

**PROJECT MORPHEUS**  
**A GRASSROOTS APPROACH TO HIGH**  
**PERFORMANCE CLUSTER COMPUTING**

**A Dissertation Report**  
**submitted by**

**NAGARAJ POTI [MY.SC.P2MCA14078]**  
**PRASOBH P.S [MY.SC.P2MCA14071]**

**in partial fulfilment of the requirements for the award of the degree of**

**MASTER OF COMPUTER APPLICATIONS**

**under the guidance of**

**MR. ADWITIYA MUKHOPADHYAY**

**Lecturer**

**Department of Computer Science**

**Amrita Vishwa Vidyapeetham**

**Mysuru Campus**



**AMRITA VISHWA VIDYAPEETHAM**  
**MYSURU CAMPUS**

**May 2016**

## BONAFIDE CERTIFICATE

This is to certify that this dissertation report entitled “**PROJECT MORPHEUS: A GRASSROOTS APPROACH TO HIGH PERFORMANCE CLUSTER COMPUTING**” submitted to **AMRITA VISHWA VIDYAPEETHAM, Mysuru Campus, Mysuru**, is a bonafide record of work done by **NAGARAJ POTI** and **PRASOBH P.S** under my supervision from **July 2015** to **May 2016**.

Mr. Adwitiya Mukhopadhyay  
Lecturer

**PROJECT COORDINATOR(PG)**

Ms. Shobha Rani N

**CHAIRPERSON**

Mr. Gokuldev S

Mysuru

16th May 2016

**Internal Examiner**

**External Examiner**

## DECLARATION BY AUTHORS

This is to declare that this report has been written by us. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. We are aware that if any part of the report is found to be plagiarized, we shall own full responsibility for it.

Nagaraj Poti  
MY.SC.P2MCA14078

Prasobh P.S  
MY.SC.P2MCA14071

Mysuru  
16th May 2016

## ACKNOWLEDGEMENT

First and foremost, we are deeply indebted to Her Holiness Most Revered **Mata Amritanandamayi Devi (Amma)** for her inspiration and guidance both in unseen and unconcealed ways.

Wholeheartedly, we thank our Respected Director, **Br. Sunil Dharmapal** and Correspondent **Br. Venu Gopal** for providing the necessary environment, infrastructure and encouragement for carrying out our dissertation work at Amrita Vishwa Vidyapeetham, Mysuru Campus, Mysuru.

We express our heartfelt gratitude to **Prof. Vidya Pai C**, Principal, Amrita School of Arts and Sciences, Amrita Vishwa Vidyapeetham, Mysuru for her continuous support and encouragement. We would also like to thank **Mr. Gokuldev S**, Chairperson, Department of Computer Science, for his invaluable guidance and input throughout the project. We sincerely thank **Ms. Shobha Rani N**, Project Coordinator, for her constant supervision and motivation to succeed. We take this opportunity to thank all review panellists for their critical feedback, comments and suggestions.

Last but not least, we are indebted to **Mr. Adwitiya Mukhopadhyay**, Project Guide, for his continuous guidance, recommendations and support in completing the project. He has been instrumental in keeping the project on schedule and for obtaining the necessary resources.

We thank all the support staff of the ICTS department for providing necessary arrangements for carrying out the project. We are grateful for the blessings our parents have bestowed on us. Their support kept us going even when the going got tough.

Nagaraj Poti  
Prasobh P.S

# List of Figures

1.1	HPCC Architecture . . . . .	1
2.1	Split Brain Condition . . . . .	11
4.1	Resource Scheduling Implementation Architecture . . . . .	23
4.2	Ad Hoc Cluster Use Case . . . . .	23
4.3	Resource Scheduling Algorithm Flow . . . . .	24
5.1	Cluster Monitor Region View . . . . .	27
5.2	Resource Scheduling Algorithm . . . . .	29
5.3	General Architecture for Building Trusted Wireless Clusters . . . . .	30
6.1	A Graphical Representation of Table 6.1 . . . . .	32
6.2	Hit/Miss Ratio Simulation Results . . . . .	34
6.3	Hit/Miss Count Chart . . . . .	34
6.4	Max Starvation Deadline Trend Chart . . . . .	34
6.5	Live Test Bed Battery Consumption Benchmark Comparison . . . . .	35
6.6	Malicious Node Intercepting Packets . . . . .	36
6.7	Destination Node Packet Loss . . . . .	36
A.1	Beowulf Cluster Mergesort Execution . . . . .	42
A.2	Cluster Monitor Application Submission . . . . .	42
A.3	Resource Scheduler . . . . .	43
A.4	Node 1 Job Execution . . . . .	43
A.5	Node 2 Job Execution . . . . .	43
A.6	Performance Simulation Node Initialization . . . . .	44
A.7	Performance Simulation Cluster Monitor Assignment . . . . .	44
A.8	Performance Simulation Statistics . . . . .	44
A.9	Benchmark Simulator . . . . .	45
A.10	NS3 Infinite Funnel Attack Visualization . . . . .	45
A.11	NS3 Infinite Funnel Attack Statistics . . . . .	45

# List of Tables

1.1	Comparison between Cluster, Grid and Cloud Paradigms . . . . .	2
2.1	Key Characteristics of High Performance Architectures . . . . .	10
2.2	Wireless Cluster Implementations . . . . .	15
3.1	Falsified resource description table at a cluster monitor . . . . .	19
5.1	Cluster Monitor Cache . . . . .	28
6.1	Execution Statistics of Sequential and Parallel Mergesort . . . . .	31

# Abbreviations

HASP	Houston Automatic Spooling Priority
JES	Job Entry System
HPCC	High Performance Cluster Computing
LAN	Local Area Network
CPU	Central Processing Unit
WAN	Wide Area Network
RMS	Resource Management and Scheduling
POP	Pile of PCs
MANET	Mobile Ad Hoc Network
N	Total no. of nodes currently in ad hoc cluster excluding master node
M	Total no. of cluster monitors
s	User defined max number of cluster neighbours to a cluster monitor
rand()	Random function with theoretical equal probability selection
R	Resource capacity of node
T+	Positive threshold for resource capacity
T-	Negative threshold for resource capacity
Q	Execution time remaining
F	Free memory available
$J_i$	Job instruction count estimate
$T_i$	Time slice of job
ARU	Average Resource Utilization
t	Total running time of simulation

## Abstract

*Cluster computing is the pooling of resources of multiple homogeneous systems in a scalable manner to perform computationally intensive tasks efficiently. It offers several benefits over traditional mainframes and supercomputers in terms of cost-effectiveness, scalability and parallel distribution of tasks. The objective for this project is to develop from scratch and deploy a high performance computational cluster in a portable environment for faster processing of multithreaded applications. There is a need for better understanding of what cluster computing can offer and how cluster computers can be constructed. The modern shift in paradigm towards worldwide distributed computing can be effectively realised by tapping into the potential wireless ad hoc clusters provide. The dynamic and relatively unstable nature of ad hoc networks render existing resource scheduling algorithms for wired clusters as quite inefficient implementations in this scenario. A resource scheduling strategy for wireless ad hoc clusters is proposed by making careful considerations on decisions that would affect battery life of nodes. This is done by using predictive calculations at nodes to communicate only when absolutely essential, such as when assigning jobs and performing infrequent state message exchanges. The strategy ensures “worst case zero loss” performance whereby every exchange of message between nodes is a gain of explicit and inferred information. The resource scheduling strategy is then further analysed experimentally with respect to some crucial performance metrics such as hit/miss ratio, starvation deadline and average resource utilization through the use of simulations. Furthermore, the physical characteristics of the wireless medium and distributed nature of computing gives rise to many targeted security vulnerabilities that need to be properly identified and addressed. A survey is carried out on security issues, threats, challenges and contingency measures that are suitable to reduce the risk of occurrence and impact should such threats materialize. Security issues that impact battery life of nodes, performance of the cluster and data integrity are analyzed carefully, culminating in a general policy and architecture proposal for establishing node trust in wireless clusters.*



# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
1.1	HIGH PERFORMANCE COMPUTING . . . . .	1
1.2	COMMODITY CLUSTERS . . . . .	2
1.3	WIRELESS AD HOC CLUSTERS . . . . .	4
1.4	RESEARCH OBJECTIVES . . . . .	6
1.5	APPLICATIONS AND CONTRIBUTION . . . . .	6
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>9</b>
2.1	TRADITIONAL WIRED CLUSTERS . . . . .	9
2.2	WIRELESS AD HOC CLUSTER SURVEY . . . . .	13
2.3	MOTIVATION . . . . .	14
2.4	A COMPARISON STUDY . . . . .	15
<b>3</b>	<b>TECHNICAL UNDERPINNING</b>	<b>16</b>
3.1	SECURITY CHALLENGES . . . . .	16
3.1.1	Denial of Service (Network Layer) . . . . .	16
3.1.2	Data Corruption (Application Layer) . . . . .	18
3.1.3	Authentication and Authorization . . . . .	20
3.2	SYSTEM SPECIFICATIONS . . . . .	21
3.3	PROGRAMMING TOOLS USED . . . . .	21
<b>4</b>	<b>LOGICAL DESIGN</b>	<b>23</b>
4.1	ARCHITECTURE DIAGRAM . . . . .	23
4.2	USE CASE DIAGRAM . . . . .	23
4.3	ALGORITHM FLOWCHART . . . . .	24
<b>5</b>	<b>SECURE SCHEDULING PROPOSAL FOR WIRELESS CLUSTERS</b>	<b>25</b>
5.1	RESOURCE SCHEDULING STRATEGY . . . . .	25
5.2	TRUST ESTABLISHMENT MODEL . . . . .	29

<b>6</b>	<b>EXPERIMENTAL ANALYSIS</b>	<b>31</b>
6.1	BUILDING A BEOWULF CLUSTER . . . . .	31
6.2	SIMULATION RESULTS . . . . .	32
6.3	DEMONSTRATION: INFINITE FUNNEL ATTACK IN A WIRE- LESS CLUSTER . . . . .	36
<b>7</b>	<b>CONCLUSION</b>	<b>37</b>
7.1	SUMMARY . . . . .	37
	<b>Bibliography</b>	<b>38</b>
	<b>Appendices</b>	<b>41</b>
<b>A</b>	<b>Screenshots</b>	<b>42</b>

# Chapter 1

## INTRODUCTION

### 1.1 HIGH PERFORMANCE COMPUTING

The preliminary idea leading to cluster computing was developed in the 1960s by IBM as a way of connecting large mainframes to provide profitable commercial parallelism. IBM developed the HASP system and its successor, JES, that provided a way of distributing work over a mainframe cluster. However, cluster computing did not gain popularity until the convergence of three trends: high performance microprocessors, high speed networks and standard tool kits for distributed computing. The increased need of computing potential for scientific, commercial and industrial applications as well as future technologies can be viewed as the fourth driving factor coupled with the high cost and low accessibility of traditional supercomputers.

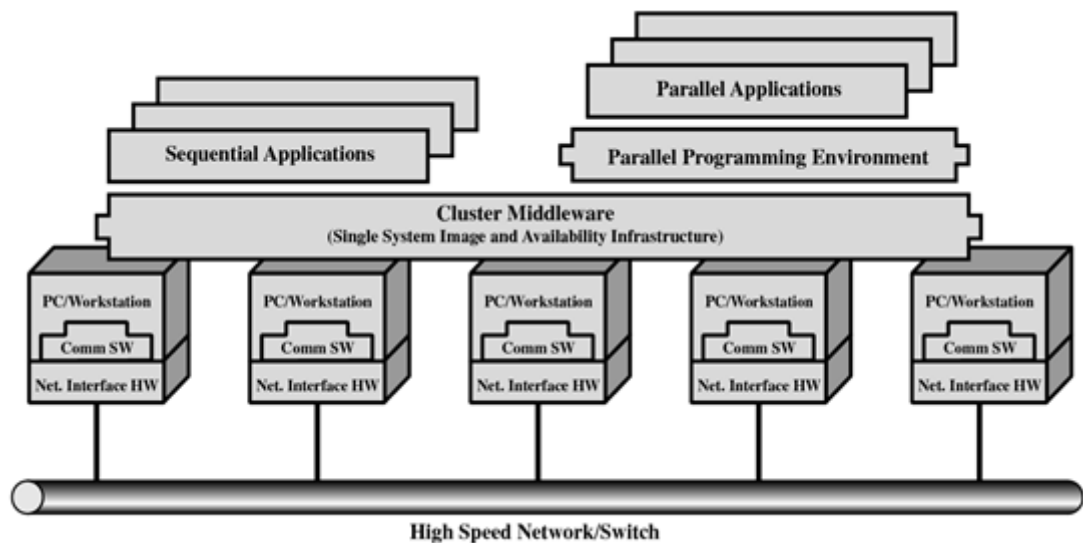


Figure 1.1: HPCC Architecture

High Performance Computing is critical for any branch of science due to the inherent complex and vast nature of computations to be performed. Cluster, Grid and Cloud paradigms of High Performance Computing show significant topology and intent differences. A brief comparison is made in Table 1.1. Cluster computing is a collection of tightly coupled computers that work together as a single system within a LAN, with each node running its own instance of an operating system [1]. The main objective of adopting cluster computing is to process huge volume of data or perform processor intensive tasks in an efficient manner with low cost, high scalability and parallel processing. Cluster cells rely on a centralized management approach with one master node and several slave nodes. Cluster technology allows processing power to be boosted using commodity hardware and software components that can be acquired at relatively low costs. This provides scalability, an affordable upgrade path that lets organizations increase their computing power, while leveraging the existing investment without incurring additional expense. Furthermore, application performance also improves with the support of a scalable software environment that a cluster can emulate.

Table 1.1: Comparison between Cluster, Grid and Cloud Paradigms

<b>Cluster</b>	<b>Grid</b>	<b>Cloud</b>
Single System Image	Distributed	Centralized Resource Virtualization
Fine Grain Synchronization	Difficult to achieve due to geographical dispersal of nodes	
Better Security	Biggest challenge yet to be overcome	
Self Ownership	Dependence on external entity	
Local Network	Internet connection is compulsory	

## 1.2 COMMODITY CLUSTERS

Clusters are segregated based upon various factors as indicated below:

### Target application

- High Performance Clusters: Primarily used to perform complex computations
- High Availability Clusters: Primarily used as stores for big data with high retrieval speeds

## **Node ownership**

- Dedicated Clusters: Cluster nodes devote all available resources to cluster
- Non-dedicated Clusters: Individuals own the cluster nodes that share idle resources to perform a task

The clusters differ on the basis of ownership of nodes in a cluster. In dedicated clusters, an individual does not own a workstation. All the resources are shared so that parallel computing can be performed across the entire cluster. Non dedicated clusters have workstations that are owned by individuals. Tasks are executed by stealing idle CPU cycles. The motivation for this approach is based on the fact that 90% of CPU cycles are unused, even during peak hours. Parallel distributed computing on a dynamically changing collection of non dedicated workstations is termed as adaptive parallel computing. In non dedicated clusters, workstation owners and remote users who need the workstations to run their application have clashing sets of requirements. The former expects quick interactive response from their workstation, while the latter is only concerned with fast turnaround times for jobs submitted. This emphasis on sharing processing resources forsakes the concept of node ownership and introduces complexities like process migration and load balancing techniques. Thus, clusters will be able to deliver satisfactory interactive performance as well as provide shared resources to demanding parallel applications.

## **Node configuration**

- Homogeneous clusters: All nodes will have similar architecture and will run the same operating systems
- Heterogeneous clusters: Nodes may have varied architectures running different operating systems

## **Levels of clustering**

- Group clusters: Nodes range between 2 to 99
- Departmental clusters: Nodes range between 10 to 100s
- Organization clusters: Nodes vary in the 100s
- National Metacomputers (WAN/Internet based): Collection of many departmental/organizational clusters

- International Metacomputers: Nodes vary from thousands to many millions

The following list highlights some of the reasons cluster computing is preferred over supercomputing alternatives:

1. Individual workstations are becoming exponentially powerful, doubling every 18 to 24 months. This is likely for a few more years, with faster processors and more efficient multiprocessor, multi-core systems coming into the market. Even though this is currently feasible, future improvements are limited by the speed of light, thermodynamic laws, and the high financial costs associated with processor fabrication.
2. The communications bandwidth between workstations is increasing whereas latency is decreasing as new networking technologies and protocols are being implemented in a LAN.
3. Workstation clusters are flexible enough to integrate into existing networks than special parallel computers.
4. Personal workstations show typically low user utilization, even at peak times.
5. The development tools for workstations are standardised compared to non-standard nature of proprietary solutions for specialised distributed systems.
6. Workstation clusters are relatively far cheaper and readily available (commodity components) alternatives to specialized computing platforms
7. Clusters are highly scalable whereby a constituent node's capability can be easily upgraded by adding additional memory or processors.

### **1.3 WIRELESS AD HOC CLUSTERS**

The proliferation of wireless networks over the past decade has allowed for the possibility of truly ubiquitous distributed computing. Traditional clusters and grids are characterised by wired connectivity [3], putting a physical restriction on the scalability of such systems. The potential of using mobile heterogeneous devices connected via ad hoc networks as a collaborative computing resource is a step towards pervasive computing. Local cluster computing units can be aggregated together to form a global grid of intercommunicating resource sharing systems.

Ad hoc networks are not reliant upon well established infrastructures to enable connectivity. Thus, several differentiating characteristics of such networks must be

thoroughly considered in the design of a resource sharing and scheduling strategy for wireless clusters. In particular, ad hoc networks are very dynamic in nature and constituent nodes generally have resource constraints such as limited battery life and processing capability [3]. Devices such as laptops, smart phones and tablets are widely available in large numbers today and thus provide great computing potential for complex tasks if utilised together. Thus a wireless cluster using commodity hardware and infrastructure is a very tempting proposition.

Nodes in ad hoc networks are characterised by their mobility and dynamic association with the network. The network topology and configuration shifts dynamically implying that the cluster implementation on top must address resource discovery, description and tracking [4]. Wireless networks are inherently vulnerable to threats, so proper authentication mechanisms must be delegated to the underlying network service.

Limited battery life of mobile devices must be given utmost priority of concern during algorithm design. Significant drain on battery reserves would negate the productivity gained by deploying an ad hoc cluster as the cluster will not be sustained. Communication over the network must be minimized to prevent abrupt node failure without productive output. However, in the case of node failure as is bound to happen, fault identification and corrective rescheduling measures must be taken according to the load assigned to current nodes in the network.

This work proposes an algorithmic strategy for the scheduling of jobs on a wireless ad hoc cluster using calculated guesstimates that guarantee “worst case zero loss” performance. The algorithm reduces inter-node transmission by estimating the various states of the nodes in the network based on cached information.

Another aspect of study that is undertaken is that of security considerations when constructing a wireless cluster. Ad hoc networks are vulnerable to some well-known threats that are intrinsic to the communication medium and that arise due to the dearth of supporting infrastructure. Specific attacks such as eavesdropping and active wiretapping that are difficult to carry out in wired networks are far easier to carry out in wireless networks [2]. Thus, securing wireless ad hoc networks is a challenge in itself. Moreover, there are performance and battery life constraints that determine the security controls established to mitigate security issues.

This work presents serious attacks that can exploit flaws and loopholes to compromise the functioning of wireless ad hoc high performance clusters. The fundamental motivation for the construction of a cluster is to obtain a significant gain in computing performance, one that can be severely hampered by security risks.

A high performance cluster is mission critical, implying that the correctness and consistency of data accumulated, transferred and processed within the cluster is paramount. In the following sections, major security vulnerabilities and challenges in the development of wireless clusters are discussed.

## **1.4 RESEARCH OBJECTIVES**

The project has the following objectives it seeks to accomplish within the stipulated time period of 7-8 months:-

1. Keenly scrutinise and simulate the Resource Management and Scheduling module for an ad-hoc cluster with suitable enhancements and optimizations as fit for the explicit requirements of the project
2. Develop a lightweight message passing protocol for communication between nodes of the ad-hoc cluster cell
3. Create the command line application interface for the user to interact with
4. Survey security vulnerabilities that may affect the functioning of wireless clusters and suggest suitable contingency measures
5. Simulation of an attack that can be carried out with relative ease and that may seriously affect cluster performance
6. Run multi threaded mathematical algorithms to exact the performance boost gained by the implemented project
7. Develop prototype wireless ad hoc high performance cluster
8. Develop documented code for the use of the application and the internal working of the developed software

## **1.5 APPLICATIONS AND CONTRIBUTION**

Resource intensive scientific and industrial applications can benefit immensely from the vastly superior processing capability to get quicker results. Furthermore, business applications can execute billions of complex calculations in parallel. A cluster computing cell provides an ideal environment for the programming and development of parallel algorithms.



Wireless cluster computing extends the traditional cluster computing paradigm to include a broad range of mobile devices enabled to communicate using radio frequency, microwave, infrared, optical and other wireless media. Devices such as sensors, Personal Digital Assistants (PDAs), mobile phones, wearable computers, laptops and special purpose computers embedded into modern appliances [5, 6, 7] are coming into use in distributed implementations. Though many of these gadgets were primarily developed to serve as autonomous entities, the potential for cooperation through the sharing of resources of a large number of devices, is quickly leading to applications similar to traditional cluster computing.

McKnight et al in [8] group wireless cluster/grid applications into three overlapping categories based on the differentiating attributes wireless devices have over traditional wired devices:

- Applications which accumulate live data
- Applications which take advantage of mobility
- applications which use cooperation among wireless devices to perform complex tasks

Wireless cluster applications exploit the distributed nature of the devices involved, aggregating the data collected from widely distributed devices to perform complex real time computations. The increasing variety of wireless cluster applications allows for the monitoring of buildings, businesses, tunnels, bridges and the people within them [9]. Communicating devices can be used to track the distribution chain (movement of goods from manufacturer to the customer) [10] or even the movement of the Earth's crust at fault-lines to predict tsunamis, volcanic eruptions or earthquakes [11]. The mobile nature of consumer wireless devices makes possible applications such as ad hoc distribution of streaming media [12] and near range communication capabilities of mobile phones. A hot emerging application is one that allows a user to use mobile phones to interact with smart devices in their environment from a distance.

In India, the study of high performance and cluster computing systems have been limited to very few universities. Even fewer universities have a cluster computing lab developed indigenously. This project provides a perfect opportunity to learn and engage in the development of a cluster cell, at the very same time attempting to develop an application interface for ad hoc parallel processing of a multithreaded application over several interconnected nodes. Existing high performance cluster cells are limited by the following:

- Existing cluster cells are extremely specific to the purpose they were developed for. Cluster cell systems have been developed for weather forecasting, business and economic big data analysis, parallel 3D rendering [13] etc. Our proposal entails a general purpose portable parallel processing interface that will enable a scalable number of homogeneous systems to form an ad hoc cluster.
- The cost associated with the setting up of a cluster cell, while considerably lesser than that of a supercomputer, is still quite high because they are generally developed and deployed for dedicated purposes. A cost-effective approach is proposed in this project where price-performance ratio is considered and common-of-the-shelf hardware is utilized.
- Cluster cells are still quite inaccessible for personal use. This project aims at bringing cluster performance closer towards personal and home users.
- Lack of code documentation is common among current prototype implementations as several design decisions are hidden. This project aims at being well documented at every stage of progress.
- Ease of installation and deployment has not been a point of focus in the development of existing cluster cells. Project Morpheus will provide an easy to use interface for the user.

# Chapter 2

## LITERATURE SURVEY

### 2.1 TRADITIONAL WIRED CLUSTERS

Pfister [14] points out that there are three major ways of improving efficiency and performance work harder, work smarter and get help. Working harder refers to using faster hardware, working smarter refers to using more efficient algorithms for carrying out tasks and getting help involves using numerous computers to solve challenging computational problems. India's supercomputer program was started in late 1980s. Param Yuva II is one of India's fastest supercomputers constructed at a cost of Rs.160 million [15]. During the past two decades, many different computer systems supporting high performance computing have emerged. The architectural and functional characteristics of these machines originally given in [16] by Hwang and Xu is given in Table 2.1.

An analysis of the aforementioned systems concludes that cluster cell architecture is best suited for our purpose of setting up temporary cluster cells when required. Moreover, the usage of clusters is practically suited because of the standardisation of many tools and utilities utilized by parallel applications. Examples of these standards are the message passing library MPI [17] and data-parallel language HPF. The choice of operating system for the implementation of this project is the Linux/Unix. Linux provides features such as pre-emptive multitasking, demand-paged virtual memory, multi-user, and multiprocessor support [18]. Furthermore, the POSIX threads interface is a standard programming environment for creating parallel running threads within a process. Increasingly the Linux environment is being extended for mobile architectures as well, making the platform an ideal choice.

The Beowulf project's [19] aim was to investigate the potential of PC clusters

Table 2.1: Key Characteristics of High Performance Architectures

<b>Characteristic</b>	<b>MPP</b>	<b>SMP/NUMA</b>	<b>Cluster</b>	<b>Distributed</b>
<b>Number of nodes</b>	100 - 1000	10 - 100	100 or less	10 - 1000
<b>Node complexity</b>	Fine/Medium grain	Medium/Coarse grained	Medium grain	Wide range
<b>Internode communication</b>	Message passing / shared variables	Centralized and DSM	Message passing	Shared files, RPC
<b>Job scheduling</b>	Single run queue on host	Single run queue	Multiple queues but coordinated	Independent queues
<b>SSI support</b>	Partially	Always in SMP and some NUMA	Desired	No
<b>Node OS type</b>	N micro-kernels monolithic or layered OSs	One monolithic SMP and many for NUMA	N OS platform - homogeneous	N OS platforms heterogeneous
<b>Address space</b>	Multiple	Single	Multiple or Single	Multiple
<b>Internode security</b>	Unnecessary	Unnecessary	Required if exposed	Required

for performing computational tasks. The communication between processors in Beowulf is through TCP/IP over the Ethernet internal to cluster. The performance of interprocessor communication is limited by the performance characteristics of the Ethernet and the system software managing message passing. To make the cluster truly portable, other communication media such as wireless Bluetooth, Laser and Wi-Fi were not considered.

There are various measures of performance for clusters. However, the most important three parameters that need to be analysed for high performance computing are (i) High Availability (ii) Load Balancing and (iii) Fault Tolerance.

### High Availability

High availability implies that services must be available round the clock despite transient failures at the hardware or software level. The performance of the clusters substantially increases when processors co-operate with each other to manage resources globally. This cooperation results in significant availability loss in the absence of high availability mechanism. Furthermore, when high availability techniques are enforced, they may result in inconsistent recovery actions [20].

High availability clusters operate by utilizing redundant nodes which provide services when system components fail. Normally, if a server hosting an applica-

tion crashes, the application will be unavailable until the server is restored. High availability clustering provides a better alternative for this situation by identifying hardware/software faults and immediately restarting the application automatically on another system without requiring any administrative intervention. This process is known as failover. Failover implies that only one server is active at any given time out of at least two servers in the system. If the active one stops responding, requests are directed to the other server the system 'fails over' to the backup server. Failovers are of two types: cold failover and hot failover. Cold failover means that the backup server is booted up only after the first one goes down. Hot failover means that all servers are running simultaneously and that the load is directed entirely towards one server at any time. High availability manages redundancy inherent in a cluster to eliminate single points of failure. A high availability cluster is designed such that there will be no loss of data or work by effortlessly switching between primary and backup nodes. This implies that data on the backup should be constantly updated in real time.

High availability clusters usually consist of two separate networks: a public network and a private network. The public network is used to access the active node of the cluster from outside the data center. The private network is used for local communication between cluster nodes. A heartbeat private network is to monitor the status of each processor in the cluster and avoids a split brain condition to occur [21]. Split brain occurs when all local links of the cluster fail simultaneously, but the nodes remain functioning. Each node in the cluster may incorrectly assume that all other nodes have gone down, and accordingly attempt to restart services in the backups. Nodes having duplicate instances of services may cause data corruption in shared storage.

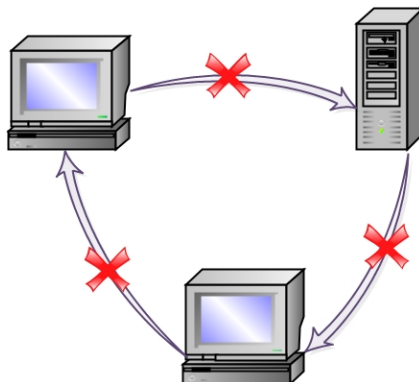


Figure 2.1: Split Brain Condition

## Load Balancing

The balancing of computational workload across the system is quite important. Load balancing is a technique for scaling the performance of the system by distributing load among multiple clusters. A cluster provides high availability by making data and application available on several clusters linked together. If a cluster ceases to function, a failover procedure automatically shifts the workload of the failed cluster to another cluster. After the failed cluster recovers, it rejoins the cluster system and is known as failback. Load balancing provides failover and failback support for continuous cluster availability. The load balancing manager can specify the load percentage that it will handle or the load that can be equally distributed across all the clusters. If a cluster fails the load balancing mechanism dynamically redistributes the load among the remaining clusters. Load balancing technique is used to enhance scalability, which boost throughput while keeping response times low [22]. The ability to easily transfer load within a fault tolerant cluster provides outstanding availability during system maintenance activities such as software upgrades. When maintenance is completed, the node rejoins the cluster and assumes its normal workload. This robust fault tolerance in clusters makes recovery from failures possible in a reasonably short time.

An effective load balancing technique maximizes efficiency by minimizing processor idling and interprocessor communication. Load balancing also improves performance by minimizing response time [23]. The crucial decisions of when, where and which tasks have to be transferred depend on accurate and up-to-date load information. If the workload is not balanced among processors, a heavily loaded processor may remain busy executing tasks assigned to it while other processors are sitting idle. This will degrade the overall speed-up as the aggregated CPU computing power is not fully utilized. The computations required for each task vary and may change dynamically during program execution. Thus it is not sufficient to map equal number of tasks among processors, rather the work must be equally distributed among processors in a dynamic manner. Many load balancing policies achieve high system performance by increasing the utilization of CPU and/or memory. It uses load information of CPU, memory and network traffic to calculate load of each node. This information on load is then combined with job properties to check whether the job is CPU, memory or I/O bound.

Load balancing is the thrust of the study in the proposed scheduling algorithm, whereby conventional load balancing strategies are fine tuned for battery life considerations.

## **Fault Tolerance**

Performance and fault tolerance are contrasting characteristics in high performance computing. A fault tolerant system is one that continues to operate unhindered and produce acceptable results in spite of occasional component failures. One of the most significant problems in implementing a fault tolerant system is the identification and elimination of single points by using substitutes. When a component becomes unavailable, the loss is detected and that component's workload is shifted to another component in the cluster in an automated fashion. Error recovery is done after the fault is detected and its spread is identified. Error recovery is done either by backward error recovery or by forward error recovery [24]. Though the backward recovery technique is quite efficient, a good diagnosis of the fault is a prerequisite. Backward recovery is generally used in cluster systems. For communication intensive applications, recovery schemes based on coordinated check-pointing are preferable over other schemes [25]. Eduardo in his work [26] deals with power conservation for clusters of workstations or PCs. The experiments showed that it is possible to conserve significant power and energy in the context of clusters.

## **2.2 WIRELESS AD HOC CLUSTER SURVEY**

During the course of the literature review, we came across several works from over the past five years that amount to detailed surveys of the scope of wireless distributed computing. This is an indicator of the need for expansive and deeper research possibilities in this domain. Sanjay P. Ahuja and Jack R. Myers elucidate the scope of the field of wireless grid computing and how it can be viewed as an extension to existing wired grid networks [3]. A special focus on sensor networks as an arm of wireless grid computing, its practical applications and standards supporting wireless grids is discussed. S. S. Manvi and M. N. Birje present a review of wireless grid computing and list out the challenges that lie ahead [27]. We seek to address some of those challenges which are equally applicable to wireless high performance clusters through our work.

Mohamed Aissa and Abdelfatteh Belghith have proposed wireless ad-hoc networks with appropriate selection of cluster-heads so as to increase Quality of Clustering parameters [28]. The main intention behind this approach is to identify favourable nodes in cluster-head election and re-election process. This work is an important foundation for obtaining optimal performance in the wireless network

backbone on which the high performance cluster would be deployed. Furthermore, Alan D. Amis and Ravi Prakash propose a load balancing strategy for the fair selection of cluster-heads to act as hubs responsible for the routing of packets in ad hoc networks [29]. Their strategy extends the life of cluster-heads as much as possible before the battery of the node drains out, allowing the cluster-head to retire gracefully. Achieving secure communication between nodes in ad-hoc networks is discussed by Li Xia and Jill Slay [30]. In their paper, they discuss about key management issues and challenges in wireless ad hoc networks, and propose an approach involving mobile agents. Since the concept of network ownership and control is a contentious issue, secure transmission of messages is necessary.

Stanislav Kurkovsky and Bhagyavati in their work propose an architecture using MANET cellular networks to provide ubiquitous computing [31]. A resource broker is responsible for the delegation of tasks to nodes in the network and keeps track of task completion status of nodes. However, this puts burden on a central node to keep track of the entire network, thus reducing scalability. Aksenti Gnarnarov et al. propose an implementation of grid computing in ad hoc networks [4]. They introduce a concept of Ad Hoc Grid Layer that acts as middleware and implements a single system image view over the network. They present their findings culminating in a simulation using the NS-2 network simulator. The scope of the work does not extend to restricted battery life considerations of nodes. Zhi Wang et al. in their paper suggest a mobile agent based approach for deploying a grid over MANET [32]. Resource description and discovery is the responsibility of the mobile agent. It also ensures that nodes have sufficient battery power and resources to execute the job before it is assigned to them - a proactive approach. However, a proactive approach again puts burden on the network bandwidth and node life with continuous state message exchanges.

## 2.3 MOTIVATION

The extensive literature survey has quite clearly indicated that wireless ad hoc clusters are potentially game changing with respect to the endless possibilities they present. Current and on-going research has just scratched the surface of this field. A future where the all the computing potential embodied by hardware can be harnessed in a collaborative manner would increase the capability of mankind to perform computations that are infeasible today. The diverse range of applications of wireless clusters ranging from earthquake detection to collaborative computing make this avenue worth exploring. Results from several simulations and exper-



iments carried out till date are highly optimistic and should be extended and carried on further until the vision of a truly ubiquitous computing environment is achieved.

## 2.4 A COMPARISON STUDY

Table 2.2 is a comparison of several approaches that have been adopted to develop wireless clusters.

Table 2.2: Wireless Cluster Implementations

Authors	Title	Problem	Methods and Results
Mohamed Aissa and Abdelfatteh Belghith	“Quality of Clustering in mobile Ad Hoc networks”	Appropriate selection of clusterheads in ad hoc networks for routing stability in dynamic topology configurations	<b>Method:</b> prioritize favourable nodes in clusterhead election and re-election processes <b>Result:</b> Algorithm outperforms Weighted Clustering Algorithm (WCA) in terms of cluster formation and stability
Aksenti Grnarov et al.	“Grid Computing Implementation in Ad Hoc Networks”	Establishment of ad hoc mobile grid and discovery of resources for task scheduling	<b>Method:</b> introduce a new independent Ad Hoc Grid Layer (AHGL) on top of network architecture <b>Result:</b> AHGL reduces unnecessary traffic by including only the nodes that satisfy the criteria for execution of certain jobs
Alan D. Amis and Ravi Prakash	“Load-Balancing Clusters in Wireless Ad Hoc Networks”	Extend the life of a clusterhead to its maximum capability before letting the clusterhead retire and give way to another node	<b>Method:</b> favour the election of clusterheads based on node id and place a limit on the contiguous amount of time that they remain so <b>Result:</b> larger clusterhead durations with decreased variance observed, implying increased stability
Stanislav Kurkovsky and Bhagyavati	“Wireless Grid Enables Ubiquitous Computing”	Architecture for a wireless grid that facilitates mobile devices to solve resource intensive tasks by harnessing the power of other devices with readily available resources	<b>Method:</b> Devices that can interact within the cluster/grid cell assigned roles corresponding to Brokering Service, Initiator and Subordinates <b>Result:</b> Concept of architecture proved by experimental simulation implementation

# Chapter 3

## TECHNICAL UNDERPINNING

### 3.1 SECURITY CHALLENGES

#### 3.1.1 Denial of Service (Network Layer)

##### **Wormhole attack**

A wormhole attack is initiated by a threat agent when the routing table of a node in the network is compromised. This compromised node surveys the network, populating the routing table with the worst possible paths to send the incoming packets through [33]. The node presents itself as being the quickest hop to all possible destination nodes, acting as wormhole portal for quick transmission, although the tunnels are actually relatively much longer and inefficient paths. Thus other nodes in the network are fooled into transmitting packets via this node [34].

Although the wormhole attack seldom incapacitates the functioning of the network, it has negative effects on the performance of the network. In wireless ad hoc cluster applications, this would degrade the throughput of the entire cluster due to the increase in latency and transmission delays in the receipt of packets. Since all the packets pass through the wormhole, the attacker can capture, view and modify passing data. Mission critical applications like clusters need to enforce required confidentiality and integrity, the lack of which is a considerable security threat. An attacker could identify each job and its parameters that have been submitted to the cluster for execution. Modification to job parameters would result in incorrect outputs. Furthermore wormholes can encourage job starvation and delayed processing at destination nodes by bouncing incoming data packets. The effects of a wormhole could possibly be alleviated in a manner similar to black holes as well as using packet leashes [35] to restrict the maximum allowed distance

over which a packet may be transmitted.

### **Sybil attack**

A single node impersonating many non-existent nodes seeks to create an illusion of surplus resources in ad hoc high performance wireless clusters. Without proper authentication, cluster monitors may assign tasks to these imaginary nodes, which are added to the job assignment priority queues, with complete trust. As these nodes are not capable of processing those requests, job processing would be considerably delayed accounting for job reallocations and in fact no task handling may occur at all. Furthermore, the fact that when we add more number of nodes in the cluster the performance of the cluster increases only up to a particular threshold, thereafter the performance gain decreases because the overheads of message passing, state information exchange and monitoring are greater than the perceived performance gain from adding more cluster nodes [34].

This attack may be prevented with proper cluster level authentication mechanisms as described previously in order to ascertain trusted nodes. Thus the non-existent nodes are ignored and not considered a part of the cluster. Strict checking for separate network addresses and an analysis of aggregated computing power of the suspicious nodes [36] could be performed to identify multiple identities of a single node.

### **Black hole attack**

A black hole attack is a security threat in mobile ad hoc networks (MANET). It is a denial of service attack whereby a threat agent manages to register a malicious node (acting as a black hole) in the network and masquerades itself to present itself as a reliable node in the network. Such nodes will simply discard all packets that are routed via the node to other destinations instead of relaying it to the appropriate neighboring node. When a node needs to communicate with some other node in the network, the sender would attempt to find the shortest path to the destination according to routing information received by broadcast messages. A malicious node would advertise itself by broadcasting as being the optimal path with the shortest number of hops to send the packets through [37]. The sender unwittingly acknowledges the reply from the malicious node, thus, sending all packets to it. The malicious node receives incoming packets, drops them and sends back falsified messages of successful delivery [34].

This attack is a very certain possibility in wireless ad hoc high performance

clusters and can cause the entire cluster processing to cease. Any node in the cluster may be transformed into a black hole with severe consequences. A cluster monitor which accepts jobs and sub-jobs from the cluster master node splits and assigns these jobs to neighboring cluster nodes that fall under its jurisdiction. If a cluster monitor goes rogue, it would accept jobs from the master node and would do nothing instead of further subdividing them into tasks for assignment to other nodes. One step further, the cluster monitor could then misrepresent node activity and load statistics, causing resource deadlocks and wastage of available resources in the network.

Black hole attacks and the perpetrating nodes are hard to detect. It would require the monitoring of the activities of each node and flagging unusual suspicious behavior [38], which is not feasible in wireless cluster applications due to the performance hit that is inevitable as a result of implementing the mechanism. A conceivable way to reduce the effect of a black hole attack would be to route the packets through different alternate paths to increase the probability of the packets reaching their intended destination [39]. Duplication of effort in the processing of jobs to eliminate the risk of loss of data in the cluster would be an extreme measure that would reduce the availability of computing resources in the cluster.

### **3.1.2 Data Corruption (Application Layer)**

#### **Falsification of resource description table**

A wireless ad hoc high performance cluster takes a regional view of cluster nodes and assigns certain cluster monitors that act as centers for job assignments and communication for other nodes in the cluster. The cluster monitor is responsible for assigning jobs to each neighboring node based on calculated guesstimates. The cluster monitors individually maintain a cache that stores the resource states of its neighboring nodes to which jobs can be assigned. Each node is assigned a state corresponding to supra positive, positive, normal and negative states [40]. The order of the states implies a decreasing availability of computational resources whereby a supra-positive node has abundant free resources, a positive node has adequate free resources for most tasks, a neutral node is a node outside the purview of a cluster monitor and a negative node has no free resources to contribute to the cluster at that instance of time due to already queued up jobs waiting to be executed at the node [34].

This cache at the cluster monitor, also termed as a resource description table, is a prime target for attack. If an attacker manages to get access to a cluster

monitor, he would be able to manipulate the resource description table. The key concept introduced to reduce the drain on the battery life of nodes is the prediction of usage of cluster resources according to probabilistic calculations in order to estimate the current load and availability of free resources at a cluster node. An abstract resource description table with a falsified record is shown in Table 3.1. This reduces the need for frequent message exchanges of state information, thus conserving battery life. However, any malicious modification of the resource description table would corrupt all the calculations and would lead to node failures, delayed job execution and other faults due to job assignments that may be beyond the capability of the node to handle. Usually cluster nodes utilize idle resources available while user interaction and local tasks and processes are being executed at a node. Improper job allocation may cause interference with the local activity performed by a user at a node and thus cause the user to withdraw the node from the cluster.

Table 3.1: Falsified resource description table at a cluster monitor

ID	State	Available Resource Capacity
2	Supra-Positive	$10^9$ MIPS, 2300 MB
5	Positive	4 GFLOPS, 700 MB
7	Negative	$10^7$ MIPS, 1600 MB
8	Supra-Positive	$10^1$ MIPS, 300 MB

### Duplicate services running on a node

A node may temporarily disconnect from the cluster network unintentionally due to the unstable nature of ad hoc networks. However, if the node stays disconnected for too long, then the node is assumed to have failed, when in fact it is only a network issue. If the node happens to reconnect to the cluster after that, the cluster master node would attempt to restart the cluster application on the node that is part of the single system image abstraction layer. However, it is entirely possible that the previous instance of the cluster application is already running at the node because the node didn't actually crash or fail. This leads to multiple copies of the application service running on the same node. Having multiple application services running at the same time would create clashes between them that could cause improper operations to be performed at the node. As a result, a mechanism for ensuring only one instance of the cluster application is running at a node at any instance of time should be ensured. Previous instances of the

application at the node should time out and shut down gracefully in the case of disconnection from the cluster over a predefined amount of time [34].

### **3.1.3 Authentication and Authorization**

#### **Impersonation of an authorized cluster node**

Impersonation is more probable in wide area networks than in local networks. However, the goal of achieving pervasive computing means local wireless clusters should be able to interact together to form inter-domain grids, thus increasing the likelihood of impersonation attacks. Impersonation with respect to high performance clusters involves an attacker that tries to add a possibly fraudulent node to the network by utilizing the authentication and uniquely identifying details of an already authorized cluster node. Various ways in which impersonation attacks are carried out include device cloning, address spoofing, exploiting rogue base stations and masquerading. Device cloning [41], of which address spoofing is a part, is the reprogramming of a foreign node in order to emulate hardware characteristics such as MAC address of another device already authorized to be a part of the network. Cluster monitors as a result cannot distinguish any suspicious behavior and entrust the foreign node that is now illegitimately part of the network. Thus the integrity of the network is compromised and can lead to serious ramifications. A cluster monitor may assign jobs to a malicious cluster neighbor and as a result disclose the nature of the job and data being processed by the cluster. Of even greater consequence is if the task execution results are falsified, resulting in cascading errors causing the improper execution of the assigned job [34].

Impersonation is difficult to detect due to the automated nature of verification of identity. A mechanism to eliminate the possibility of impersonation would require the use of an authentication mechanism that uses some unforgeable, non-transferrable ID. Hence, an application layer authentication scheme must be introduced on top of the basic network level authentication.

#### **Single system image level cluster authentication**

Cluster nodes must be properly authenticated to ensure required confidentiality and integrity of jobs that are processed. A highly effective authentication mechanism should foil impersonation attempts actively. However, due to the lack of infrastructure in ad hoc networks, dedicated authentication hardware such as VASCO DIGIPASS [42] is not an option. However, software based strong two-factor authentication schemes for authentication provide a viable alternative. The

cluster master node would be responsible for allowing initial registration of a cluster node with the wireless ad hoc cluster. Successive accesses and connections would require a uniquely generated numeric PIN at the requesting node that is sent to the master node for verification [43]. The same numeric PIN is generated both at the requesting node and master node in an asynchronous manner, thus disallowing the interception or counterfeit fabrication of the PIN by rogue agents. The trustworthiness and security of the cluster network would be dramatically improved. Thus, any cluster node would have to pass through the underlying wireless network authentication mechanisms as well as the application layer cluster authentication. This layer of additional authentication places an additional burden on the master node with a tradeoff on the battery life of the master node. The master node would need to meet certain minimum specifications for the wireless ad hoc cluster to function optimally.

## **3.2 SYSTEM SPECIFICATIONS**

### **Hardware (Minimum Requirements)**

- Processor: Intel Pentium 4
- Memory: 1 GB Minimum
- Storage: 64GB HDD
- Communication: Wireless (802.11a/b/g/n compliant)

### **Software**

- Operating System: Ubuntu or any Debian OS
- System Software: GCC, SSH, Open MPI
- Application Software: Qt Framework, .NET Framework, NS3 Framework

## **3.3 PROGRAMMING TOOLS USED**

### **G++**

The GNU Compiler Collection includes front ends for C, C++, Objective-C, Fortran, Java, Ada, and Go, as well as libraries for these languages (libstdc++, libgcj,...). GCC was originally written as the compiler for the GNU operating

system. Released by the Free Software Foundation, g++ is a \*nix-based C++ compiler usually operated via the command line. It often comes distributed with a \*nix installation,

### **NS3**

NS (from network simulator) is a name for a series of discrete event network simulators, specifically NS-1, NS-2 and NS-3, that are primarily used in research and teaching. NS-3 is free software, publicly available under the GNU GPLv2 license for research, development, and use. The core of the simulator is written in C++.

### **Qt**

Qt is a cross-platform application framework that is widely used for developing application software that can be run on various software and hardware platforms with little or no change in the underlying codebase, while still being a native application with the capabilities and speed thereof. Qt is available with both commercial and open source GPL v3, LGPL v3 and LGPL v2 licenses.

### **.NET**

.NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library known as Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (as contrasted to hardware environment), known as Common Language Runtime (CLR).

### **Open MPI**

The Open MPI Project is an open source Message Passing Interface implementation that is developed and maintained by a consortium of academic, research, and industry partners. Open MPI is therefore able to combine the expertise, technologies, and resources from all across the High Performance Computing community in order to build the best MPI library available. Open MPI offers advantages for system and software vendors, application developers and computer science researchers.



# Chapter 4

## LOGICAL DESIGN

### 4.1 ARCHITECTURE DIAGRAM

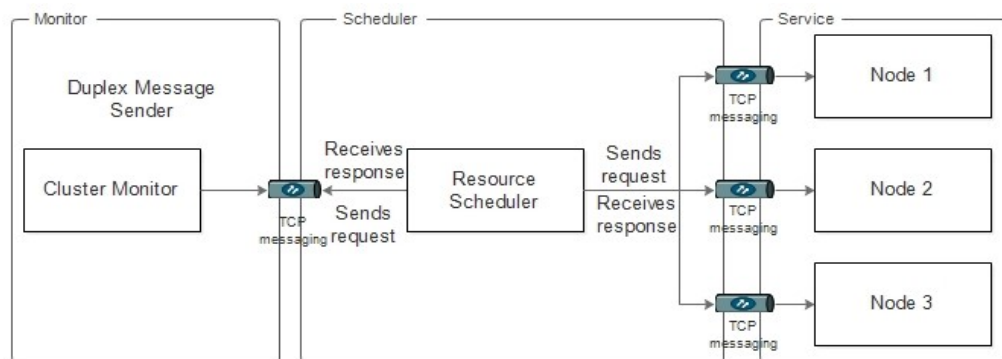


Figure 4.1: Resource Scheduling Implementation Architecture

### 4.2 USE CASE DIAGRAM

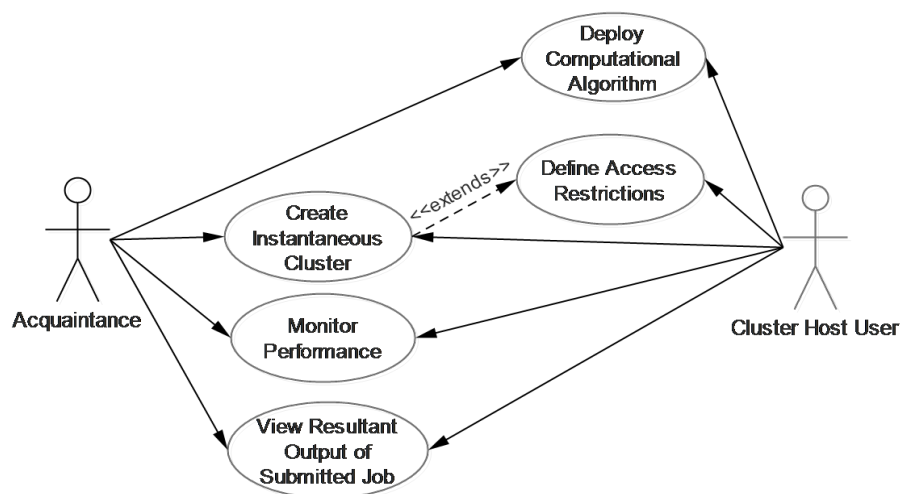


Figure 4.2: Ad Hoc Cluster Use Case

### 4.3 ALGORITHM FLOWCHART

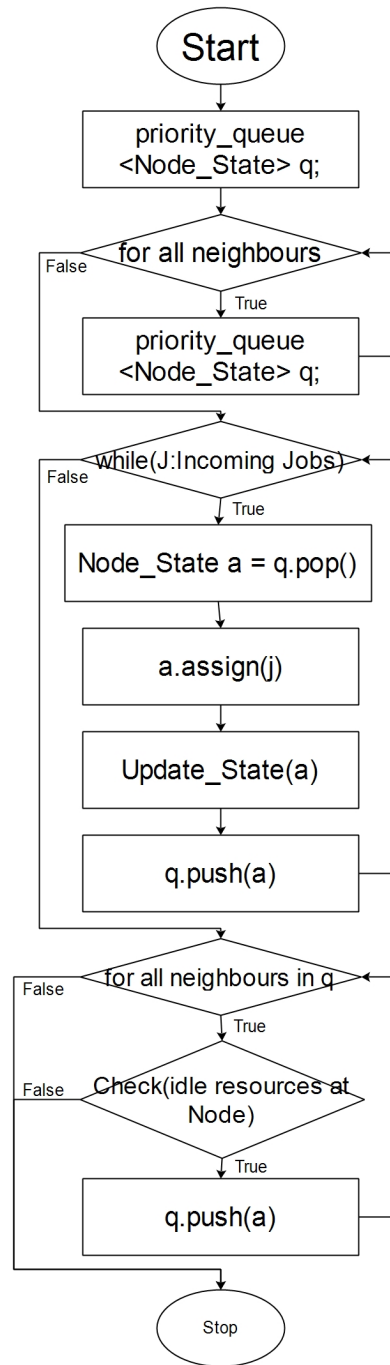


Figure 4.3: Resource Scheduling Algorithm Flow

## Chapter 5

# SECURE SCHEDULING PROPOSAL FOR WIRELESS CLUSTERS

### 5.1 RESOURCE SCHEDULING STRATEGY

The proposed algorithmic strategy makes the following assumptions [44]:

#### **Protocol Stack**

All nodes in the network have basic network (IPv4/IPv6) and transport layer services that will form the underlying communication protocol in the ad hoc network [45]. The idea is to use existing network protocols and infrastructure as a backbone of the wireless cluster.

#### **TCP**

TCP is the transport layer protocol that is used to communicate between nodes as it provides adequate error checking and control at the underlying layer. Accuracy of transmissions is crucial as the results of the computations performed must be correct. However, certain TCP parameters must be tweaked to minimize overheads - `TCPMaxDataRetransmissions` is set at a maximal value of 3 and `KeepAliveTime` is set according to the maximum node starvation period, which is defined in the following section.

## Application Execution Environment

All nodes are assumed to be capable of providing an execution environment for jobs assigned to them. A single programming language may be assumed although it is immaterial. The exact nature of the compiler features and application requirements are not discussed here but is certainly worth consideration.

The proposed algorithm discards a global view of cluster for a region centric view of the cluster [46]. The attempt of a node to keep track of the state of all cluster nodes would be futile and would reduce the scalability of such a system. Instead any node in a network is at most aware of its neighbours. Certain nodes in the network are assigned as cluster monitors that act as local centres for task delegation and state communication. These cluster monitors are selected randomly by the cluster master node and the roles are switched after periodic intervals of time. The number of cluster monitors in the cluster is given by [44]:

$$M = \left\lceil \frac{N}{2 \times s} \right\rceil \quad (5.1)$$

The master node selects M cluster monitors randomly from the set of nodes  $\{n_1, n_2 \dots n_N\}$ . There is no restriction limiting a monitor's participation as a neighbour to another monitor. Each node in the network must be associated with atleast one cluster monitor, although some may have more associations. Selection of neighbours by a monitor is given by [44]:

$$\{\cup_{i=1}^x rand(\{n_1\}, \{n_2\} \dots \{n_N\}) \mid x = rand(1..s)\} \quad (5.2)$$

The nodes have hardware resources such as memory, storage space and processors that can be utilised to perform computational tasks. The percentage of idle resources is quite high and the objective is to distribute tasks to to utilize them effectively for parallel computations [47]. We define the Resource capacity of the node as a quantitative measure of the idle resources of a node that may be utilised. It is an aggregated measure of the number of processor instructions per second (in MIPS and GFLOPS) the processor of a node is capable of and the primary memory space that is free and available (in Megabytes). Based on this metric, we assign states to each node [48]:

### Supra-positive ( $R > T+$ )

Resource capacity of the node is much greater than the positive threshold, implying a lot of idle resources.

**Positive ( $R > T+$ )**

Resource capacity of the node is greater than the positive threshold.

**Negative ( $R < T-$ )**

Resource capacity of the node is lesser than the negative threshold.

**Neutral ( $R$  not defined)**

Nodes outside the regional view of a cluster monitor.

A cluster monitor's abstract view of its neighbours is shown in Figure 5.1.

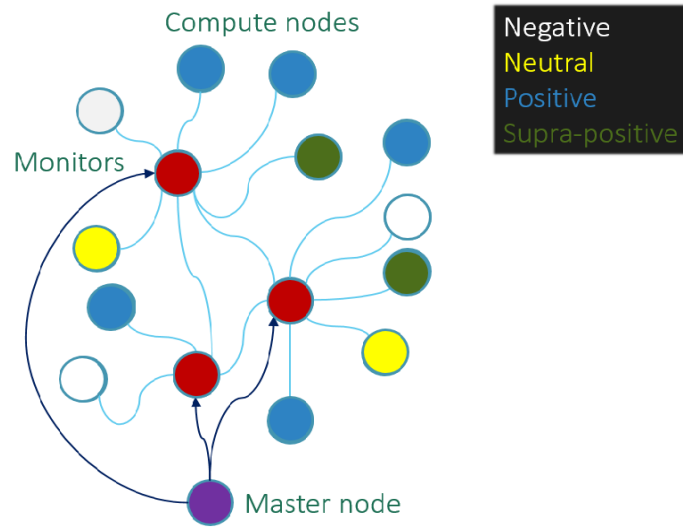


Figure 5.1: Cluster Monitor Region View

These states determine the scheduling of jobs. The states of neighbouring nodes are maintained at the local cache of each cluster monitor. An abstract view of the local cache is given in Table 5.1. However, these states are not live updated by control message exchanges between nodes after preliminary initialization. This design choice is primarily used to address the concern of battery drain in devices due to constant network interaction. Instead, calculated guesstimates of approximate time for execution of jobs are made to predict resource utilization affecting node states in the local cache of the cluster monitor. The Execution time remaining field in the cache is updated by calculations:

$$Q_{\text{new}} = Q + \frac{J_i}{R \times T_i} \quad (5.3)$$

$F$  is deduced by the compile time size of the job process in memory. Dynamic heap memory allocation cannot be predicted, but can only be assumed to be linear

at max with respect to the size of the input to the job. This is easily incorporated in the final estimate. The node states are thus updated in the local monitor caches on the basis of these locally made predictions.

Table 5.1: Cluster Monitor Cache

ID	State	Resource Capacity	Exec. Time Remaining
2	Supra-Positive	$10^4$ MIPS, 300 MB	2s
4	Positive	2 GFLOPS, 700 MB	0.3s
7	Negative	$10^7$ MIPS, 1600 MB	1.76s

Two situations have been identified where the cache would need to be refreshed with live statistics.

### Node failure

Due to the inherent possibility of disconnections in wireless networks, a node failure can be viewed as a disconnection of a node from the network for a time period greater than a permissible limit. Temporary loss of a node from the network must be accounted for before declaring it a failure. Since, a node may be under the purview of multiple cluster monitors, a collation of notifications can be used to identify a failed node. A node that cannot be contacted by multiple supervising cluster monitors implies that the probability of the node having failed is high and can thus be treated as such. However, for all nodes in general, if a node stays disconnected over a defined permissible limit, it is treated as a failure as well. Unfinished jobs are then reassigned to other nodes for completion [50]. A new cluster monitor may also be elected if the outgoing node is a monitor itself. In case the master node in the cluster itself goes down, one of the cluster monitors takes over the role of the master node. This is made easy by the fact that a single system image at the application level is running on all nodes.

### Actual performance dips below minimum expected performance

The master node of the wireless ad hoc cluster has a performance monitor running in the background keeping track of the job execution statistics and other cluster performance statistics mentioned in the next section. If the performance falls short of a minimum amount of expected performance, we can conclude that either there are severe networking issues or that local cache and state information is incorrect, leading to constant request rejections. In this case, the entire network nodes would need to be refreshed to reflect consistent state information once again.

Job scheduling is done in two phases - the master node assigns jobs to the cluster monitors, then, the cluster monitors are responsible for further delegation of jobs to neighbouring nodes. The cluster monitors assign jobs to neighbouring nodes on the basis of priority, which is in turn decided by the states of the nodes. The algorithm is given in Figure 5.2.

<p><b>Input:</b> Neighbour Node States  <b>Output:</b> Job assignments, Node State Updates</p> <pre> 1. priority_queue&lt;Node_State&gt; q; 2. for all neighbours 3.   q.push(Node_State) 4. while(J : Incoming Jobs) 5.   Node_State a = q.pop() 6.   a.assign(J) 7.   Update_State(a) 8.   q.push(a) 9. for all neighbours in q 10.  Check (idle resources at Node) 11.  Increment Node_State priority </pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Figure 5.2: Resource Scheduling Algorithm

A dynamic priority queue is maintained at each cluster monitor. Node states are prioritized in the manner Supra-positive > Positive > Negative. However, the nodes are not always in constant state throughout and change according to the jobs assigned to the nodes, the time for which a node has not been allocated jobs and the processing capacity of nodes. Thus the priority of nodes change according to the above factors to get better utilization of all the resources that make up the wireless cluster.

## 5.2 TRUST ESTABLISHMENT MODEL

The wide array of threats that may affect wireless ad hoc clusters negate the possibility of having a singular mechanism to implement the required security. Instead, several security policies may be enforced to achieve the desired effect. A policy proposal to provide incremental increase in access permissions based on the current trust rating of cluster nodes is suggested to prevent security vulnerabilities. Nodes entering a cluster may be registered in one of two possible ways: self-registration of nodes or administrative registration [49]. An implicit inference is that nodes registering in the latter manner can be considered more trustworthy than nodes registering using the former method at the instance of time under consideration. A global trust table consisting of quantitative trust parameters,

representing the degree of trust when aggregated, for each node in the cluster is maintained. Initial values for certain parameters are set for joining nodes, considering a scheme where nodes registered in the administrative way have higher initial values than those of the self-registered nodes. A parameter among several others in the table may be the ratio of number of tasks assigned to number of tasks completed as shown in the equation:

$$Trust\ Parameter(N_k) = \frac{Number\ of\ tasks\ successfully\ completed(n)}{Total\ number\ of\ tasks\ assigned(T)} \quad (5.4)$$

Nodes having the maximum permissible value of 1 will be comparatively more trustworthy than other selfish nodes with lower (approaching 0) parameter values. The parameters are updated at regular intervals by mining collected job statistics for individual nodes and the aggregated values over several intervals consolidates the trust rating of cluster nodes. Based on these values, the nodes may be categorized as trusted, partially trusted or untrusted . Untrusted nodes are the most probable offenders and suitable action may be taken to remove them. Trusted nodes may be assigned higher access rights (for e.g. to participate in the election of cluster monitors) and priorities (for e.g. critical task assignment involving critical data) as required. Thus, the higher the trust rating, the more the privileges that may be allotted.

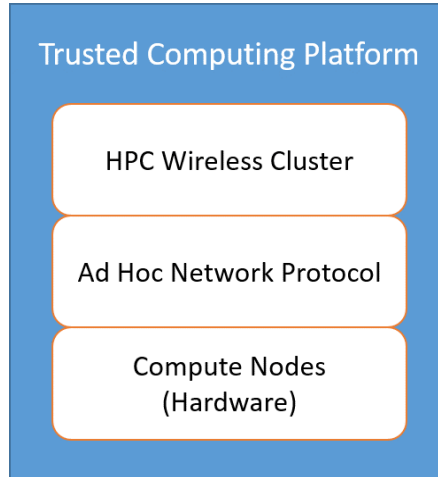


Figure 5.3: General Architecture for Building Trusted Wireless Clusters

A group of trusted compute nodes would lead to the formation of a trust platform, which would ensure better security, reliability and overall performance. The cluster should be proactive in selecting legitimate compute nodes to be part of the cluster and weeding out the malicious nodes. A general architecture for building trusted wireless cluster networks is depicted in Figure 5.3.



# Chapter 6

## EXPERIMENTAL ANALYSIS

### 6.1 BUILDING A BEOWULF CLUSTER

The initial step of the implementation was an attempt to build a Beowulf Cluster from scratch using the POP architecture. The cluster was set up using two cluster nodes, with one of them acting as the master node. The laptops that acted as the nodes had configurations of i3 processor, 2GB RAM and i3 processor, 4GB RAM respectively. A Network File System (NFS) was established to allow the nodes to access shared storage.

After the successful creation of the cluster cell, a parallel mergesort algorithm to sort numbers using the OpenMPI libraries for multithreaded programming was developed. The mergesort problem was given inputs of 1 million numbers and 10 million numbers. The execution times were compared with respect to a normal sequential mergesort algorithm running on a single computer. The results obtained are shown in Table 6.1.

Table 6.1: Execution Statistics of Sequential and Parallel Mergesort

	Parallel Algorithm	Sequential Algorithm
Mergesort (1 million)	0.5 s	5 s
Mergesort (10 million)	5 s	249 s

The results show a considerable performance boost gained by the cluster implementation. The implementation language choice was C programming language. The screenshot corresponding to the execution of mergesort on the beowulf cluster is shown in Figure A.1. The task of sorting was split amongst the two nodes, and then the results were aggregated at the master node. The performance gain observed shows the processing capability inherent in the collaboration of hardware.

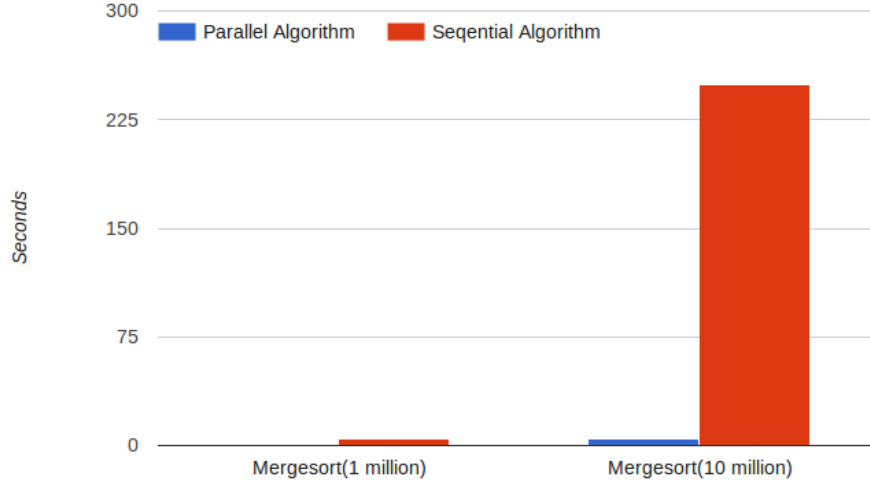


Figure 6.1: A Graphical Representation of Table 6.1

## 6.2 SIMULATION RESULTS

As an initial step, we have simulated our algorithmic strategy to evaluate its performance with respect to a few performance metrics we identified as being crucial to gauge the effectiveness of the proposal. The algorithm was implemented in C++ using the standard POSIX library interface. The various nodes in the ad hoc cluster network were simulated as forked processes from the main master process, each with its own memory space cache definitions. State information between nodes was passed by allocating a global common shared memory space.

The simulation was tested by providing an input of 100 jobs and 500 jobs of equal size each taking roughly 5000 microseconds to execute. The main objective of the simulation is to measure the performance of the local caches that are updated by guesstimates defined in the previous section. The cluster size was also varied to initially account for 100 nodes in the ad hoc network, and then increased to approximately 500 nodes. To incorporate the dynamic and fluctuating nature of ad hoc networks, at random intervals of time, certain operations were counted as being unsuccessfully completed, thus affecting the miss count, which is described next.

Two different kinds of simulations were performed with respect to the proposed algorithmic strategy:

### Activity simulation

This simulation was developed to model the assignment and actual processing of activities in the cluster. The calculation of  $\pi$  was split into increments to be

performed in an interleaved fashion among the cluster computational nodes. The cache node state assignments for the various nodes were varied to view the effect of job scheduling.

A cluster monitor application submission portal as shown in figure A.2 is used to simulate the process of submitting a job for processing. In this case, the job is fixed and it is the calculation of the value of  $\pi$ . The output of the processing is displayed in the same user window. The submitted job is then passed to the Resource Scheduler (Figure A.3) which is responsible for assigning the sub tasks to the nodes in the cluster. Based on the node states in the cluster monitor register cache, the nodes are assigned sub tasks to execute. A sample execution is shown in Figures A.4 and A.5.

### **Performance simulation**

This simulation was developed to measure certain performance metrics of the proposed scheduling strategy. The performance metrics measured were Hit/Miss Ratio and Starvation Deadline. A sample screenshot of the observations from the simulation run on an ad hoc network consisting of 100 compute nodes with 500 jobs assigned for execution is shown in Figures A.6, A.7 and A.8.

We evaluated the proposed algorithm with respect to the following performance metrics:

#### **Hit/Miss Ratio**

This is defined as the ratio of number of correct cache entry hits to the number of overall cache references. However, a miss is not a loss as there is information that is inferred from it used to update the local cache at the cluster monitor. Thus, a hit refers to useful message exchanges with respect to the outside viewer and miss is a useful message exchange with respect to the ad hoc cluster state updates. The results of the simulation is shown in Figures 6.2 and 6.3.

#### **Starvation Deadline**

We define this metric as the overall time period upto which a node is not assigned any task and sits idle in the view of the cluster. The dynamic priority queue introduces the possibility of certain nodes with minimal resources to contribute to always be relegated to the bottom of the queue. To overcome this situation, priorities of nodes are slowly increased over time to increase visibility of the node

during job assignment. The maximum starvation time for a node in a cluster during simulation runs is shown in Figures 6.2 and 6.4.

	No. of Nodes	No. of Jobs	Approximate Neighbour Count							
			10	15	18	20	25	29	37	45
HitMiss ratio(%)	100 Nodes	100 Jobs	97.52	94.73	96.8	90.34	90.91	92.44	95.87	93.64
Starvation-deadline $\mu$ s			2384.572	1859.138	1988.528	1976.544	1813.669	2256.716	2029.535	1939.572
HitMiss ratio(%)	100 Nodes	500 Jobs	94.85	94.43	97.35	96.71	96.79	96.53	94.74	96.05
Starvation-deadline $\mu$ s			4137.711	4856.905	9454.288	13225.876	3440.999	9386.521	6012.485	11561.605
HitMiss ratio(%)	500 Nodes	100 Jobs	71.05	92.13	90.82	95.72	94.15	88.41	89.84	93.98
Starvation-deadline $\mu$ s			11084.57	14559.78	3149.366	14300.591	7766.666	6028.845	8216.866	6490.1
HitMiss ratio(%)	500 Nodes	500 Jobs	96.48	95.75	90.91	95.69	96.22	96.06	96.73	94.76
Starvation-deadline $\mu$ s			41174.501	35846.862	58253.532	62226.75	47420.942	45654.518	59575.386	69895.108

Figure 6.2: Hit/Miss Ratio Simulation Results

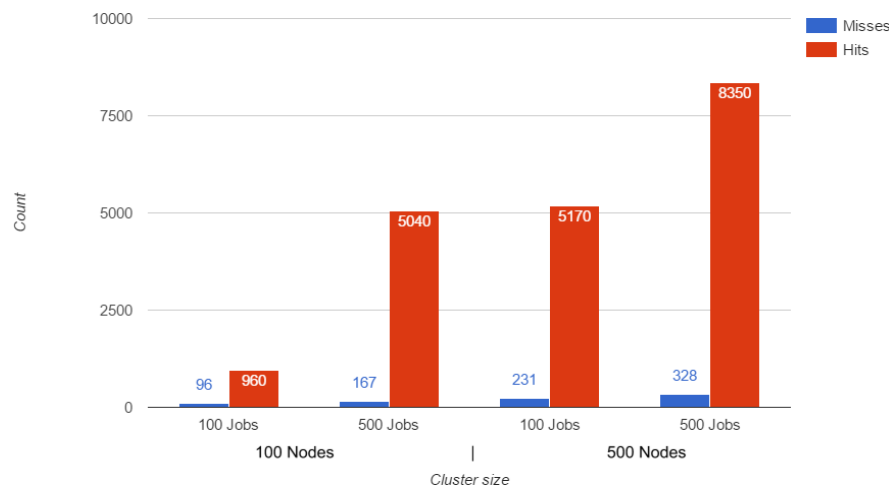


Figure 6.3: Hit/Miss Count Chart

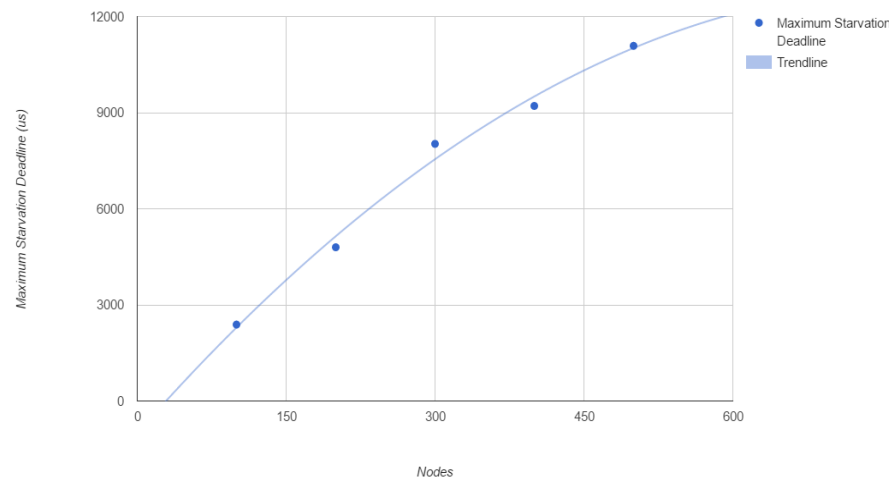


Figure 6.4: Max Starvation Deadline Trend Chart

## Average Resource Utilization

This is a measure of actual practical utilization to theoretical capacity [50]. The value will always range between 0 and 1 and is defined mathematically as:

$$ARU = \frac{\sum J_i}{Q \times t} \quad (6.1)$$

The results obtained are promising with respect to providing good QoS, but would need further careful considerations with respect to several idiosyncrasies that physical implementations would naturally associate with. Culminating, a benchmark simulator has been developed as shown in Figure A.9 that would enable for comparison with other approaches with respect to various performance parameters.

## Battery Usage

The drainage of battery reserves is a very key aspect of the study. The live test bed implementation of the wireless ad hoc high performance cluster was compared with an equivalent wired cluster counterpart and the battery drainage over time was observed with the usage of a battery life monitor. The results are shown in Figure 6.5. We observe that the battery consumption is approximately 7.6% more than the wired cluster over multiple tests lasting for a duration of upto 60 mins of continuous usage.

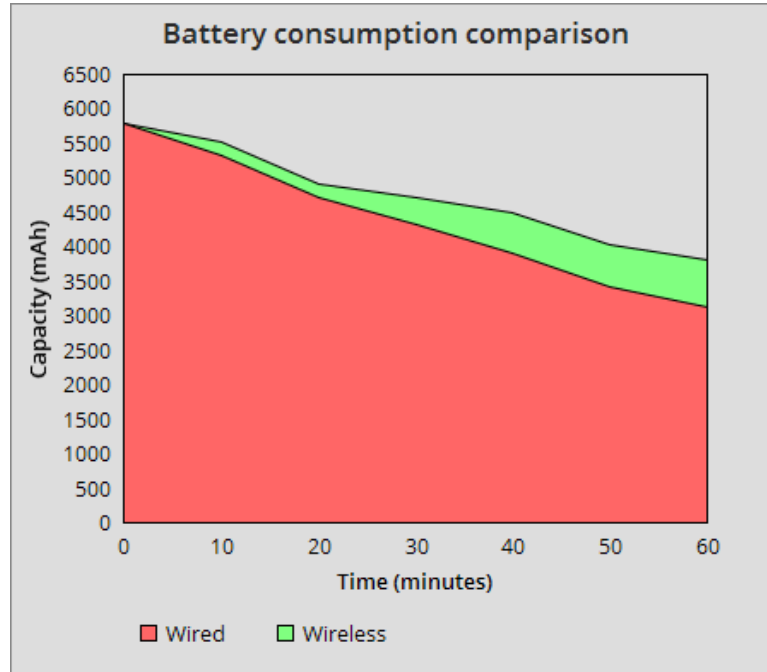


Figure 6.5: Live Test Bed Battery Consumption Benchmark Comparison

### 6.3 DEMONSTRATION: INFINITE FUNNEL ATTACK IN A WIRELESS CLUSTER

The infinite funnel attack was carried out in a simulated environment using NS3 simulator. The AODV routing protocol was utilised as the backbone routing protocol. By compromising a compute node connected to a wireless cluster, the routing tables were modified such that the destination was rendered unreachable from the sending node. The routing tables of all compute nodes were constantly populated with falsified large integer value RREQ sequence numbers corresponding to the latest routing information and best path to the destination from any node. However, all the paths lead through the malicious node, which simply drops any packet received by it. The packet trace files are shown below in Figures 6.6 and 6.7. The implementation screenshots corresponding to Figures A.10 and A.11 are depicted in the Appendix.

95	40.011427	00:00:00_00:00:02	Broadcast	ARP	64 Who has 10.1.2.1?
96	40.011477	00:00:00_00:00:01	00:00:00_00:00:02	ARP	64 10.1.2.1 is at 00:00:00
109	40.050886	10.1.2.2	10.1.2.1	ICMP	92 Time-to-live exceeded
99	40.023376	10.1.2.2	10.1.2.4	UDP	1104 Source port: 49153
107	40.049574	10.1.2.2	10.1.2.4	UDP	1104 Source port: 49153
111	40.061584	10.1.2.2	10.1.2.4	UDP	1104 Source port: 49153
113	40.094864	10.1.2.2	10.1.2.4	UDP	1104 Source port: 49153
115	40.128144	10.1.2.2	10.1.2.4	UDP	1104 Source port: 49153

Figure 6.6: Malicious Node Intercepting Packets

81	39.988001	00:00:00_00:00:03	Broadcast	ARP	64 Who has 10.1.2.1?
85	39.994601	00:00:00_00:00:03	Broadcast	ARP	64 Who has 10.1.2.4?
86	39.994651	00:00:00_00:00:04	00:00:00_00:00:03	ARP	64 10.1.2.4 is at 00:00:00
90	40.003039	00:00:00_00:00:03	00:00:00_00:00:02	ARP	64 10.1.2.3 is at 00:00:00
93	40.025765	10.1.2.2	10.1.2.4	UDP	1104 Source port: 49153

Figure 6.7: Destination Node Packet Loss

The consequences of such an attack are severe as important data that must be communicated or computed is not reaching the intended target. Thus the performance of the cluster declines and ceases to be useful. This implies that special consideration of the security issues must be taken seriously. The proposed trust establishment model seeks to address such issues at a fundamental level in order to reduce the possibility of such attacks and vulnerability exploits.

# Chapter 7

## CONCLUSION

### 7.1 SUMMARY

An algorithmic approach to share resources in a wireless ad hoc high performance cluster has been presented. With a key focus on reducing the consumption of battery in the network nodes, we propose an algorithm that uses guesstimates to predict node states. The overall reduction in communication along with "worst case zero loss" performance of the entire cluster makes this proposal effective, as indicated by preliminary and extended simulation results.

Specific security threats that have serious consequences to the performance and functioning of a wireless ad hoc high performance cluster are analysed and some counter measures that must be in place to negate them have been specified. Nonetheless, there is no integrated mechanism to prevent all attacks. Specific protection measures have to be taken against different attacks. All attacks studied and their ramifications to the security of wireless clusters spreads awareness of providing adequate security for wireless ad hoc high performance clusters. Wireless computation clusters and grids is a branch that is getting more concentration in the recent few years as the processing capability of mobile nodes increase exponentially and communication technologies advance. Conventional security mechanisms cannot be applied directly and must be suitably extended or scratched out in favor of other methods to ensure requisite security requirements are met. A policy for establishing a trusted wireless cluster compute network is proposed for establishing a baseline trust of nodes in the cluster. Employing such a strategy is a big step forward in reducing probable security vulnerabilities.

# Bibliography

- [1] Cluster computing. (2015, July 30). [http://en.wikipedia.org/wiki/Computer\\_cluster](http://en.wikipedia.org/wiki/Computer_cluster)
- [2] Buttyán, L., and Hubaux, J. P. (2003). *Report on a working session on security in wireless ad hoc networks*. ACM SIGMOBILE Mobile Computing and Communications Review, 7(1), 74-94.
- [3] Sanjay P. Ahuja and Jack R. Myers. (2006). *A Survey on Wireless Grid Computing*. The Journal of Supercomputing. 37. 3-21.
- [4] Aksenti Grnarov, Bekim Cilku, Igor Miskovski, Sonja Filiposka and Dimitar Trajanov. (2008). *Grid Computing Implementation in Ad Hoc Network*. Advances in Computer and Information Sciences and Engineering. 196-201.
- [5] C. Lin, and Y. Tseng. (2004, October). *Structures for in-network moving object tracking in wireless sensor networks*. In Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on (pp. 718-727). IEEE.
- [6] T. Phan, L. Huang, and C. Dulan. (2002, September). *Challenge: Integrating wireless devices into the computational grid*. In Proceedings of the 8th annual international conference on Mobile computing and networking. pp. 271-278. ACM, 2002.
- [7] A. Hesseldahl. TVs join the wireless grid. (2003, February). <http://www.forbes.com/2003/09/>
- [8] L. McKnight, J. Howison, and S. Bradner. (2004, July). *Distributed resource sharing by mobile, nomadic, and fixed devices*. IEEE Internet Computing.
- [9] M. Gaynor, S. Moulton, M. Welsh, E. LaCombe, A. Rowan, and J. Wynne. (2004) *Integrating wireless sensor networks with the grid*, IEEE Internet Computing, 8(4), 32-39.
- [10] C. Lin, and Y. Tseng. (2004, October). *Structures for in-network moving object tracking in wireless sensor networks*, First International Conference on Broadband Networks. Proceedings. First International Conference on (pp. 718-727). IEEE.
- [11] A. Agarwal, D. Norman, and A. Gupta. (2004) *Wireless grids: approaches, architectures and technical challenges*. MIT Sloan School of Management.
- [12] S. Ghandeharizadeh. (2004, October) *Science of continuous media application design in wireless networks of mobile devices*. First International Conference on Broadband Networks on (pp. 506-515).
- [13] Chao-Tung Yang, Chuan-Lin Lai. (2004). *Apply Cluster and Grid Computing on Parallel 3D Rendering*. IEEE International Conference on (Vol. 2, pp. 859-862). IEEE.
- [14] G. Pfister. (1998). *In Search of Clusters*. Prentice Hall PTR, NJ. 2nd Edition.
- [15] Super Computing.(2015, August). [https://en.wikipedia.org/wiki/Supercomputing\\_in\\_India](https://en.wikipedia.org/wiki/Supercomputing_in_India).



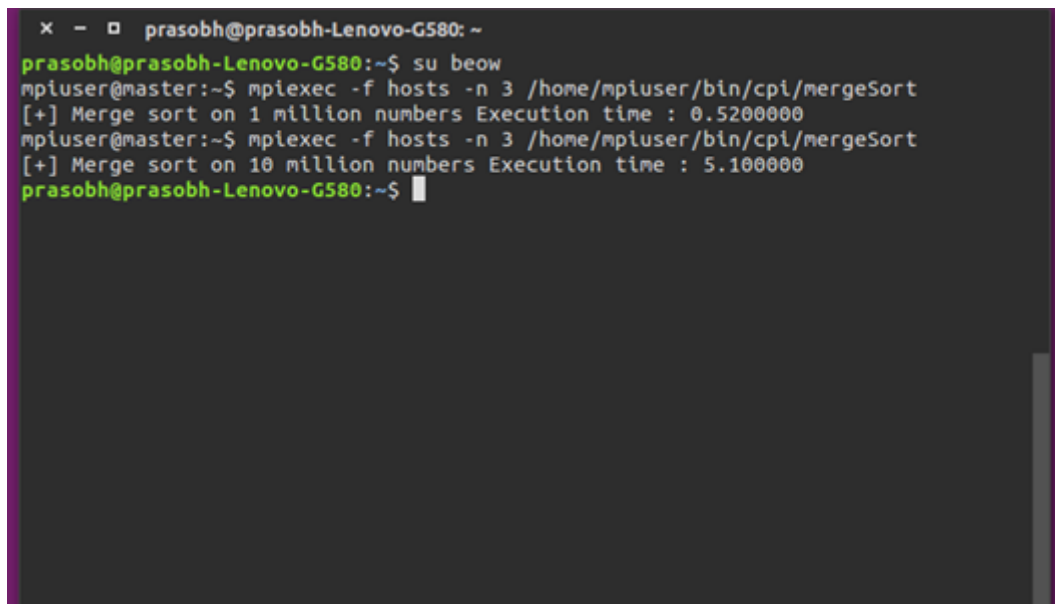
- [16] K. Hwang and Z. Xu. (2001). *Scalable Parallel Computing: Technology, Architecture, Programming*. Scalable Computing: Practice and Experience, 2(1).
- [17] MPI Forum. (2014). <http://www.mpi-forum.org/docs/docs.html>.
- [18] The Linux Documentation Project. (2014). <http://sunsite.unc.edu/mdw/linux.html>.
- [19] The Beowulf Project. (2013). <http://www.beowulf.com>.
- [20] C. Kobhio, S. Pierre and A. Quintero. (2006, Jan). *Redundancy Schemes for High Availability Computer Clusters*. Journal of Computer Science, Vol. 2, No. 1, pp 33-47.
- [21] Top Clusters. (2014). <http://www.topclusters.org>
- [22] S. Olivier and J. Prins.(2008, March) *Scalable Dynamic Load Balancing Using UPC*. Proc. of the 37th International Conference on Parallel Processing, pp 123-131, 08-12.
- [23] C. J. Lai and W. Polak. (2006, November). *A Collaborative Approach to Stochastic Load Balancing with Networked queues of autonomous service clusters*. Proc. of 2nd IEEE Collaborative Computing Conference, pp 1-817-20.
- [24] Z. Chen, M. Yang, G. Francia and J. Dongarra. (2007, March) *Self Adaptive Application Level Fault Tolerance for Parallel and Distributed Computing*. Proc. of IEEE International Parallel and Distributed Processing Symposium, pp 414-421, 26-30.
- [25] P. Pop, V. Izosimov, P. Eles and Z. Peng. (2009, March) *Design Optimization of Time-and Cost- Constrained Fault- Tolerant Embedded Systems With Checkpointing and Replication*. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 17, No.3, pp 389-402.
- [26] Pinheiro, E., Bianchini, R., Carrera, E. V., and Heath, T. (2001, September). *Load balancing and unbalancing for power and performance in cluster-based systems*. In Workshop on compilers and operating systems for low power (Vol. 180, pp. 182-195).
- [27] S. S. Manvi and M. N. Birje. (2010). *A Review on Wireless Grid Computing*, International Journal of Computer and Electrical Engineering, Vol. 2.
- [28] Mohamed Aissa and Abdelfettah Beighith. (2014). *Quality of Clustering in mobile Ad Hoc networks*. 5<sup>th</sup> International Conference on Ambient Systems, Networks and Technologies, 32, pages 245-252.
- [29] Alan D. Amis and Ravi Prakash. (2000). *Load-Balancing Clusters in Wireless Ad Hoc Networks*. Application-Specific System and Software Engineering Technology, pages 25-32.
- [30] Li Xia and Jill Slay. (2004). *Securing Wireless Ad Hoc Networks:Towards A Mobile Agent Security Architecture*. 2<sup>nd</sup> Australian information Security Management Conference.
- [31] Stanislav Kurkovsky and Bhagyavati. (2009). *Wireless Grid Enables Ubiquitous Computing*. ISCA International Conferences on Parallel and Distributed Computing (and Communications) Systems. pages 399-404.
- [32] Zhi Wang et al. (2005, November). *Wireless Grid Computing over Mobile Ad-Hoc Networks with Mobile Agent*. International Conference on Semantics, Knowledge and Grid (SKG 2005), 27-29.
- [33] Hu, Y. C., Perrig, A. and Johnson, D. B. (2006). *Wormhole attacks in wireless networks*. Selected Areas in Communications, IEEE Journal on, 24(2), 370-380.
- [34] Shivaramu K.N, Prasobh P.S and Nagaraj Poti (2016). *A Survey on Security Vulnerabilities in Wireless Ad Hoc High Performance Clusters*. Elsevier Procedia Technology Journal. In Press

- [35] Hu, Y. C., Perrig, A. and Johnson, D. B. (2003, April). *Packet leashes: a defense against wormhole attacks in wireless networks*. In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies (Vol. 3, pp. 1976-1986). IEEE.
- [36] Levine, B. N., Shields, C. and Margolin, N. B. (2006). *A survey of solutions to the sybil attack*. University of Massachusetts Amherst, Amherst, MA.
- [37] Tseng, F. H., Chou, L. D. and Chao, H. C. (2011). *A survey of black hole attacks in wireless mobile ad hoc networks*. Human-centric Computing and Information Sciences, 1(1), 1-16.
- [38] Singh, U. K., Patidar, J. and Phuleriya, K. C. (2015). *On Mechanism to Prevent Cooperative Black Hole Attack in Mobile Ad Hoc Networks*.
- [39] Gupta, A., Patel, B., Rana, K. and Pradhan, R. (2015). *Improved AODV Performance in DOS and Black Hole Attack Environment*. In Computational Intelligence in Data Mining-Volume 2 (pp. 541-549). Springer India.
- [40] Kremien, O., Michael, K. and Irit, E. (1999). *Preserving Mutual Interests in High Performance Computing Clusters*. INFORMATICA-LJUBLJANA-, 23, 41-48.
- [41] Barbeau, M., Hall, J. and Kranakis, E. (2006). *Detecting impersonation attacks in future wireless and mobile networks*. In Secure Mobile Ad-hoc Networks and Sensors (pp. 80-95). Springer Berlin Heidelberg.
- [42] Vasco.com. (2015, December). *VASCO Strong Two Factor Authentication - Hardware Authentication*.
- [43] Almeshekah, M. H., Atallah, M. J. and Spafford, E. H. (2015). *Defending against Password Exposure using Deceptive Covert Communication*.
- [44] Shivaramu K.N, Nagaraj Poti and Prasobh P.S (2016). *A Resource Sharing Strategy Using Guesstimates For Wireless Ad Hoc High Performance Clusters*. 6th IEEE International Advance Computing Conference (IACC-2016). In Press
- [45] Carns, P.H. et al. (1999, March). *An Evaluation of Message Passing Implementations on Beowulf Workstations*, Aerospace Conferences, IEEE.
- [46] O. Kremien, J. Kramer and J. Magee.(1993, August). *Scalable, Adaptive Load Sharing Algorithms*. IEEE Parallel and Distribute Technology, pages 62-70.
- [47] Ridge, D. et al. (1997, February). *Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs*. Aerospace Conference, IEEE.
- [48] O. Kremien, M. Kemelmakher and I. Eshed. (1999). *Preserving Mutual Interests in High Performance Computing Clusters*. Informatica Journal, vol. 23(1).
- [49] Sivagurunathan S and Prathapchandran K (2015, March). *Trust and Cluster based Security architecture in MANET for a Collaborative Computing Environment*. In Proceedings of the UGC Sponsored National Conference on Advanced Networking and Applications.
- [50] P. Keerthika and P. Suresh. (2015). *A Multiconstrained Grid Scheduling Algorithm with Load Balancing and Fault Tolerance*. The Scientific World Journal.
- [51] Naeem, T., and Loo, K. K. (2009). *Common security issues and challenges in wireless sensor networks and IEEE 802.11 wireless mesh networks*. 3; 1.

# Appendices

# Appendix A

## Screenshots



```
prasobh@prasobh-Lenovo-G580: ~  
prasobh@prasobh-Lenovo-G580:~$ su beow  
mpiuser@master:~$ mpiexec -f hosts -n 3 /home/mpiuser/bin/cpl/mergeSort  
[+] Merge sort on 1 million numbers Execution time : 0.5200000  
mpiuser@master:~$ mpiexec -f hosts -n 3 /home/mpiuser/bin/cpl/mergeSort  
[+] Merge sort on 10 million numbers Execution time : 5.100000  
prasobh@prasobh-Lenovo-G580:~$
```

Figure A.1: Beowulf Cluster Mergesort Execution

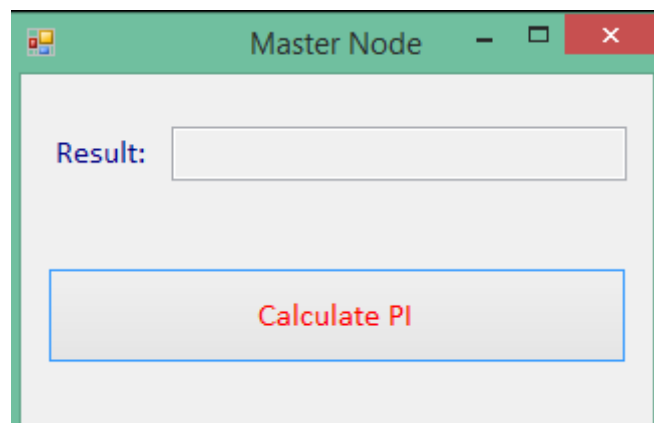


Figure A.2: Cluster Monitor Application Submission

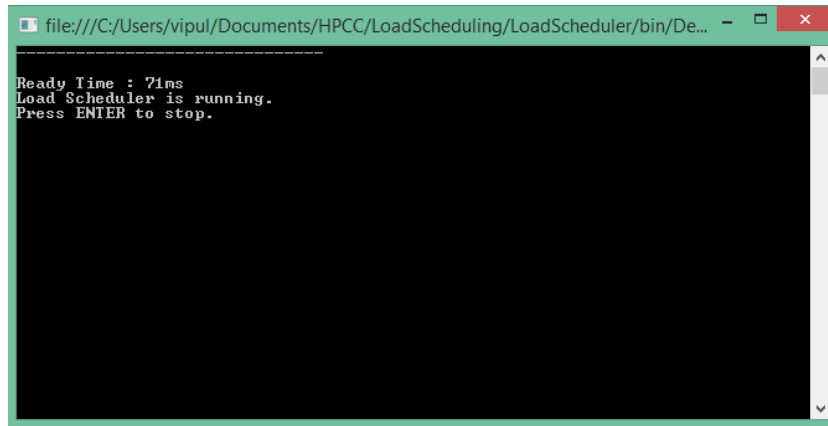


Figure A.3: Resource Scheduler

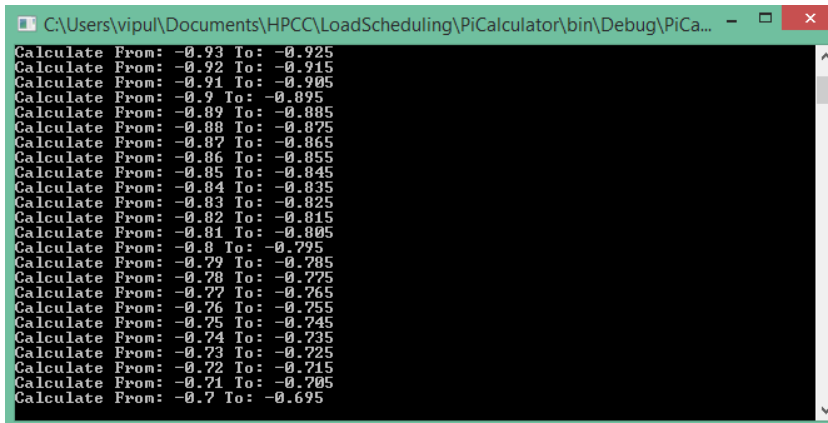


Figure A.4: Node 1 Job Execution

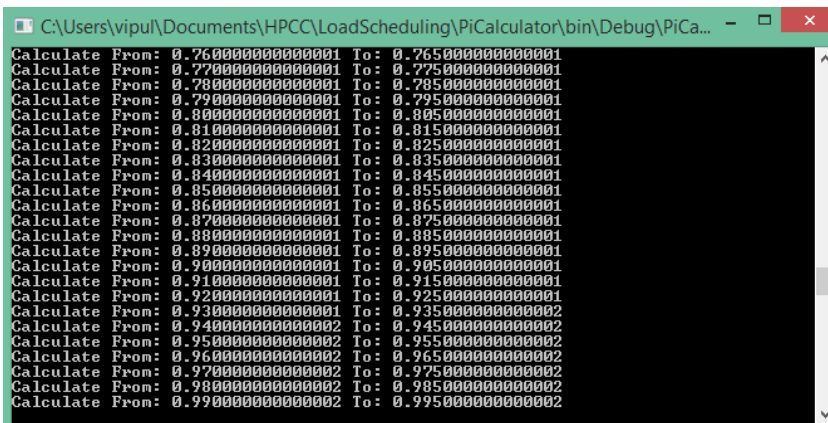


Figure A.5: Node 2 Job Execution

```

nagaraj@nagaraj-MS-7798:~$ g++ sim.cpp -lpthread -lrt -std=c++11
nagaraj@nagaraj-MS-7798:~$ ./a.out
Time elapsed: 0.000000

-----
PERFORMANCE ANALYSIS
-----
Please specify compute node count: 100
Time elapsed: 2.000000

-----
Node initialization statistics:
Node ID - 1      Node state - Positive
Node ID - 2      Node state - Positive
Node ID - 3      Node state - Negative
Node ID - 4      Node state - Neutral
Node ID - 5      Node state - Supra positive
Node ID - 6      Node state - Supra positive
Node ID - 7      Node state - Supra positive
Node ID - 8      Node state - Neutral
Node ID - 9      Node state - Positive
Node ID - 10     Node state - Negative
Node ID - 11     Node state - Supra positive
Node ID - 12     Node state - Negative
Node ID - 13     Node state - Neutral
Node ID - 14     Node state - Neutral
Node ID - 15     Node state - Supra positive

```

Figure A.6: Performance Simulation Node Initialization

```

Neighbours - 4, 6, 28, 31, 33, 55, 67, 68, 88, 89,
Cluster monitor - 12
Neighbours - 11, 13, 15, 26, 27, 40, 56, 58, 77, 80,
Cluster monitor - 13
Neighbours - 22, 47, 49, 60, 61, 65, 72, 81, 95, 97,
Cluster monitor - 14
Neighbours - 1, 8, 21, 30, 34, 59, 74, 76, 78, 90,
Cluster monitor - 15
Neighbours - 10, 14, 19, 42, 44, 48, 82, 84, 85, 91,
Cluster monitor - 16
Neighbours - 12, 18, 24, 32, 36, 45, 53, 73, 83, 86,
Cluster monitor - 17
Neighbours - 17, 25, 29, 35, 37, 57, 62, 63, 92, 98,
Cluster monitor - 18
Neighbours - 2, 5, 9, 20, 50, 54, 66, 69, 75, 87,
Cluster monitor - 19
Neighbours - 38, 39, 43, 51, 52, 64, 71, 94, 96, 100,
Cluster monitor - 20
Neighbours - 3, 7, 16, 23, 41, 46, 70, 79, 93, 99,

```

Figure A.7: Performance Simulation Cluster Monitor Assignment

```

-----
Neighbour approx count - 37
Hits - 1580
Misses - 68
Hits/Miss ratio - 95.87%
-----
Neighbour approx count - 45
Hits - 1310
Misses - 89
Hits/Miss ratio - 93.64%
-----
348 GB Volume
-----
Starvation-Deadline performance statistics
-----
Network
Neighbour approx count - 10
Starvation deadline - 2384.572us
-----
Connect to Server
Neighbour approx count - 15
Starvation deadline - 1859.138us
-----
Neighbour approx count - 18

```

Figure A.8: Performance Simulation Statistics

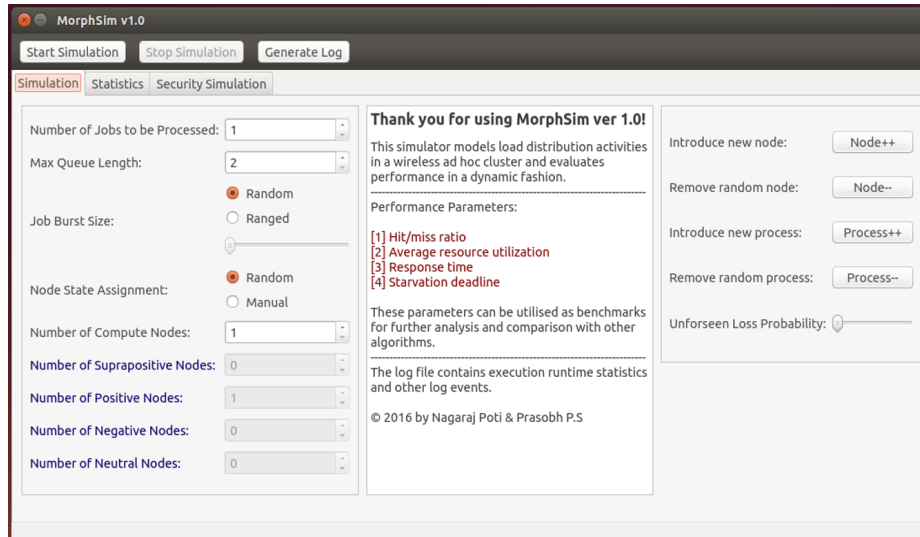


Figure A.9: Benchmark Simulator

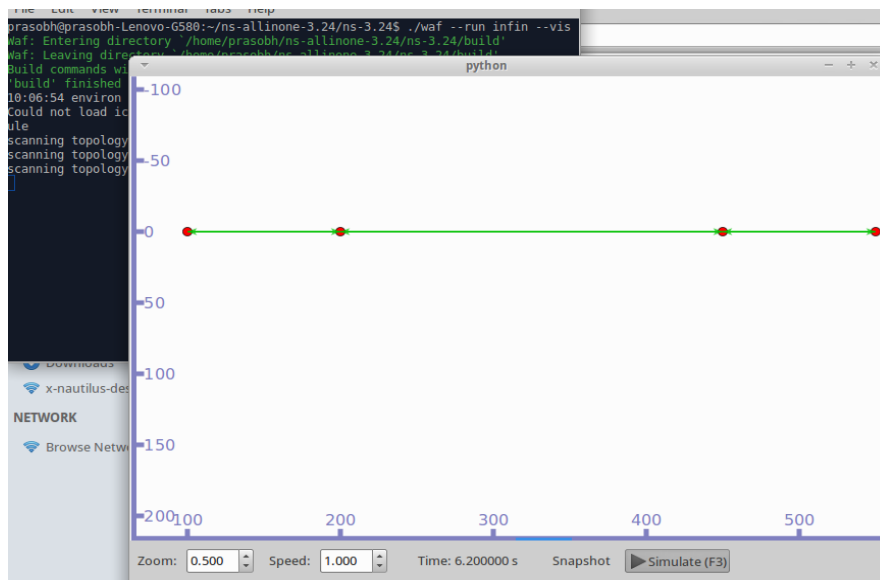


Figure A.10: NS3 Infinite Funnel Attack Visualization

```

prasobh@prasobh-Lenovo-G580:~/ns-allinone-3.24/ns-3.24$ ./waf --run infin
Waf: Entering directory '/home/prasobh/ns-allinone-3.24/ns-3.24/build'
Waf: Leaving directory '/home/prasobh/ns-allinone-3.24/ns-3.24/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (2.251s)
40.0468 1040
Launching Infinite Funnel Attack! Packet dropped . . .
Launching Infinite Funnel Attack! Packet dropped . . .
Launching Infinite Funnel Attack! Packet dropped . . .
Launching Infinite Funnel Attack! Packet dropped . . .
Flow 1 (10.1.2.2 -> 10.1.2.4)
Tx Bytes: 5340
Rx Bytes: 1068
Throughput: 0.174236 Mbps
prasobh@prasobh-Lenovo-G580:~/ns-allinone-3.24/ns-3.24$

```

Figure A.11: NS3 Infinite Funnel Attack Statistics