	63.00	0.07	826.00
919.00	56.00	0.06	863.00
928.00	60.00	0.06	868.00
840.00	61.00	0.07	779.00
1210.00	60.00	0.05	1150.00
899.00	60.00	0.07	839.00
917.00	65.00	0.07	852.00
912.00	62.00	0.07	850.00
943.00	57.00	0.06	886.00
914.00	80.00	0.09	834.00
924.00	57.00	0.06	867.00
948.00	58.00	0.06	890.00
954.00	59.00	0.06	895.00
915.00	60.00	0.07	855.00
899.43	59.87	0.07	839.57

Table D.4: Single Agent (BFS)(N = 9)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
979.00	72.00	0.07	907.00	1150.00	94.00	0.08	1056.00
1110.00	70.00	0.06	1040.00	1272.00	88.00	0.07	1184.00
1089.00	76.00	0.07	1013.00	1250.00	83.00	0.07	1167.00
1058.00	98.00	0.09	960.00	1251.00	83.00	0.07	1168.00
1048.00	75.00	0.07	973.00	1195.00	84.00	0.07	1111.00
1039.00	72.00	0.07	967.00	1221.00	86.00	0.07	1135.00
1064.00	71.00	0.07	993.00	1261.00	85.00	0.07	1176.00
1113.00	71.00	0.06	1042.00	1360.00	95.00	0.07	1265.00
1046.00	69.00	0.07	977.00	1302.00	82.00	0.06	1220.00
1112.00	74.00	0.07	1038.00	1279.00	84.00	0.07	1195.00
1050.00	70.00	0.07	980.00	1256.00	84.00	0.07	1172.00
1142.00	77.00	0.07	1065.00	1227.00	81.00	0.07	1146.00
1094.00	74.00	0.07	1020.00	1279.00	87.00	0.07	1192.00
991.00	75.00	0.08	916.00	1294.00	81.00	0.06	1213.00
1116.00	72.00	0.06	1044.00	1305.00	88.00	0.07	1217.00
1140.00	75.00	0.07	1065.00	1185.00	83.00	0.07	1102.00
1055.00	68.00	0.06	987.00	1267.00	87.00	0.07	1180.00
1096.00	70.00	0.06	1026.00	1239.00	82.00	0.07	1157.00
1027.00	97.00	0.09	930.00	1252.00	82.00	0.07	1170.00
1060.00	68.00	0.06	992.00	1249.00	81.00	0.06	1168.00
1062.00	68.00	0.06	994.00	1402.00	90.00	0.06	1312.00
1117.00	71.00	0.06	1046.00	1246.00	91.00	0.07	1155.00
1067.00	71.00	0.07	996.00	1275.00	91.00	0.07	1184.00
1109.00	71.00	0.06	1038.00	1179.00	86.00	0.07	1093.00
1078.00	68.00	0.06	1010.00	1232.00	87.00	0.07	1145.00
1059.00	69.00	0.07	990.00	1234.00	125.00	0.10	1109.00
1142.00	72.00	0.06	1070.00	1354.00	121.00	0.09	1233.00
1111.00	96.00	0.09	1015.00	1145.00	86.00	0.08	1059.00
1143.00	72.00	0.06	1071.00	1331.00	92.00	0.07	1239.00
1054.00	71.00	0.07	983.00	1183.00	94.00	0.08	1089.00
1079.03	74.10	0.07	1004.9	1255.83	88.77	0.07	1167.07

Table D.5: Single Agent (BFS)(N = 11) Table D.6: Single Agent (BFS)(N = 13)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
1396.00	107.00	0.08	1289.00	1523.00	116.00	0.08	1407.00
1443.00	96.00	0.07	1347.00	1662.00	119.00	0.07	1543.00
1507.00	105.00	0.07	1402.00	1707.00	117.00	0.07	1590.00
1433.00	95.00	0.07	1338.00	1811.00	116.00	0.06	1695.00
1394.00	101.00	0.07	1293.00	1544.00	113.00	0.07	1431.00
1411.00	97.00	0.07	1314.00	1623.00	121.00	0.07	1502.00
1431.00	99.00	0.07	1332.00	1578.00	116.00	0.07	1462.00
1500.00	137.00	0.09	1363.00	1721.00	119.00	0.07	1602.00
1463.00	108.00	0.07	1355.00	1689.00	121.00	0.07	1568.00
1483.00	100.00	0.07	1383.00	1656.00	113.00	0.07	1543.00
1462.00	100.00	0.07	1362.00	1519.00	119.00	0.08	1400.00
1411.00	96.00	0.07	1315.00	1597.00	125.00	0.08	1472.00
1403.00	97.00	0.07	1306.00	1598.00	113.00	0.07	1485.00
1479.00	99.00	0.07	1380.00	1691.00	106.00	0.06	1585.00
1513.00	128.00	0.08	1385.00	1739.00	110.00	0.06	1629.00
1500.00	106.00	0.07	1394.00	1624.00	113.00	0.07	1511.00
1392.00	95.00	0.07	1297.00	1564.00	152.00	0.10	1412.00
1405.00	99.00	0.07	1306.00	1827.00	113.00	0.06	1714.00
1389.00	100.00	0.07	1289.00	1573.00	111.00	0.07	1462.00
1446.00	119.00	0.08	1327.00	1845.00	118.00	0.06	1727.00
1483.00	91.00	0.06	1392.00	1653.00	106.00	0.06	1547.00
1417.00	98.00	0.07	1319.00	1581.00	115.00	0.07	1466.00
1413.00	114.00	0.08	1299.00	1588.00	113.00	0.07	1475.00
1448.00	96.00	0.07	1352.00	1631.00	113.00	0.07	1518.00
1426.00	93.00	0.07	1333.00	1716.00	112.00	0.07	1604.00
1498.00	102.00	0.07	1396.00	1750.00	136.00	0.08	1614.00
1434.00	102.00	0.07	1332.00	1662.00	111.00	0.07	1551.00
1395.00	98.00	0.07	1297.00	1631.00	110.00	0.07	1521.00
1498.00	91.00	0.06	1407.00	1739.00	111.00	0.06	1628.00
1369.00	99.00	0.07	1270.00	1713.00	115.00	0.07	1598.00
1441.40	102.27	0.07	1339.13	1658.50	116.43	0.07	1542.07

Table D.7: Single Agent (BFS)(N = 15) Table D.8: Single Agent (BFS)(N = 17)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
1811.00	143.00	0.08	1668.00	1852.00	234.00	0.13	1618.00
1777.00	127.00	0.07	1650.00	1812.00	198.00	0.11	1614.00
1809.00	134.00	0.07	1675.00	1784.00	152.00	0.09	1632.00
2006.00	147.00	0.07	1859.00	1915.00	167.00	0.09	1748.00
1696.00	126.00	0.07	1570.00	1827.00	147.00	0.08	1680.00
1792.00	131.00	0.07	1661.00	1848.00	142.00	0.08	1706.00
1903.00	125.00	0.07	1778.00	1876.00	155.00	0.08	1721.00
1720.00	123.00	0.07	1597.00	1798.00	153.00	0.09	1645.00
1843.00	130.00	0.07	1713.00	2040.00	204.00	0.10	1836.00
2052.00	136.00	0.07	1916.00	1993.00	180.00	0.09	1813.00
1726.00	124.00	0.07	1602.00	1932.00	207.00	0.11	1725.00
1769.00	125.00	0.07	1644.00	1872.00	169.00	0.09	1703.00
1826.00	125.00	0.07	1701.00	1832.00	184.00	0.10	1648.00
1739.00	125.00	0.07	1614.00	2273.00	408.00	0.18	1865.00
1980.00	177.00	0.09	1803.00	2123.00	161.00	0.08	1962.00
1977.00	135.00	0.07	1842.00	1847.00	133.00	0.07	1714.00
1751.00	126.00	0.07	1625.00	1877.00	151.00	0.08	1726.00
1784.00	126.00	0.07	1658.00	1827.00	133.00	0.07	1694.00
1828.00	125.00	0.07	1703.00	1835.00	137.00	0.07	1698.00
1843.00	121.00	0.07	1722.00	2072.00	135.00	0.07	1937.00
1982.00	129.00	0.07	1853.00	1764.00	135.00	0.08	1629.00
1840.00	121.00	0.07	1719.00	2016.00	181.00	0.09	1835.00
1863.00	157.00	0.08	1706.00	1882.00	135.00	0.07	1747.00
1810.00	120.00	0.07	1690.00	1868.00	128.00	0.07	1740.00
1926.00	119.00	0.06	1807.00	1853.00	133.00	0.07	1720.00
1909.00	121.00	0.06	1788.00	1859.00	157.00	0.08	1702.00
1908.00	154.00	0.08	1754.00	2030.00	133.00	0.07	1897.00
1845.00	122.00	0.07	1723.00	1862.00	127.00	0.07	1735.00
1981.00	122.00	0.06	1859.00	1921.00	132.00	0.07	1789.00
1917.00	122.00	0.06	1795.00	2055.00	213.00	0.10	1842.00
1853.77	130.60	0.07	1723.17	1911.50	167.47	0.09	1744.03

Table D.9: Single Agent (BFS)(N = 19) Table D.10: Single Agent (BFS)(N = 21)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
1787	181	0.1	1606	1764	170	0.1	1594
1753	145	0.08	1608	1974	196	0.1	1778
1927	175	0.09	1752	2018	168	0.08	1850
2002	161	0.08	1841	2250	266	0.12	1984
2022	165	0.08	1857	1980	203	0.1	1777
2002	146	0.07	1856	2077	178	0.09	1899
2054	161	0.08	1893	1942	153	0.08	1789
1764	147	0.08	1617	1851	149	0.08	1702
1961	159	0.08	1802	1952	149	0.08	1803
1900	163	0.09	1737	2009	148	0.07	1861
2061	149	0.07	1912	2268	210	0.09	2058
2117	166	0.08	1951	1897	160	0.08	1737
2059	143	0.07	1916	1965	177	0.09	1788
1967	219	0.11	1748	1966	161	0.08	1805
2250	164	0.07	2086	2064	152	0.07	1912
1962	139	0.07	1823	2372	299	0.13	2073
1792	143	0.08	1649	1931	147	0.08	1784
1885	141	0.07	1744	2006	158	0.08	1848
1916	141	0.07	1775	2031	160	0.08	1871
2075	140	0.07	1935	2019	155	0.08	1864
1799	149	0.08	1650	1957	146	0.07	1811
2094	154	0.07	1940	2052	169	0.08	1883
1982	197	0.1	1785	2340	189	0.08	2151
1992	159	0.08	1833	1981	144	0.07	1837
1925	189	0.1	1736	1861	156	0.08	1705
1940	153	0.08	1787	1975	147	0.07	1828
1789	140	0.08	1649	1945	148	0.08	1797
2279	162	0.07	2117	2203	203	0.09	2000
1958	151	0.08	1807	1997	154	0.08	1843
2053	167	0.08	1886	2084	175	0.08	1909
1968.90	158.97	0.08	1809.93	2024.37	173.00	0.08	1851.37

Table D.11: Single Agent (BFS)(N = 23) Table D.12: Single Agent (BFS)(N = 25)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
1796	181	0.1	1615	1829	160	0.09	1669
1998	213	0.11	1785	2316	258	0.11	2058
2089	181	0.09	1908	2258	220	0.1	2038
2122	189	0.09	1933	1910	200	0.1	1710
2153	206	0.1	1947	2082	241	0.12	1841
2030	162	0.08	1868	1924	170	0.09	1754
1972	161	0.08	1811	2025	164	0.08	1861
1940	162	0.08	1778	2157	177	0.08	1980
1873	149	0.08	1724	2078	182	0.09	1896
2211	194	0.09	2017	2608	262	0.1	2346
2106	154	0.07	1952	1928	161	0.08	1767
2318	245	0.11	2073	1868	229	0.12	1639
2000	165	0.08	1835	2315	178	0.08	2137
2599	245	0.09	2354	2050	170	0.08	1880
1850	148	0.08	1702	1947	161	0.08	1786
1847	153	0.08	1694	2414	224	0.09	2190
1927	148	0.08	1779	1921	167	0.09	1754
2078	151	0.07	1927	2084	156	0.07	1928
1999	158	0.08	1841	2019	196	0.1	1823
2298	180	0.08	2118	1982	176	0.09	1806
2276	207	0.09	2069	2116	187	0.09	1929
2044	158	0.08	1886	2477	206	0.08	2271
1807	161	0.09	1646	2044	163	0.08	1881
1924	151	0.08	1773	2159	185	0.09	1974
2026	151	0.07	1875	2123	237	0.11	1886
2176	196	0.09	1980	1906	170	0.09	1736
1822	152	0.08	1670	2588	237	0.09	2351
2275	176	0.08	2099	2163	160	0.07	2003
2058	150	0.07	1908	2119	237	0.11	1882
1934	158	0.08	1776	1935	159	0.08	1776
2051.60	173.50	0.08	1878.10	2111.50	193.10	0.09	1918.40

Table D.13: Single Agent (BFS)(N = 27) Table D.14: Single Agent (BFS)(N = 28)

Appendix E

Single Agent With Variable Network Size (DFS)

Search Time	router	overhead %	search without overhead
258	19	0.07	239
240	17	0.07	223
321	18	0.06	303
266	18	0.07	248
267	18	0.07	249
317	19	0.06	298
265	18	0.07	247
306	18	0.06	288
264	19	0.07	245
312	19	0.06	293
261	19	0.07	242
316	18	0.06	298
263	18	0.07	245
317	19	0.06	298
269	25	0.09	244
312	21	0.07	291
265	19	0.07	246
315	18	0.06	297
260	18	0.07	242
312	18	0.06	294
273	19	0.07	254
320	19	0.06	301
278	19	0.07	259
317	18	0.06	299
281	18	0.06	263
312	20	0.06	292
315	27	0.09	288
305	20	0.07	285
267	18	0.07	249
316	20	0.06	296
289.67	19.13	0.07	270.53

Search Time	router	overhead %	search without overhead
546	66	0.12	480
592	49	0.08	543
562	37	0.07	525
515	38	0.07	477
549	38	0.07	511
585	44	0.08	541
549	37	0.07	512
494	43	0.09	451
582	39	0.07	543
557	37	0.07	520
549	40	0.07	509
509	38	0.07	471
554	37	0.07	517
599	37	0.06	562
566	39	0.07	527
512	39	0.08	473
576	37	0.06	539
591	38	0.06	553
567	37	0.07	530
508	38	0.07	470
652	37	0.06	615
527	39	0.07	488
584	37	0.06	547
576	37	0.06	539
594	42	0.07	552
546	43	0.08	503
608	39	0.06	569
539	41	0.08	498
553	38	0.07	515
621	38	0.06	583
562.07	39.97	0.07	522.10

Table E.1: Single Agent (DFS)(N = 3) Table E.2: Single Agent (DFS)(N = 5)

Search Time	router	overhead %	search without overhead
605	47	0.08	558
603	45	0.07	558
623	45	0.07	578
670	45	0.07	625
696	59	0.08	637
621	43	0.07	578
599	45	0.08	554
689	49	0.07	640
693	48	0.07	645
627	44	0.07	583
672	45	0.07	627
631	46	0.07	585
682	48	0.07	634
686	43	0.06	643
601	45	0.07	556
669	44	0.07	625
678	45	0.07	633
662	45	0.07	617
620	45	0.07	575
689	50	0.07	639
574	49	0.09	525
655	46	0.07	609
667	48	0.07	619
617	49	0.08	568
708	46	0.06	662
643	57	0.09	586
682	45	0.07	637
620	51	0.08	569
734	47	0.06	687
616	49	0.08	567
651.07	47.10	0.07	603.97

Table E.3: Single Agent (DFS)(N = 7)

Search Time	router	overhead %	search without overhead
858	58	0.07	800
894	59	0.07	835
949	59	0.06	890
890	58	0.07	832
912	56	0.06	856
990	56	0.06	934
906	57	0.06	849
935	59	0.06	876
965	58	0.06	907
1039	58	0.06	981
864	56	0.06	808
939	56	0.06	883
893	54	0.06	839
913	62	0.07	851
968	59	0.06	909
930	60	0.06	870
984	56	0.06	928
988	56	0.06	932
930	61	0.07	869
866	56	0.06	810
925	59	0.06	866
971	64	0.07	907
902	59	0.07	843
913	60	0.07	853
950	59	0.06	891
908	62	0.07	846
952	62	0.07	890
1018	63	0.06	955
932	56	0.06	876
885	67	0.08	818
932.30	58.83	0.06	873.47

Table E.4: Single Agent (DFS)(N = 9)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
1099	87	0.08	1012	1261	93	0.07	1168
997	70	0.07	927	1266	95	0.08	1171
1052	77	0.07	975	1209	83	0.07	1126
1072	82	0.08	990	1265	88	0.07	1177
1067	82	0.08	985	1230	85	0.07	1145
1022	68	0.07	954	1243	91	0.07	1152
1073	72	0.07	1001	1225	84	0.07	1141
1021	104	0.1	917	1254	86	0.07	1168
1128	73	0.06	1055	1164	83	0.07	1081
1020	70	0.07	950	1358	95	0.07	1263
1056	103	0.1	953	1301	80	0.06	1221
1036	72	0.07	964	1229	83	0.07	1146
1090	70	0.06	1020	1193	82	0.07	1111
1073	75	0.07	998	1327	87	0.07	1240
1029	68	0.07	961	1174	82	0.07	1092
1075	69	0.06	1006	1273	88	0.07	1185
1076	68	0.06	1008	1222	93	0.08	1129
1027	69	0.07	958	1266	84	0.07	1182
1066	68	0.06	998	1306	103	0.08	1203
1151	80	0.07	1071	1305	88	0.07	1217
1028	70	0.07	958	1194	108	0.09	1086
1097	70	0.06	1027	1346	124	0.09	1222
1061	69	0.07	992	1242	81	0.07	1161
1074	71	0.07	1003	1229	84	0.07	1145
1058	69	0.07	989	1257	81	0.06	1176
1150	74	0.06	1076	1244	97	0.08	1147
1070	74	0.07	996	1264	84	0.07	1180
1123	75	0.07	1048	1371	84	0.06	1287
1135	74	0.07	1061	1202	81	0.07	1121
1096	83	0.08	1013	1242	85	0.07	1157
1070.7	75.20	0.07	995.53	1255.40	88.73	0.07	1166.67

Table E.5: Single Agent (DFS)(N = 11) Table E.6: Single Agent (DFS)(N = 13)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
1423	117	0.08	1306	1450	109	0.08	1341
1325	97	0.07	1228	1572	116	0.07	1456
1533	112	0.07	1421	1616	126	0.08	1490
1451	99	0.07	1352	1601	132	0.08	1469
1474	107	0.07	1367	1541	120	0.08	1421
1364	100	0.07	1264	1725	163	0.09	1562
1425	96	0.07	1329	1501	112	0.07	1389
1390	93	0.07	1297	1559	122	0.08	1437
1634	119	0.07	1515	1717	159	0.09	1558
1502	95	0.06	1407	1687	113	0.07	1574
1397	95	0.07	1302	1583	142	0.09	1441
1490	95	0.06	1395	1790	123	0.07	1667
1351	98	0.07	1253	1526	112	0.07	1414
1308	96	0.07	1212	1547	111	0.07	1436
1611	110	0.07	1501	1701	110	0.06	1591
1358	92	0.07	1266	1628	112	0.07	1516
1415	93	0.07	1322	1618	122	0.08	1496
1367	97	0.07	1270	1666	132	0.08	1534
1391	96	0.07	1295	1725	116	0.07	1609
1442	110	0.08	1332	1668	137	0.08	1531
1549	100	0.06	1449	1701	107	0.06	1594
1406	100	0.07	1306	1729	113	0.07	1616
1411	100	0.07	1311	1610	111	0.07	1499
1393	93	0.07	1300	1716	106	0.06	1610
1520	95	0.06	1425	1606	108	0.07	1498
1595	99	0.06	1496	1662	123	0.07	1539
1523	101	0.07	1422	1730	109	0.06	1621
1389	100	0.07	1289	1732	112	0.06	1620
1502	98	0.07	1404	1675	116	0.07	1559
1460	103	0.07	1357	1628	108	0.07	1520
1446.63	100.20	0.07	1346.43	1640.33	120.07	0.07	1520.27

Table E.7: Single Agent (DFS)(N = 15) Table E.8: Single Agent (DFS)(N = 17)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
1689	137	0.08	1552	1832	152	0.08	1680
1755	170	0.1	1585	1743	156	0.09	1587
1968	134	0.07	1834	2008	148	0.07	1860
1757	127	0.07	1630	1943	170	0.09	1773
1700	138	0.08	1562	1893	155	0.08	1738
1885	138	0.07	1747	1871	152	0.08	1719
1749	137	0.08	1612	1862	140	0.08	1722
1895	129	0.07	1766	1957	143	0.07	1814
1921	148	0.08	1773	1911	142	0.07	1769
1846	177	0.1	1669	1920	172	0.09	1748
4728	170	0.04	4558	1821	159	0.09	1662
1768	124	0.07	1644	1898	143	0.08	1755
1829	132	0.07	1697	1822	143	0.08	1679
1913	172	0.09	1741	2131	158	0.07	1973
1828	198	0.11	1630	1953	146	0.07	1807
1848	172	0.09	1676	1817	139	0.08	1678
1790	141	0.08	1649	1848	144	0.08	1704
1787	127	0.07	1660	1879	149	0.08	1730
1939	126	0.06	1813	1936	144	0.07	1792
1805	115	0.06	1690	1890	168	0.09	1722
1855	130	0.07	1725	1886	140	0.07	1746
1736	118	0.07	1618	2059	154	0.07	1905
1832	129	0.07	1703	1871	155	0.08	1716
1825	122	0.07	1703	1898	185	0.1	1713
1929	129	0.07	1800	1980	134	0.07	1846
1844	124	0.07	1720	1856	144	0.08	1712
1875	130	0.07	1745	1805	172	0.1	1633
2339	127	0.05	2212	1954	146	0.07	1808
1870	129	0.07	1741	1853	141	0.08	1712
1937	127	0.07	1810	1949	136	0.07	1813
1948.07	139.23	0.07	1808.83	1901.53	151.00	0.08	1750.53

Table E.9: Single Agent (DFS)(N = 19) Table E.10: Single Agent (DFS)(N = 21)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
1827	192	0.11	1635	1816	177	0.1	1639
2056	190	0.09	1866	2112	171	0.08	1941
1826	150	0.08	1676	2024	173	0.09	1851
1950	193	0.1	1757	2121	196	0.09	1925
1935	156	0.08	1779	2066	170	0.08	1896
2410	172	0.07	2238	2065	175	0.08	1890
2029	155	0.08	1874	2024	166	0.08	1858
2024	205	0.1	1819	2116	183	0.09	1933
1935	153	0.08	1782	1908	171	0.09	1737
1885	148	0.08	1737	1891	149	0.08	1742
1845	149	0.08	1696	2096	165	0.08	1931
1959	172	0.09	1787	1965	161	0.08	1804
2328	211	0.09	2117	2233	217	0.1	2016
1929	168	0.09	1761	2232	173	0.08	2059
1907	153	0.08	1754	1904	154	0.08	1750
1932	142	0.07	1790	1878	155	0.08	1723
1771	147	0.08	1624	2085	175	0.08	1910
1866	173	0.09	1693	1888	159	0.08	1729
2026	148	0.07	1878	2179	181	0.08	1998
2021	158	0.08	1863	1836	148	0.08	1688
2018	157	0.08	1861	2189	162	0.07	2027
2245	141	0.06	2104	1897	160	0.08	1737
1831	142	0.08	1689	2113	163	0.08	1950
2031	149	0.07	1882	2056	163	0.08	1893
2003	150	0.07	1853	1907	153	0.08	1754
1965	151	0.08	1814	2005	169	0.08	1836
1883	140	0.07	1743	1865	191	0.1	1674
1873	153	0.08	1720	2104	166	0.08	1938
1996	185	0.09	1811	2070	156	0.08	1914
2095	159	0.08	1936	1936	166	0.09	1770
1980.03	162.07	0.08	1817.97	2019.37	168.93	0.08	1850.43

Table E.11: Single Agent (DFS)(N = 23) Table E.12: Single Agent (DFS)(N = 25)

Search Time	router	overhead %	search without overhead	Search Time	router	overhead %	search without overhead
1858	179	0.1	1679	1875	167	0.09	1708
2191	171	0.08	2020	2163	222	0.1	1941
1974	180	0.09	1794	2208	198	0.09	2010
2198	205	0.09	1993	2169	241	0.11	1928
2150	199	0.09	1951	2009	174	0.09	1835
2138	168	0.08	1970	1970	174	0.09	1796
1928	167	0.09	1761	2090	170	0.08	1920
2676	269	0.1	2407	1987	199	0.1	1788
1934	175	0.09	1759	1975	182	0.09	1793
1918	156	0.08	1762	2119	171	0.08	1948
2087	292	0.14	1795	2356	246	0.1	2110
2080	176	0.08	1904	2257	171	0.08	2086
2140	208	0.1	1932	2067	156	0.08	1911
2347	161	0.07	2186	2166	179	0.08	1987
1993	189	0.09	1804	2176	221	0.1	1955
1974	160	0.08	1814	1993	172	0.09	1821
1859	168	0.09	1691	2198	172	0.08	2026
2181	162	0.07	2019	1996	166	0.08	1830
2124	168	0.08	1956	2055	164	0.08	1891
2057	168	0.08	1889	2209	176	0.08	2033
2109	170	0.08	1939	2344	213	0.09	2131
2268	187	0.08	2081	1988	187	0.09	1801
2084	172	0.08	1912	2131	176	0.08	1955
2068	154	0.07	1914	2152	162	0.08	1990
2006	177	0.09	1829	1962	165	0.08	1797
2357	252	0.11	2105	2003	198	0.1	1805
2059	164	0.08	1895	2358	194	0.08	2164
2168	168	0.08	2000	2109	179	0.08	1930
2002	193	0.1	1809	2257	172	0.08	2085
2132	162	0.08	1970	2116	171	0.08	1945
2102.00	184.00	0.09	1918.00	2115.27	184.60	0.09	1930.67

Table E.13: Single Agent (DFS)(N = 27) Table E.14: Single Agent (DFS)(N = 28)

Appendix F

Average Execution On Different Network Size And Algorithm

Network size	Search Time	router	white board	black board	total	overhead percentage	search without overhead
3	389.70	12.93	6.60	94.83	114.37	0.29	275.33
5	718.67	21.77	16.10	191.83	229.70	0.32	488.97
7	705.73	23.50	24.53	190.47	238.50	0.34	467.23
9	999.60	30.03	27.33	267.87	325.23	0.32	674.37
11	991.40	30.00	32.53	262.67	325.20	0.33	666.20
13	1028.70	37.10	37.20	284.33	358.63	0.35	670.07
15	1128.73	30.27	42.33	292.23	364.83	0.34	763.90
17	1362.93	38.83	45.70	366.33	450.87	0.33	912.07
19	1310.48	37.38	49.59	348.69	435.66	0.33	874.83
21	1319.57	40.37	57.50	324.93	422.80	0.32	896.77
23	1250.60	37.17	52.33	300.73	390.23	0.31	860.37
25	1395.20	37.47	55.13	336.07	428.67	0.30	966.53
27	1385.03	37.27	58.57	331.73	427.57	0.31	957.47
28	1308.37	37.10	55.30	280.67	373.07	0.29	935.30

Table F.1: Average Execution Time with Various Network (Remote Flooding)

Network size	Search Time	router	white board	black board	total	overhead percentage	search without overhead
3	385.80	12.63	6.53	71.97	91.13	0.23	294.67
5	718.27	22.80	18.37	122.17	163.33	0.23	554.93
7	756.40	24.90	25.50	185.40	235.80	0.31	520.60
9	1022.70	32.43	27.80	197.60	257.83	0.25	764.87
11	994.87	33.83	31.40	213.00	278.23	0.28	716.63
13	1051.33	30.90	36.00	273.13	340.03	0.32	711.30
15	1107.37	37.03	44.13	320.37	401.53	0.36	705.83
17	1288.00	39.87	39.70	292.87	372.43	0.29	915.57
19	1319.93	38.30	43.37	325.40	407.07	0.31	912.87
21	1342.27	40.93	45.50	307.83	394.27	0.29	948.00
23	1440.27	34.67	57.20	359.10	450.97	0.31	989.30
25	1512.53	36.43	57.57	466.07	560.07	0.37	952.47
27	1655.33	37.00	57.83	583.87	678.70	0.40	976.63
28	1701.90	35.33	60.03	622.93	718.30	0.42	983.60

Table F.2: Average Execution Time with Various Network (Local Flooding)

Network size	Search Time	router	white board	black board	total	overhead percentage	search without overhead
3	359.60	14.60	6.33	14.67	35.60	0.10	324.00
5	729.23	23.55	17.93	18.18	59.67	0.08	669.57
7	699.40	26.47	29.17	23.25	78.88	0.11	620.52
9	1074.80	35.32	32.46	24.44	92.22	0.09	982.58
11	1037.93	35.46	36.16	30.52	102.13	0.10	935.80
13	1095.83	38.10	43.70	29.77	111.57	0.10	984.27
15	1056.97	42.83	48.77	36.97	128.57	0.12	928.40
17	1467.39	47.42	50.87	36.87	135.16	0.09	1332.23
19	1402.10	48.47	51.03	35.43	134.93	0.10	1267.17
21	1482.30	46.27	52.57	39.50	138.33	0.09	1343.97
23	1446.33	48.90	61.57	39.23	149.70	0.10	1296.63
25	1471.83	48.93	58.27	38.47	145.67	0.10	1326.17
27	1513.97	53.83	62.23	39.07	155.13	0.10	1358.83

Table F.3: Average Execution Time with Various Network (Regular Flooding)

Network size	Search Time	router	overhead percentage	search without overhead
3	289.67	19.13	0.07	270.53
5	562.07	39.97	0.07	522.10
7	651.07	47.10	0.07	603.97
9	932.30	58.83	0.06	873.47
11	1070.73	75.20	0.07	995.53
13	1255.40	88.73	0.07	1166.67
15	1446.63	100.20	0.07	1346.43
17	1640.33	120.07	0.07	1520.27
19	1948.07	139.23	0.07	1808.83
21	1901.53	151.00	0.08	1750.53
23	1980.03	162.07	0.08	1817.97
25	2019.37	168.93	0.08	1850.43
27	2102.00	184.00	0.09	1918.00
28	2115.27	184.60	0.09	1930.67

Table F.4: Average Execution Time with Various Network (DFS)

Network size	Search Time	router	overhead percentage	search without overhead
3	289.03	19.63	0.07	269.40
5	472.80	32.63	0.07	440.17
7	653.63	46.40	0.07	607.23
9	899.43	59.87	0.07	839.57
11	1079.03	74.10	0.07	1004.93
13	1255.83	88.77	0.07	1167.07
15	1441.40	102.27	0.07	1339.13
17	1658.50	116.43	0.07	1542.07
19	1853.77	130.60	0.07	1723.17
21	1911.50	167.47	0.09	1744.03
23	1968.90	158.97	0.08	1809.93
25	2024.37	173.00	0.08	1851.37
27	2051.60	173.50	0.08	1878.10
28	2111.50	193.10	0.09	1918.40

Table F.5: Average Execution Time with Various Network (BFS)

Appendix G

Average Execution Time With Various Number Of Neighbors And Itinerary Algorithms

Number of Neighbors	Search Time	router	white board	black board	total	overhead percentage	search without overhead
2	1358.633	40.467	47.933	338.233	426.633	0.313	932.000
4	1188.567	26.167	43.000	200.000	269.167	0.226	919.400
6	1695.167	24.000	27.567	201.467	253.033	0.149	1442.133
8	2112.700	23.667	19.900	176.067	219.633	0.103	1893.067
10	2412.867	23.533	15.033	186.333	224.900	0.092	2187.967
12	2495.533	22.267	12.233	205.500	240.000	0.095	2255.533
14	2422.767	22.500	9.300	169.100	200.900	0.083	2221.867
16	2480.200	23.367	7.567	163.433	194.367	0.078	2285.833
18	2464.933	25.867	6.500	142.833	175.200	0.070	2289.733
19	2494.567	17.533	11.600	146.267	175.400	0.070	2319.167

Table G.1: Average Execution Time with Various Number of Neighbors (Remote Flooding)

Number of Neighbors	Search Time	router	white board	black board	total	overhead percentage	search without overhead
2	1228.000	39.788	51.362	289.023	380.177	0.310	847.867
4	1227.933	23.333	43.533	403.567	470.433	0.382	757.500
6	1581.700	24.833	29.133	178.700	232.667	0.148	1349.033
8	2066.467	28.200	20.433	150.633	199.267	0.097	1867.200
10	2374.767	23.833	16.300	125.167	165.300	0.069	2209.467
12	2507.000	23.133	16.067	132.800	172.000	0.068	2335.000
14	2425.067	24.467	12.600	124.067	161.133	0.066	2263.933
16	2500.633	29.067	11.067	116.200	156.333	0.061	2344.300
18	2477.200	29.233	6.333	146.067	181.633	0.073	2295.567
19	2562.100	13.667	6.867	134.467	155.000	0.060	2407.100

Table G.2: Average Execution Time with Various Number of Neighbors (Local Flooding)

Number of Neighbors	Search Time	router	white board	black board	total	overhead percentage	search without overhead
2	1499.933	52.733	65.667	42.967	161.367	0.107	1338.567
4	1216.900	30.400	47.200	25.733	103.333	0.085	1113.567
6	1834.600	30.167	31.967	19.967	82.100	0.043	1752.500
8	2283.567	25.500	22.367	16.300	64.167	0.029	2219.400
10	2673.233	23.667	19.000	14.000	56.667	0.021	2616.567
12	3067.967	28.367	17.300	15.300	60.967	0.020	3007.000
14	2988.067	23.400	11.633	13.067	48.100	0.017	2939.967
16	3301.067	28.133	8.300	15.700	52.133	0.016	3248.933
18	3631.867	34.600	18.000	28.367	80.967	0.022	3550.900
19	3683.414	21.655	10.793	36.621	69.069	0.019	3614.345

Table G.3: Average Execution Time with Various Number of Neighbors (Regular Flooding)

Number Of Neighbors	Search Time	router	overhead percentage	search without overhead
2	1799.111	154.583	0.083	1644.528
4	1769.886	143.829	0.081	1626.057
6	1827.622	146.189	0.080	1681.432
8	1964.714	137.714	0.071	1827.000
10	1964.714	137.714	0.071	1827.000
12	1754.056	154.556	0.087	1599.500
14	1784.250	134.833	0.076	1649.417
16	1789.722	138.333	0.078	1651.389
18	1822.944	136.861	0.076	1686.083
19	1832.438	134.563	0.074	1697.875

Table G.4: Average Execution Time with Various Number of Neighbors (Single Agent (DFS))

Number Of Neighbors	Search Time	router	overhead percentage	search without overhead
2	1956.520	142.880	0.073	1813.640
4	1778.200	138.460	0.079	1639.740
6	1887.110	159.710	0.083	1727.400
8	1887.310	136.630	0.073	1750.690
10	1768.970	146.490	0.083	1622.490
12	1781.860	133.500	0.075	1648.360
14	1806.460	136.240	0.076	1670.220
16	1817.740	132.670	0.074	1685.080
18	1853.000	130.430	0.071	1722.570
19	1864.760	133.670	0.072	1731.090

Table G.5: Average Execution Time with Various Number of Neighbors (Single Agent (BFS))

Appendix H

Single And Multiple Agents With Different Topology And Various Network Size

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1,536.00	43.00	31.00	321.00	395.00	0.26	1,141.00
2	1,546.00	40.00	35.00	315.00	390.00	0.25	1,156.00
3	1,608.00	173.00	34.00	324.00	531.00	0.33	1,077.00
4	1,518.00	40.00	33.00	392.00	465.00	0.31	1,053.00
5	1,562.00	41.00	32.00	306.00	379.00	0.24	1,183.00
6	1,520.00	38.00	32.00	308.00	378.00	0.25	1,142.00
7	1,563.00	40.00	31.00	308.00	379.00	0.24	1,184.00
8	1,538.00	97.00	31.00	374.00	502.00	0.33	1,036.00
9	1,563.00	38.00	34.00	311.00	383.00	0.25	1,180.00
10	1,562.00	38.00	33.00	317.00	388.00	0.25	1,174.00
11	1,532.00	41.00	31.00	305.00	377.00	0.25	1,155.00
12	1,572.00	42.00	58.00	364.00	464.00	0.30	1,108.00
13	1,520.00	40.00	34.00	315.00	389.00	0.26	1,131.00
14	1,586.00	39.00	33.00	298.00	370.00	0.23	1,216.00
15	1,599.00	40.00	31.00	307.00	378.00	0.24	1,221.00
16	1,555.00	38.00	45.00	369.00	452.00	0.29	1,103.00
17	1,510.00	38.00	33.00	299.00	370.00	0.25	1,140.00
18	1,578.00	40.00	36.00	304.00	380.00	0.24	1,198.00
19	1,608.00	41.00	31.00	361.00	433.00	0.27	1,175.00
20	1,513.00	41.00	32.00	312.00	385.00	0.25	1,128.00
21	1,608.00	40.00	32.00	432.00	504.00	0.31	1,104.00
22	1,525.00	38.00	32.00	308.00	378.00	0.25	1,147.00
23	1,551.00	74.00	32.00	314.00	420.00	0.27	1,131.00
24	1,614.00	40.00	33.00	319.00	392.00	0.24	1,222.00
25	1,521.00	39.00	33.00	319.00	391.00	0.26	1,130.00
26	1,523.00	40.00	33.00	333.00	406.00	0.27	1,117.00
27	1,510.00	42.00	32.00	303.00	377.00	0.25	1,133.00
28	1,610.00	66.00	32.00	309.00	407.00	0.25	1,203.00
29	1,531.00	40.00	33.00	326.00	399.00	0.26	1,132.00
30	1,538.00	42.00	33.00	342.00	417.00	0.27	1,121.00
Average	1,554.00	48.30	33.83	327.17	409.30	0.26	1,144.70

Table H.1: Bidirectional Ring with Flooding (N = 16)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	2743.00	109.00	104.00	1081.00	1294.00	0.47	1449.00
2	2541.00	111.00	110.00	918.00	1139.00	0.45	1402.00
3	2499.00	106.00	107.00	914.00	1127.00	0.45	1372.00
4	3356.00	138.00	344.00	1188.00	1670.00	0.50	1686.00
5	2557.00	107.00	102.00	949.00	1158.00	0.45	1399.00
6	2529.00	109.00	98.00	895.00	1102.00	0.44	1427.00
7	2957.00	113.00	100.00	1269.00	1482.00	0.50	1475.00
8	2470.00	106.00	127.00	850.00	1083.00	0.44	1387.00
9	2554.00	106.00	102.00	931.00	1139.00	0.45	1415.00
10	2999.00	109.00	104.00	1286.00	1499.00	0.50	1500.00
11	2544.00	127.00	100.00	890.00	1117.00	0.44	1427.00
12	2645.00	147.00	102.00	1013.00	1262.00	0.48	1383.00
13	2803.00	109.00	105.00	1169.00	1383.00	0.49	1420.00
14	2587.00	111.00	97.00	967.00	1175.00	0.45	1412.00
15	2729.00	122.00	127.00	912.00	1161.00	0.43	1568.00
16	2736.00	107.00	105.00	1090.00	1302.00	0.48	1434.00
17	2672.00	125.00	105.00	907.00	1137.00	0.43	1535.00
18	2689.00	107.00	106.00	969.00	1182.00	0.44	1507.00
19	2615.00	106.00	99.00	935.00	1140.00	0.44	1475.00
20	2851.00	139.00	175.00	948.00	1262.00	0.44	1589.00
21	2701.00	138.00	168.00	924.00	1230.00	0.46	1471.00
22	2755.00	108.00	100.00	1015.00	1223.00	0.44	1532.00
23	2722.00	113.00	137.00	900.00	1150.00	0.42	1572.00
24	2699.00	109.00	101.00	1012.00	1222.00	0.45	1477.00
25	2539.00	105.00	106.00	932.00	1143.00	0.45	1396.00
26	2792.00	106.00	102.00	1078.00	1286.00	0.46	1506.00
27	2718.00	107.00	106.00	959.00	1172.00	0.43	1546.00
28	2655.00	107.00	100.00	999.00	1206.00	0.45	1449.00
29	2728.00	106.00	108.00	1047.00	1261.00	0.46	1467.00
30	2785.00	107.00	102.00	1051.00	1260.00	0.45	1525.00
Average	2705.67	114.00	118.30	999.93	1232.23	0.45	1473.43

Table H.2: Unidirectional Ring with Flooding ($N = 16$)

Test Num	Search Time	router	white board	black board	total	overhead percentage	search without overhead
1	1010.00	79.00	67.00	200.00	346.00	0.34	664.00
2	937.00	28.00	52.00	228.00	308.00	0.33	629.00
3	952.00	24.00	41.00	148.00	213.00	0.22	739.00
4	952.00	25.00	50.00	203.00	278.00	0.29	674.00
5	931.00	28.00	54.00	209.00	291.00	0.31	640.00
6	940.00	25.00	48.00	231.00	304.00	0.32	636.00
7	953.00	25.00	48.00	197.00	270.00	0.28	683.00
8	1017.00	27.00	64.00	263.00	354.00	0.35	663.00
9	933.00	29.00	54.00	224.00	307.00	0.33	626.00
10	1138.00	24.00	56.00	333.00	413.00	0.36	725.00
11	978.00	26.00	55.00	211.00	292.00	0.30	686.00
12	977.00	25.00	46.00	219.00	290.00	0.30	687.00
13	1021.00	24.00	48.00	206.00	278.00	0.27	743.00
14	1008.00	25.00	56.00	188.00	269.00	0.27	739.00
15	1083.00	24.00	49.00	182.00	255.00	0.24	828.00
16	1097.00	25.00	55.00	179.00	259.00	0.24	838.00
17	1098.00	26.00	54.00	195.00	275.00	0.25	823.00
18	1064.00	25.00	53.00	179.00	257.00	0.24	807.00
19	1115.00	25.00	45.00	191.00	261.00	0.23	854.00
20	1217.00	67.00	57.00	369.00	493.00	0.41	724.00
21	992.00	23.00	61.00	320.00	404.00	0.41	588.00
22	973.00	25.00	52.00	197.00	274.00	0.28	699.00
23	985.00	25.00	64.00	223.00	312.00	0.32	673.00
24	1020.00	25.00	60.00	213.00	298.00	0.29	722.00
25	1109.00	24.00	61.00	295.00	380.00	0.34	729.00
26	1078.00	26.00	60.00	197.00	283.00	0.26	795.00
27	1123.00	25.00	54.00	308.00	387.00	0.34	736.00
28	1217.00	26.00	58.00	375.00	459.00	0.38	758.00
29	1166.00	69.00	52.00	252.00	373.00	0.32	793.00
30	1051.00	24.00	59.00	284.00	367.00	0.35	684.00
Average	1037.83	29.93	54.43	233.97	318.33	0.31	719.50

Table H.3: Hypercube with Flooding (N = 16)

Search Time	router	overhead percentage	search without overhead	Search Time	router	overhead %	search without overhead
1378	114	0.08	1264	1370	119	0.09	1251
1683	112	0.07	1571	1626	150	0.09	1476
1412	126	0.09	1286	1409	112	0.08	1297
1368	105	0.08	1263	1425	116	0.08	1309
1389	126	0.09	1263	1439	127	0.09	1312
1379	109	0.08	1270	1417	111	0.08	1306
1737	154	0.09	1583	1864	300	0.16	1564
1403	103	0.07	1300	1413	124	0.09	1289
1394	100	0.07	1294	1436	116	0.08	1320
1368	107	0.08	1261	1412	135	0.1	1277
1622	108	0.07	1514	1669	120	0.07	1549
1498	104	0.07	1394	1524	215	0.14	1309
1379	107	0.08	1272	1438	170	0.12	1268
1380	106	0.08	1274	1414	130	0.09	1284
1626	106	0.07	1520	1695	130	0.08	1565
1512	129	0.09	1383	1536	115	0.07	1421
1408	107	0.08	1301	1406	116	0.08	1290
1503	107	0.07	1396	1448	119	0.08	1329
1519	111	0.07	1408	1687	188	0.11	1499
1514	109	0.07	1405	1492	137	0.09	1355
1448	107	0.07	1341	1433	110	0.08	1323
1465	110	0.08	1355	1572	128	0.08	1444
1502	107	0.07	1395	1513	118	0.08	1395
1613	103	0.06	1510	1493	150	0.1	1343
1450	108	0.07	1342	1555	130	0.08	1425
1479	111	0.08	1368	1541	118	0.08	1423
1570	119	0.08	1451	1551	200	0.13	1351
1493	106	0.07	1387	1635	179	0.11	1456
1514	103	0.07	1411	1510	118	0.08	1392
1507	108	0.07	1399	1494	119	0.08	1375
1483.77	111.07	0.08	1372.70	1513.90	140.67	0.09	1373.23

Table H.4: Bidirectional Ring with Single Agent (N = 16) Table H.5: Unidirectional Ring with Single Agent (N = 16)

Test Num	Search Time	router	overhead percentage	search without overhead
1	1390.00	107.00	0.08	1283.00
2	1714.00	136.00	0.08	1578.00
3	1482.00	111.00	0.07	1371.00
4	1649.00	109.00	0.07	1540.00
5	1421.00	138.00	0.10	1283.00
6	1646.00	112.00	0.07	1534.00
7	1519.00	132.00	0.09	1387.00
8	1640.00	117.00	0.07	1523.00
9	1438.00	107.00	0.07	1331.00
10	1639.00	113.00	0.07	1526.00
11	1484.00	110.00	0.07	1374.00
12	1662.00	117.00	0.07	1545.00
13	1462.00	108.00	0.07	1354.00
14	1669.00	107.00	0.06	1562.00
15	1565.00	100.00	0.06	1465.00
16	1626.00	115.00	0.07	1511.00
17	1504.00	111.00	0.07	1393.00
18	1608.00	110.00	0.07	1498.00
19	1610.00	106.00	0.07	1504.00
20	1708.00	101.00	0.06	1607.00
21	1509.00	109.00	0.07	1400.00
22	1649.00	109.00	0.07	1540.00
23	1589.00	110.00	0.07	1479.00
24	1630.00	107.00	0.07	1523.00
25	1598.00	119.00	0.07	1479.00
26	1697.00	136.00	0.08	1561.00
27	1654.00	136.00	0.08	1518.00
28	1597.00	108.00	0.07	1489.00
29	2071.00	111.00	0.05	1960.00
30	1641.00	109.00	0.07	1532.00

Table H.6: Hypercube with Single Agent ($N = 16$)

Appendix I

**Execution Time For Both AGLET
And TACOMA Moving From Host
A To Host B And Return Back To
Host A**

Test Num	the required time for TACOMA agent to move from host A to B and return back to A	the required time for AGLET agent to move from host A to B and return back to A
1	1493	207
2	1520	145
3	1530	146
4	1400	175
5	1270	163
6	1690	147
7	1610	152
8	1510	173
9	1540	161
10	1290	145
11	1690	184
12	1460	148
13	1410	149
14	1510	188
15	1530	139
16	1590	152
17	1830	193
18	1520	139
19	1910	151
20	1630	188
21	1530	140
22	1570	171
23	1520	165
24	1500	169
25	1410	142
26	1600	193
27	1720	153
28	1810	163
29	1310	171
30	1640	170
Average	1551.57	162.32

Table I.1: AGLET and TACOMA'S Moving Cost