6-2004

On the Effects of Device Mobility Upon Resource-Sharing in a Simulated Wireless Grid Network

Arris Ray

ON THE EFFECTS OF DEVICE MOBILITY UPON RESOURCE-SHARING IN A SIMULATED WIRELESS GRID NETWORK

Arris Eugene Ray

Columbus State University

The College of Science

The Graduate Program in Computer Science

On the Effects of Device Mobility Upon Resource-Sharing in a Simulated Wireless Grid Network

A Thesis in

Computer Science

by

Arris Eugene Ray

Submitted in Partial Fulfillment of the
Requirements for the
Degree of

Master of Science

June 2004

I have submitted this thesis in partial fulfillment of the requirements for the degree of Master of Science.

5/9/05
Date

[illegible]
Arris Eugene Ray

We approve the thesis of Arris Eugene Ray as presented here.

5/9/05
Date

[illegible]
Dr. Stanislav Kurkovsky,
Associate Professor of Computer Science,
Thesis Advisor

5/9/05
Date

[illegible]
Dr. Bhagyavati,
Assistant Professor of Computer Science,
Thesis Advisor

5/10/05
Date

[illegible]
Chris Whitehead,
Assistant Professor of Computer Science
Member, Thesis Committee

5/9/05
Date

[illegible]
Dr. Wayne Summers,
Professor and Distinguished Chair of Computer Science
Member, Thesis Committee

Abstract

The primary aim of modern computer networks is to permit simplified sharing of resources (e.g., files and data) or to provide access to remote resources (e.g., printers or remote file-systems). Grid networks aim to provide a more sophisticated framework for securely sharing, selecting and/or aggregating a wide-variety of remote resources, whether logical (e.g., files and data) or physical (e.g., remote sensors, processing power or disk storage space). Provided that circuit design continues to improve at the relatively steady pace that it has for the past decade, in addition to standard desktop systems, miniature and mobile devices will soon become substantially significant nodes in extant networks and emerging paradigms of computing, such as grids.

Although modern mobile devices, such as tablets and personal digital assistants, are powerful computers in their own right, they are simply inadequate systems for most computer applications, which tend to target the *de facto* standard of desktop-class computer systems. Mobile devices, by merit of their reduced size and portable nature, are inherently subject to a number of constraints, including the fact that they possess substantially fewer computational resources (e.g. processing power, disk-storage space) than desktop-class systems and that they operate on an independent and limited power supply. Additionally, mobile devices tend to adopt a wireless, rather than wired, means of communication for networking, making them subject to the set of constraints that characterize wireless communication. Wireless communication technologies are generally subject to concerns such as signal interference and unpredictable levels of connection quality or connectivity.

These unique constraints have already required re-evaluation and re-design in various aspects of the Internet, from the network level (e.g. Mobile IP) to the application level (e.g. compact, device-specific Web browsers). Because grid networks are anticipated to build on the existing framework of the Internet and Internet technologies, it seems safe to assume that mobile devices will have a similarly significant impact on grid networks. Unfortunately, the majority of current grid applications target clusters of resource-rich desktop systems for the express purpose of scientific or engineering research and there seems to be relatively little information available on the impact of mobile devices on wireless grid environments.

Thus, the goal of this work is to develop a simple simulation system to investigate the effects of introducing a population of mobile devices into a grid network. This was done to determine the nature and extent of any effects that mobility may have upon the resource-sharing capabilities of a wireless grid network.

Table Of Contents

List Of Figures

List Of Tables

Glossary

AAR	Active Agent Repository. The AAR is a database utilized by the GSS-Broker to store and retrieve the state information of all devices registered with the Grid Network............................ 13, 28, 33, 37, 38, 39, 40, 41, 42
Computational Grid	A Computational Grid is defined here as a Grid Network in which the primary resource available for sharing, selection and aggregation is processing cycles. 2
DSSS	Direct Sequence Spread Spectrum..................................... 9
FHSS	Frequency Hopping Spread Spectrum. 10
GSS-Broker	The GSS-Broker is a server-based administrative application whose primary purpose is to administrate and coordinate resource requests in the GSS network environment. It is composed of two system services, the Brokering Service and the Keep-Alive Service, which co-operate to maintain the stability and efficient operation of the simulated wireless network environment.. iv, 14, 15, 16, 17, 18, 19, 20, 21, 27, 28, 29, 30, 31, 34, 36, 37, 39, 40, 41, 52
Grid Network	Grid Networks enable the aggregation, sharing and selection of a heterogeneous population of resources available within or among networked environments. 1, 2, 3, 5, 8, 13, 15, 16, 43, 55
GSS	Grid Simulator System. The GSS refers to a collection of applications and services that are required to establish, administrate and simulate a wireless Grid Network. It is capable of managing a population of actual, user-controlled devices; creating and administrating a population of autonomous virtual devices; or some combination of both. 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 26, 27, 28, 29, 32, 33, 37, 38, 40, 41, 43, 44, 52, 53, 54, 55
GSS-CA	Grid Simulator System - Client Application. A GSS-CA is a light-weight application which manages communication between client devices and the GSS-Broker......... 13, 14, 29
GSS-M	Grid Simulator System - Module. A GSS-M is an application that may be used to customize, execute and monitor simulation sessions. A GSS-M is predominately intended to allow users to customize and generate a population of autonomous virtual devices to populate the Grid Network. 13, 14, 19, 26, 27, 28
GSS-RC	Grid Simulator System - Remote Control. The GSS-RC is a supplemental application, which may be used to automate and coordinate the execution of multiple GSS-Ms that are distributed across a network.......... 13, 19, 21, 22, 26, 27, 28

HTML	Hyper-Text Markup Language. 6
IP	Internet Protocol....................... 3, 4, 5, 6, 18, 21, 26, 29, 38
Mobile Computing	Mobile Computing describes a paradigm of computing in which mobile devices have access to a shared network infrastructure independent of their physical location.... iv, 8, 55, 57
OFDM	Orthogonal Frequency Division Multiplexing................. 10
PDA	Personal Digital Assitant..iii, 1
RF	Radio Frequency. 9, 11
SOAP	Simple Object Access Protocol... 6
TAT	Task Allocation Table. The TAT is a database utilized by the GSS-Broker to store and retrieve the state information of all tasks that have been initiated in the Grid Network. 13, 28, 34, 35, 36, 37, 40, 41
TCP	Transmission Control Protocol. 5, 6, 15, 18, 26, 29
TIF	Task Initiation Frequency. vi, 22, 26, 44, 45, 46, 47, 48, 49, 50, 51
UDDI	Universal Discovery, Description and Integration...... 6, 7, 8
UDP	Universal Datagram Protocol... 5
VO	Virtual Organization. 3
WLAN	Wireless Local Area Network...................................... 9, 58
WPAN	Wireless Personal Area Network 10
WSDL	Web Services Description Language......................... 6, 7, 8
XML	Extensible Markup Language. ... iv, 1, 5, 6, 7, 18, 28, 30, 40

1. Introduction

Grid networks enable the aggregation, sharing and selection of a heterogeneous population of resources available within or among networked environments. This includes logical resources, such as documents, files and programs, as well as physical resources, such as processing cycles, disk storage space and sensors or other specialized equipment **[15]**. Grid technologies are currently undergoing the process of standardization as working groups attempt to address issues such as authentication, authorization, security and resource management, which are being designed to utilize open-standard Internet protocols and services such as XML Web Services and related technologies **[15]**. For this reason, when the technology has matured, grid networks may be more easily deployed over the current Internet infrastructure, allowing Internet users and developers to more readily adopt and adapt to grid technologies.

The specifications for the Internet Protocol (IP) make no assumption about the types of devices that will implement the protocol stack. Similarly, the specifications for grid protocols will make no assumption about the types of devices to implement the grid protocol stack. This permits grid networks or inter-grid networks to accommodate a heterogeneous population of client devices. Among the wide variety of computers that are capable of accessing the Internet, a typically constrained class is comprised of mobile devices, defined as any portable computational device capable of wireless networking **[14]**; examples of mobile devices include laptops, personal digital assistants (PDA) and smart-phones. Mobile devices typically possess fewer physical resources, limited power supplies and levels of connectivity significantly more unpredictable than desktop-class systems **[29]**. These constraints have presented a substantial challenge to Internet developers, requiring amendments to be made in various specifications of the IP stack [2] from the network layer (Mobile IP) to the application layer (compact, device-specific Web browsers). Therefore, we assume that these constraints will have non-trivial impact on the development of grid networks.

There already exist several grid implementations [10], most of which are primarily being used in high-performance computing and large-scale scientific simulations, such as Legion [26], EcoGrid [7], the Grid Physics Network (GriPhyN) [1], NASA's Information Power Grid [24], Earth Systems Grid [3] and the Network for Earthquake Engineering Simulation Grid (NEESGrid) [30]. Additionally, there are organizations and projects dedicated to a more general approach to grid implementation and development -- most notably, the Globus Alliance. Globus is an effort dedicated to developing a general-purpose toolkit (the "Globus Toolkit" [12]) capable of establishing inter-operable grid networks but, much like the aforementioned grid applications, Globus' aim is high-end scientific and engineering computing [16]. Such Grid applications target clusters of resource-rich desktop systems and are unsuitable for deployment across a network of resource-constrained mobile devices. For this reason, there appears to be little practical data describing the effects of mobile devices and their unique constraints on resource-sharing in grid networks.

In order to determine these effects, we designed a grid infrastructure specifically for mobile nodes communicating through wireless media. To demonstrate the proof of our concept, we then developed, implemented and tested a simulator to emulate the functionality of a computational grid. The simulator was populated by autonomous virtual agents that were modeled upon mobile devices. These virtual devices requested computational resources at random intervals from the network. Carefully chosen properties of the simulator were adjusted prior to simulation runs to more accurately model the unpredictable behavior of real network activity; during the course of a simulation session, logs were generated, permitting careful examination to determine the significance of select variables on the sharing of resources.

In this thesis we describe in detail the motivation for creating the grid simulator, the infrastructure of the grid simulator, and finally, the results obtained from running a series of simulation trials. Based on the results of these experiments, we conclude with observations regarding the effects of mobility on resource sharing in wireless grid networks.

1.1. Grid Networks

Analogous to the phase that characterized information sharing across computer networks prior to the development of the Internet, the potential for generalized resource sharing in current computer networks is hindered by the overwhelming heterogeneity of computer configurations across the Internet, as well as the absence of a coherent set of standards or protocols for generalized resource sharing. In response to these facts, groups such as the Globus Alliance **[16]** have worked to develop a protocol-based architecture for general resource sharing, referred to as grid networks. The primary aim of such networks is to allow flexible, secure and coordinated resource sharing amongst dynamic and heterogeneous collections of individuals, organizations and their resources **[15]**. These collections are referred to as "Virtual Organizations" (VO) **[15]**, a fundamental concept in grid networks that is useful for describing the increasingly dynamic and distributed nature of network resources in contemporary business and scientific ventures.

VOs are defined by a common set of sharing rules that dictate those who can share resources, what resources can be shared, and under what circumstances they may be shared. Ideally, anyone should be able to establish or participate in a VO, which implies that grids are highly interoperable systems. A reliable method for ensuring interoperability in networked environments is the establishment of a standard set of protocols. Consequently, the grid protocol stack was developed and organized similarly to the IP stack. Both protocol stacks specify a hierarchical series of layers wherein each layer represents a collection of related functions necessary for preparing messages for transmission across the network or decoding messages received from the network. Network messages are typically sent by passing packets of bits down the protocol stack from a user to the underlying physical network; conversely, messages are received by passing packets of bits up the protocol stack from the underlying network to a user.

1.1.1. Grid Protocol Architecture

A brief introduction to the layers of the grid protocol architecture follows, with comparative descriptions of related layers in the IP architecture **[15]**.

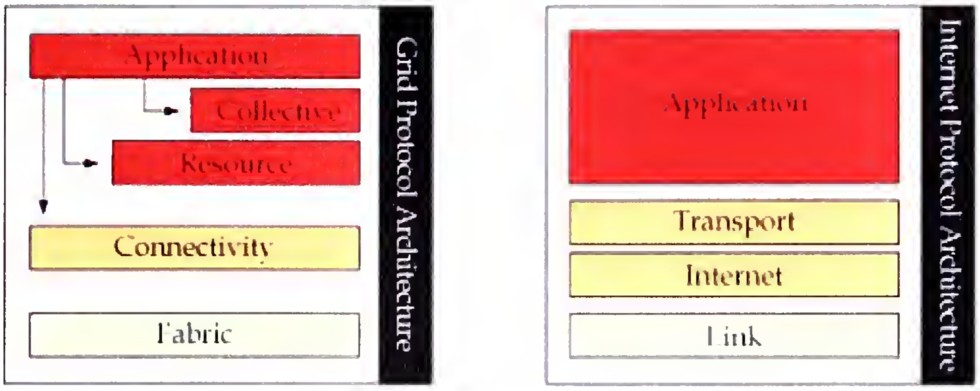


Figure 1. Grid Protocol and the Internet Protocol Architecture [15]

- *Fabric*: The Fabric layer represents the diversity of logical and physical resources available to a grid network and to which shared access is negotiated by grid protocols. Resources may be logical entities, such as programs, files and data or physical entities such as processing cycles, disk storage, sensors or computer clusters **[4]**. The lowest layer of the IP stack is the Link layer, which represents the actual interface between the computer and the underlying physical network and dictates to the application on how it may use the inter-network to transmit an IP packet.
- *Connectivity*: The Connectivity layer encompasses the collection of communication and authentication protocols that permit the exchange of data between Fabric layer resources. Communication protocols will initially include those found in the current TCP/IP protocol stack, e.g. Internet protocols (IP, ICMP) and transport protocols (TCP, UDP). Authentication protocols extend communication protocols by providing cryptographic mechanisms for verifying the identity of users and resources. The corresponding layers of the IP stack are the Transport and

Inter-network layers, which handle the routing of messages across the network and provide for the end-to-end integrity of message packets.

- *Resource / Collective*: The Resource layer co-operates with the Fabric layer to handle the secure negotiation, initiation, monitoring, control, accounting and payment of individual local resources. The Collective layer is concerned with these same issues applied to interactions amongst a collection of distributed resources. The Resource and Collective layers have no suitably comparable counterparts in the IP stack.
- *Application*: The Application layer provides user access to services available at each level of the grid architecture via the protocols defined for each layer. Examples of services the application layer exposes include resource management, data access and resource discovery. The Application layer of the IP stack operates in the same capacity, providing the user with access to services specific to its architecture.

1.1.2. Grid Services and XML Web Services

The grid protocol architecture ensures the development of interoperable grid networks. Interoperability is a common characteristic of protocol architectures, due to their emphasis on defining the structure of interactions for network elements, rather than the mechanisms for those interactions. Protocols are utilized by services, which provide a specific capability within a network. Thus, services are predominantly defined by the protocol they implement and the behavior they allow. Working groups have defined a standard set of grid services that provide a core set of functionality necessary for establishing and managing VOs. The groups have addressed such issues as resource discovery, dynamic service creation and lifetime management **[17]**. Grid services are intended to provide a consistent interface for various grid behaviors, while abstracting the resource-specific details of how these behaviors are provided. In order to further ensure the interoperability of grids, these services are based upon the framework of XML Web Services.

XML Web Services are an emerging technology that has introduced a paradigm shift in computing, transferring the focus of software development from standalone desktop applications to Web-based and distributed applications. Web Services utilize open-standard technologies like Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL) and Universal Discovery, Description and Integration (UDDI) to expose application level logic over Internet-based protocols, such as HTTP and TCP/IP, as well as providing mechanisms to allow clients to discover them **[6]**, **[11]**. A brief examination of these four primary components of Web Services will provide a better understanding of how they permit the deployment of distributed applications over the Internet. Since we use Web Services to extend the grid's functionality to wireless environments, an understanding of how the components work is warranted here.

- *XML:* XML is a tag-based meta-language that allows developers to create self-describing documents that organize or markup data, like HTML does. However, XML allows developers to define the semantics of their tags, whereas the semantics and behavior of tags in HTML is pre-defined and largely immutable.

- *SOAP:* SOAP is an XML specification which serves as the communication protocol for Web Services. SOAP interprets data into a simple XML format, wrapping data into a few basic SOAP tags, for transmission over a standard Web communication protocol, typically Hyper-Text Transfer Protocol (HTTP). It should be noted that the SOAP specification does not restrict transmission protocol bindings for SOAP messages to HTTP, but it is the most common implementation due to its ubiquity. The core of the SOAP specification defines the standard format for a SOAP message. A SOAP message can vary in complexity from a simple text document to application-level data. For text documents, the SOAP wrapping and transmission process is relatively simple. However, the SOAP specification also provides recommendations for representing application data as XML, thereby permitting Remote Procedure Calls (RPC) over the World Wide Web (WWW). Application-oriented SOAP messages are capable of containing information about function definitions, including required parameters and return types, as well as the actual values of passed parameters and the results of a function call.
- *WSDL:* WSDL is another XML specification used to describe the interfaces for XML Web Services as well as the recommended protocols for communication with those services. Because a WSDL description is simply an XML document, it is human readable and editable, but WSDL documents are typically created by software tools or plug-ins dedicated to

automatically generating WSDL descriptions from an XML Web Service interface.

- *UDDI:* UDDI is a service discovery technology that provides users with a directory of available XML Web Services which can be searched according to various criteria. The UDDI directory is composed of individual directory entries that are basically XML files that describe an organization and the services it offers. UDDI directory entries are composed of three main categories: White Pages, Yellow Pages and Green Pages. White Pages provide information related to an organization offering a particular service, including the name of the organization, address and contact information. Yellow Pages provide a categorical classification of the organization offering a service, allowing entries to be grouped by industry, service type or geography. Green Pages provide a description of the interface of an XML Web Service in enough detail to permit clients to create applications utilizing the service.

With such an array of open-standard technologies related to Web Services already available, grid services will immediately benefit in two ways. First, mechanisms already exist to provide select features of grid networks such as the service description (via WSDL), the service registration and service discovery (via UDDI). Second, because grid services rely on the Web Service architecture, which, in turn, relies on the widely adopted Web/Internet architecture, Internet users will be able to adopt grid technologies with relative ease without facing interoperability or compatibility problems.

1.2. Mobile Computing

Continued improvements in integrated circuit design have radically redefined computer systems, enabling the miniaturization of computing devices and freeing them from the bulk and tether of traditional desktops, resulting in what is appropriately referred to as "mobile devices". Concurrently, the field of wireless networking has matured significantly from earlier attempts, like IrDA and the initial IEEE 802.11 specification. The confluence of these technologies establishes an unprecedented and practical foundation for the paradigm of mobile computing and wireless grids.

1.2.1. Wireless Communication

There have been multiple attempts to implement wireless standards. Despite the efforts of working groups dedicated to creating and implementing these standards, market forces have ultimately determined standards in wireless communication. Currently, the most prevalent wireless standards fall into two categories: short-range (IEEE 802.15) wireless communication protocols, and long-range (IEEE 802.11x) protocols.

An infrared-based standard (IrDA interface) for wireless communication was proposed by the Infrared Data Association (IrDA) for short-range wireless communication that supported transmission rates of 115 Kb/s, 0.576 Mbit/s, 1.152 Mbit/s, 4.0 Mbit/s and 16 Mbit/s **[31]**. Unfortunately, infrared is highly susceptible to interference and requires line-of-sight communication for devices to engage in data transmission. Another standard proposed for short-range wireless transmission utilized radio frequencies (RF) as a carrier signal for data transmission rather than infrared. Classified as the IEEE 802.15 standard or more popularly known as Bluetooth **[5]**, it is a low-power, short-range RF specification for wireless networking. Bluetooth supports transmission rates up to 16 Mb/s with a range of approximately 10 meters and doesn't suffer from the stringent line-of-sight requirement characteristic of IrDA devices.

RF transmissions are also utilized in protocols for long-range wireless communication **[9]**, **[27]**, the most popular of which is the 802.11x family of protocols for Wireless Local Area Networks (WLANs). The initial 802.11 specification was introduced in 1997 and declared operational frequency ranges in the unlicensed Industrial, Scientific, Medical (ISM) band (2.4 GHz) capable of supporting transmission rates up to 2 Mb/s **[20]**. The 802.11 protocol specified two methods of RF signal encoding, Direct Sequence Spread Spectrum (DSSS) and Frequency Hopping Spread Spectrum (FHSS), as well as a method for infrared signal encoding. Although these modulation schemes were incompatible with each other, the 802.11 specification did attempt to ensure inter-operability and standardization amongst devices which implemented similar WLAN physical layer schemes. However, flaws in the initial specification were exposed when the wireless market saw a proliferation of devices incapable of operating with products manufactured by other vendors **[9]**.

In 1999, the 802.11 specification was amended and ratified as 802.11b, which specified operation in the ISM band (2.4 GHz) and increased the maximum signal throughput from 2 Mb/s to 11 Mb/s. 802.11b specified a high-rate extension to the DSSS encoding scheme of the original 802.11 specification **[22]**. Subsequent 802.11x specifications were 802.11a and 802.11g. Both significantly improve wireless network throughput from 802.11b's 11 Mb/s to 54 Mb/s. The 802.11a standard uses a different frequency band than 802.11 and 802.11b, utilizing the Unlicensed National Information

Infrastructure (U-NII) band (5.15–5.25, 5.25–5.35 and 5.725–5.825 GHz). The higher frequency range of 802.11a enables significantly improved network throughput rates of up to 54 Mb/s **[21]**, **[27]**. The 802.11g protocol achieves a similar rate of throughput, but does so utilizing the same frequency band as 802.11b (ISM, 2.4 GHz) **[23]**; the increased bandwidth is achieved through the use of a different encoding scheme – Orthogonal Frequency Division Multiplexing (OFDM) instead of DSSS.

Although the short-range (Bluetooth) and long-range (802.11x) classes of wireless protocols appear to be in contention for the same wireless market, it has been noted that they serve clearly separable purposes and may, in fact, be used to supplement each other in wireless networks **[18]**, **[28].** Bluetooth provides short-range, low-power communication and is considered a Wireless Personal Area Network (WPAN) technology, intended to replace inter-device cabling and permit ad-hoc connectivity over short distances (~10 meters). The 802.11x protocols belong to a family of long-range wireless technologies usually intended to provide "last hop" connectivity to a nearby LAN or major network; WLANs can typically service a geographical area significantly larger than that covered by Bluetooth (~100 meters).

1.2.2. Inherent Constraints of Mobile Devices

Two characteristics define mobile devices and determine their unique benefits and constraints with regard to network performance – size and capability for wireless networking. In order to appreciate the potential role of mobile devices in evolving network systems, we must first examine the significant consequences of these characteristics.

Mobile devices are necessarily small to ensure portability. However, the reduced size of mobile devices directly influences a number of other design factors, including power supply, processor speed and storage space **[14]**. Batteries are the largest single source of weight in mobile devices. A tenuous balance exists between portability and power supply. A larger battery will provide a longer lifetime per charge, but at the cost of added weight and impeded mobility. A smaller battery will result in a more portable device, but reduce the amount of device usage per charge, requiring the recharging or changing of batteries more often, eventually impeding the overall portability of the device. Furthermore, processing power and storage space require battery power for operation and are subsequently constrained by the design of the power supply.

Another fundamental component of mobile devices reliant upon the power supply is a wireless transceiver. However, wireless networking introduces a host of unique constraints beyond power consumption. Despite the fact that there are a number of interfaces currently available for wireless communication, they all suffer from related problems. One of the most prominent drawbacks to wireless networking is the unreliable quality of connectivity due to interference **[19]**. Wireless interfaces, e.g. infrared or RF, may experience reflection, echoes or the introduction of noise due to environmental factors, which include atmospheric interference and physical obstructions. Additionally, mobile and wireless connections will suffer unplanned disconnections and significant bandwidth variability as they experience handoff across transmission boundaries, passing from the coverage of one wireless transceiver to another, potentially even over short distances. As a result, wireless network connections are frequently of lower quality in comparison to wired connections because they are much more susceptible to unexpected disconnections, poor signal strength and unreliable bandwidth.

1.3. Motivation

If circuit design continues to improve at the rate it has for the past decade, mobile and embedded devices will soon become substantial nodes in computer networks and emergent paradigms of computing such as grid networks. By design, however, they are subject to three unique constraints, which are atypical of traditional desktop-class machines. Mobile and embedded devices:

- typically operate on a finite and independent power supply,
- possess a significantly lower capacity for computational resources than traditional desktops, and
- are subject to unpredictable levels of network connectivity due to the prohibitive characteristics of wireless communication.

Although it would seem safe to assume that the constraints inherent in each of these respective issues will eventually be reduced or redressed through advances in chemical, manufacturing or design processes, this work has been predicated on the assumption that they will all remain fundamentally prohibitive factors in the design and manufacture of mobile devices for at least the near future. Therefore, it would seem to be prudent for developers in emerging fields of computing and computer networking to be aware of the influence of these constraints. Unfortunately, the majority of current grid applications target clusters of high-performance, resource-rich desktop systems for scientific or engineering purposes, making these systems unsuitable for testing across networks largely comprised of resource-constrained devices, e.g. mobile or embedded devices.

Because of these issues, we developed a grid simulator system to be light-weight and simple to accommodate wireless and mobile devices, yet sufficiently sophisticated to provide the basic features of a wired grid network. We primarily developed this system to determine the results of introducing a population of mobile devices into a grid network. We focused on observing the effects of mobility on the resource-sharing aspect of grid networks. For our purposes, mobility is defined as movement of the device or the resultant characteristic of unpredictable network connectivity (whether the cause is signal attenuation, signal interference, signal loss or power drain of a mobile device). We limited the definition of resource-sharing to computational resources (or processing power) only.

2. Grid Simulator System Infrastructure

The Grid Simulator System (GSS) refers to a collection of applications and services that are required to establish, administer and simulate a wireless grid network. The GSS is capable of managing a population of actual, user-controlled devices; creating and administering a population of autonomous virtual devices; or some combination of both. The GSS architecture is analogous to the 3-tier model of client/server computing. The GSS consists of the following components:

- Client Tier
  - Grid Simulator System - Module (GSS-M)
  - Grid Simulator System - Client Application (GSS-CA)

    - Grid Simulator System - Remote Control (GSS-RC)
- Application-Server Tier
    - Brokering Service
    - Keep-Alive Service
- Data-Server Tier
    - Active Agent Repository (AAR)
    - Task Allocation Table (TAT)

The infrastructure of the GSS represents a grid application, conforming to the description set forth by the grid protocol architecture drafted by the Globus Alliance and presented in §1.1.1. However, in an effort to reduce the complexity of development, select features of grid networks were excluded, particularly authentication and cryptographic protocols. Rather, the development effort was focused primarily on establishing a distributed network environment and implementing the mechanisms necessary to enable resource sharing.

The GSS-M provides the means to generate a virtual population of mobile devices to populate a grid network. Devices that wish to participate in the network established by the GSS are required to install a Grid Simulator System - Client Application (GSS-CA), a light-weight application which manages communication between client devices and the GSS-Broker. For the purpose of simulation and testing, virtual devices generated by a GSS-M possess an embedded version of the GSS-CA, the use of which is automated by behavior algorithms, discussed further in §2.2.1. The execution of these algorithms relies on three environment variables, which may be adjusted through the GSS-M: *initial population size*, *device mobility* and *task initiation frequency* (*TIF*). By adjusting the value of these variables and executing various configurations of simulated network scenarios, it is possible to calculate the relative impact of each of the environment variables on network efficiency, interpreted as the time taken to satisfy a resource request averaged over all such requests in a particular simulation session.

The GSS-RC is a supplemental application that provides the ability to co-ordinate multiple instances of the GSS-M, which may be distributed across multiple computers. This alleviates the burden of hosting an entire network population in the physical memory of one computer, allowing for testing with significantly larger population sizes.

The GSS-Broker is a server-based application responsible for administering resource requests in the grid. It is primarily composed of two system services that work in tandem to maintain a constantly updated record of the state of the grid network. The first is the Brokering Service, which provides logic for record-keeping and task distribution. The other is the Keep-Alive Service, which employs a TCP-based keep-alive protocol to confirm the presence of all devices registered with the GSS-Broker at regular intervals.

The remainder of this section will present a brief overview of the GSS infrastructure, a technical analysis of its components and a summary of the events that may occur in the course of a simulation session.

2.1. Grid Simulator System: Overview

The primary goal of this research and subsequently, of the GSS, is to investigate the effects of various environment variables on resource sharing in grid networks. In order to accomplish this, the following goals were established at the outset of this work:

- Create a distributed network environment to conduct testing.

- Create a method or process for emulating resource-sharing.
- Determine metrics for gauging network performance with regard to resource-sharing.

The details of the GSS network environment will be covered in depth throughout the remainder of the following section.

In the GSS, resource-requests and resource-sharing involve the use of computational tasks considered too intensive for an individual device's processing power. Within the context of the GSS, a computational task is defined as any task that may be executed in parallel on multiple devices. When requesting resources from the network, a device submits a computational task to the GSS-Broker for execution on other devices, and specifies the number of devices it wishes to utilize. Such devices are referred to as Initiators; any devices in the network that are not classified as Initiators are referred to as Subordinates. It should be noted that all devices in the network are equally eligible to become an Initiator. The GSS-Broker selects devices from the available network population to run a pending task based on the following factors, which dictate a particular device's eligibility for task assignment:

- *Activity state*. Devices that have initiated a grid task or have already been assigned a sub-task are ineligible for task assignments.
- *Resource ranking*. A simple algorithm is employed by the GSS-Broker to rank the eligibility of available devices for task assignment: Rank = (Physical Memory) * (Processor Speed). Larger values of the rank indicate increased eligibility for task assignment.

In this way, devices may share or request processing-cycles across the network, hence emulating the resource-sharing characteristic of grid networks.

The concept of the computational task also presents a convenient metric for determining the relative efficiency of the network environment. It was determined that the time taken to satisfy a resource request, represented by same computational task, run with the same parameters across the same number of devices, could provide a baseline measure for network efficiency. By averaging the time taken to satisfy a resource request over all the resource requests in a given simulation session, one can determine the relative effect of each GSS environment variable on network performance.

For purposes of testing, an application utilizing the Iterative-Deepening A* (IDA*) algorithm was chosen as the computational task. The IDA* algorithm is a depth-first search algorithm that can determine the shortest path between specified nodes in a graph. The IDA* search method is a modified depth-first search algorithm that utilized a cost limit, rather than depth limit, to restrict the search space for each iteration **[25]**. The IDA* algorithm utilized by the GSS is modified for implementation in a distributed environment, with special emphasis on the following characteristics:

- It requires minimal co-ordination for execution.
- It requires no inter-process communication for execution.
- It is fault-tolerant, recovering easily from lost processes.

In preparation for distribution, it is incumbent upon the submitting device (or the Initiator) to provide a number of execution parameters to accompany the distribution of the application, including a start node, a goal node and a list of partial-solution paths generated by the submitting device. The GSS-Broker assigns the grid task to an appropriate number of devices, i.e. one partial-solution path per device for all partial-solution paths, and distributes the executable with parameters accordingly. Each assigned device completes its sub-task, generating a sub-optimal solution, which is returned to the GSS-Broker. Finally, it is the responsibility of the submitting device to retrieve and aggregate the collection of sub-optimal solutions from the GSS-Broker.

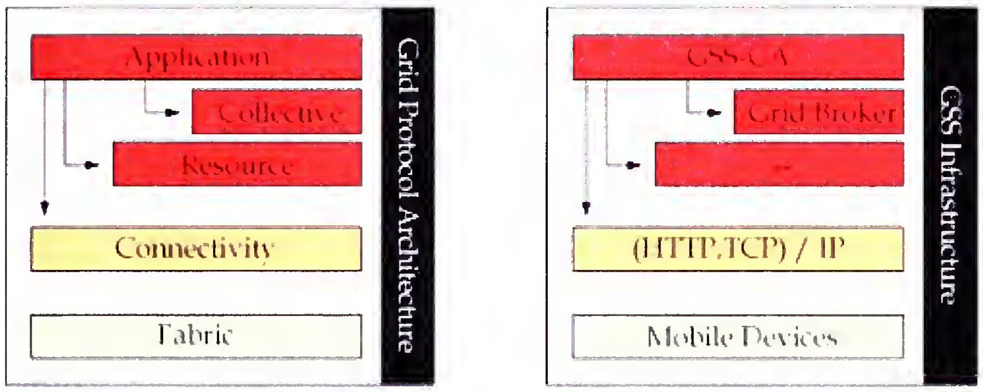


Figure 2. Grid Protocol Architecture and Grid Simulator System Architecture

To provide a clearer picture of the grid network represented by the GSS, a brief analysis is presented in a layer-based comparison to the Grid Protocol Architecture, illustrating its nature as a grid application.

- *Fabric:* The Fabric layer represents the collection of resources available to a grid network. While standard grids are designed to allow a variety of logical and physical resources to be shared, the GSS is primarily designed to support the sharing of computational resources, i.e. processing cycles.
- *Connectivity:* The Connectivity layer specifies the protocols required to permit communication between Fabric layer resources. The Grid Protocol Architecture currently allows grid implementations to utilize the TCP/IP protocol stack and anticipates support for less popular protocols in the

future. The GSS simply utilizes HTTP/TCP over IP for network communication via XML Web Services and direct TCP connections.

- *Resource/Collective:* The Resource and Collective layers specify the protocols involved in administrating the Fabric layer resources; the Resource layer pertains to individual resources, while the Collective layer pertains to interactions among a collection of resources. The Resource/Collective layers are implemented in the GSS as the GSS-Broker, which administers and provides the mechanisms for resource-sharing.
- *Application:* The Application layer provides access to user services that utilize the protocols defined in each of the lower levels. A GSS-M provides access to the underlying grid services available to client devices in the GSS.

The remainder of this section focuses on presenting the technical details of the components that comprise the GSS, with particular attention to their architectures and interactions.

2.2. Grid Simulator System: Component Architecture

The GSS is a collection of applications and services, which are capable of establishing a resource-sharing network environment and generating a virtual population of mobile devices to utilize that network. The GSS environment is defined by the interaction of a GSS-M and a GSS-Broker. Simulation sessions of greater complexity may be executed, involving multiple GSS-Ms and a GSS-RC to coordinate the parallel execution of the distributed simulation modules. The following sections discuss the architecture of these components in greater depth, enumerating the range of their functions and the extent of their interactions.

2.2.1. Grid Simulator System: Module

A GSS-M is an application that may be used to customize, execute and monitor simulation sessions. A GSS-M is primarily intended to allow users to customize and generate a population of autonomous virtual devices to populate the grid network. In this sense, the GSS-M represents the client-tier of the GSS for simulation sessions. It may be run in a stand-alone mode or a distributed mode, in which multiple modules are coordinated to run in parallel, distributed across a network.

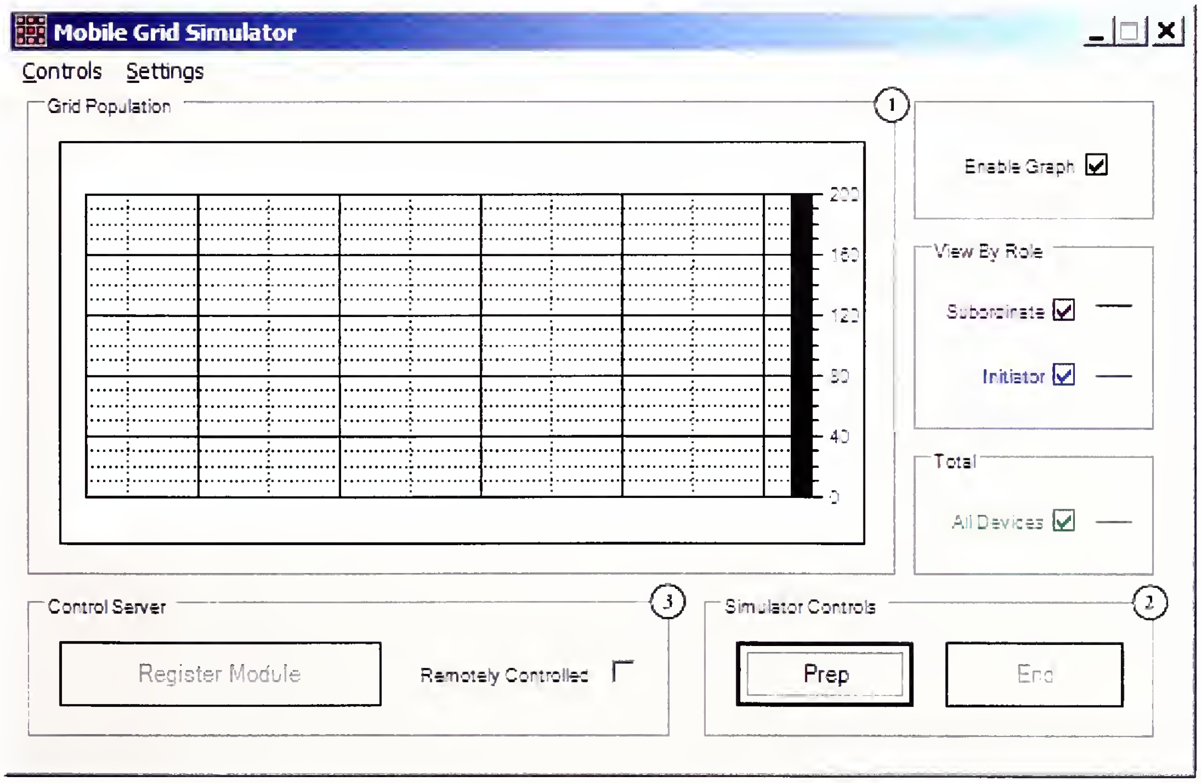


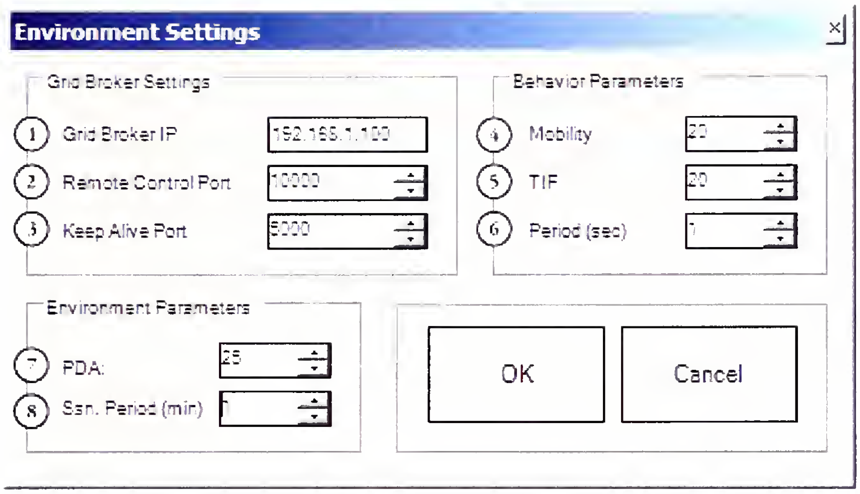
Figure 3. Grid Simulator System - Module

Figure 3 above reveals the main interface for a GSS-M, whose primary features are described below:

1. *Population Monitor*, a graph control that presents a real-time display of population behavior during simulation sessions.
2. *Simulator Controls*, which provide manual control over starting/stopping a simulation session. These controls are intended for use in stand-alone mode. A session cycle consists of three phases:
   - *Prep*. The initial device population is generated and registered with the GSS-Broker.
   - *Start*. Device behavior algorithms are invoked and the session begins.
   - *End*. All devices are un-registered from the GSS-Broker. The simulation session terminates.
3. *Remote Control Panel*, which allows a user to register the current instance of a GSS-M with the GSS-RC. This control is intended for use in distributed simulation sessions. Use of this feature is documented in §2.2.2.

In addition to these basic features, the GSS-M allows a user to modify a collection of system settings which serve as the operational parameters of the simulator environment. These system settings are organized into the following four categories: Environment, Task, Log and Graph Settings.

The Environment Settings panel (Figure 4) allows a user to modify settings that are required to establish a GSS Grid Network or settings that are related to executing a simulation session. The Environment Settings are categorized as GSS-Broker Settings, Environment Parameters and Behavior Parameters.



Environment Settings

Grid Broker Settings

① Grid Broker IP: 192.168.1.100
② Remote Control Port: 10000
③ Keep Alive Port: 5000

Behavior Parameters

④ Mobility: 20
⑤ TIF: 20
⑥ Period (sec): 1

Environment Parameters

⑦ PDA: 25
⑧ Ssn. Period (min): 1

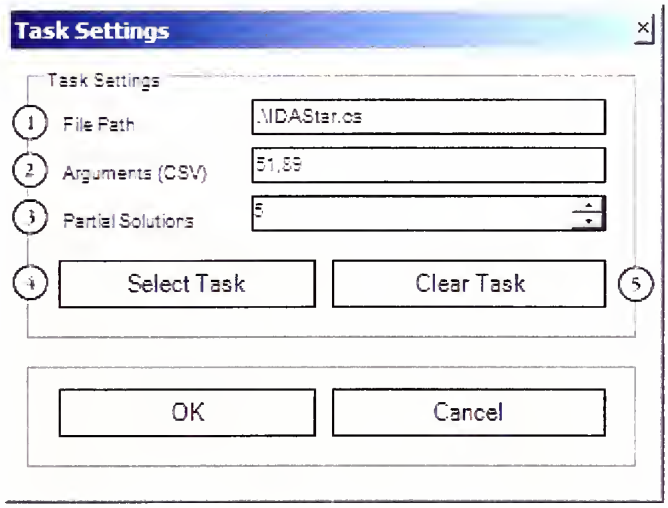
Figure 4. Grid Simulator System – Module (Environment Settings)

GSS-Broker Settings specify the communication parameters required to establish a connection between client devices and the GSS-Broker, including the IP address of the GSS-Broker, represented by ① in the figure, the communication port of the GSS-Broker, represented by ③, and the communication port of the GSS-RC (②). As a brief note, there is no need to provide the IP address of the GSS-RC, because it is intended to run on the server which hosts the GSS-Broker.

Environment Parameters specify the initial values for environment variables prior to the start of a simulation session. These variables include the initial population size, represented by ⑦ in Figure 4, and the length of a session period (⑧).

Behavior Parameters specify those values required by the simulator's algorithms to generate the behavior of virtual devices. These values indicate the level of mobility exhibited by the overall device population, represented by devices added/dropped (④); the frequency of resource requests generated by the overall device population, represented by the number of computational tasks initiated (⑤); and the frequency with which these events are calculated during a simulator session (⑥). The same behavior algorithm is utilized to calculate the number of virtual devices that will be affected by the *device mobility* and *Task Initiation Frequency* (*TIF*) events in each period specified in the Environment Settings panel. The values are calculated as a randomly generated Poisson distribution based on the percentage of the *initial population size* **[13]**.

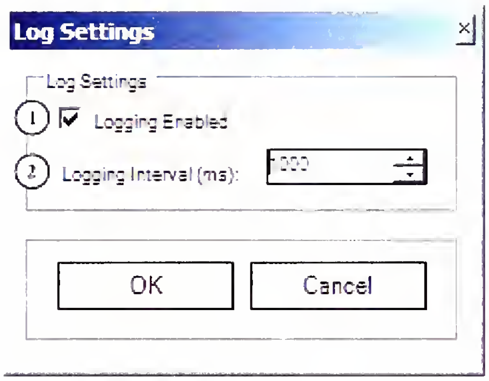
The Task Settings panel (Figure 5) provides controls to specify details related to the computational task to be used in the course of a simulator session. The GSS is only configured to process the submission of console applications as computational tasks, in the interest of maintaining the transparency of resource-sharing. Users must specify the file path to the task source code, represented by ① in Figure 5,, as well as the number of sub-tasks to generate (represented by ③). Optionally, users may specify a comma-separated argument string with which to initialize all sub-tasks (②).



File Path .\IDAStar.cs
Arguments (CSV) 51,59
Partial Solutions 5

Figure 5. Grid Simulator System – Module (Task Settings)

The Log Settings panel (Figure 6) provides access to the logging feature of a GSS-M, allowing a user to enable/disable logging (see ① in the figure below). If logging is enabled, a user may also specify the interval at which the state of the simulator environment is logged (②).


Log Settings

Log Settings

1 ☑ Logging Enabled

2 Logging Interval (ms): 1000

OK Cancel

Figure 6. Grid Simulator System – Module (Log Settings)

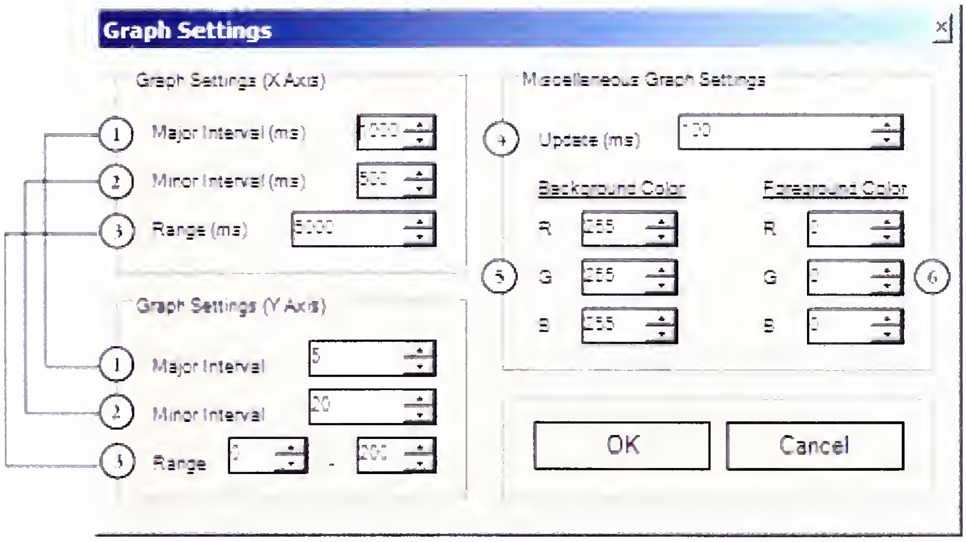
Three log files are produced at the end of every simulation session in which logging has been enabled – a Session Log, Task Results Log and Task Events Log. The Session Log is used to record the general state of the simulator environment throughout the course of a simulation session. The following list indicates variables that are tracked by the Session Log. *Incremental* indicates the change in value from the previous logging interval. *Total* indicates the change in value from the beginning of the simulation session.

- Total Subordinates
- Total Active Subordinates
- Total Initiators
- Devices Added (*Incremental*/*Total*)
- Devices Dropped (*Incremental*/*Total*)
- Active Subordinates Dropped (*Incremental*/*Total*)
- Initiators Dropped (*Incremental*/*Total*)
- Tasks Initiated (*Incremental*/*Total*)
- Tasks Completed (*Incremental*/*Total*)

The Task Results Log documents the completion state and execution time for grid tasks. Tasks may assume one of three completion states by the end of a simulation session:

- *Completed.* All sub-tasks have been executed and the results have been collected by the Initiator.
- *Aborted, Device Dropped.* An Initiator was unexpectedly dropped from the network and the associated task was aborted.
- *Aborted, Simulation Terminated.* The simulation session terminated before the associated task could be completed.

At the end of a simulator session, the execution times for all grid tasks with a documented completion state of COMPLETED are averaged together. This average task execution time is then used to calculate the efficiency of the network, based on the configuration of the environment variables. This process will be discussed further in §3.1. Finally, the Task Events Log documents the life-cycle of all grid tasks initiated during a simulation session. The events comprising a grid task's life-cycle are presented in §2.3.



Graph Settings (X Axis)

Major Interval (ms): 1000

Minor Interval (ms): 500

Range (ms): 5000

Graph Settings (Y Axis)

Major Interval: 5

Minor Interval: 20

Range: 0 - 200

Miscellaneous Graph Settings

Update (ms): 100

	Background Color		Foreground Color
R	255	R	0
G	255	G	0
B	255	B	0

Figure 7. Grid Simulator System – Module (Graph Settings)

The Graph Settings panel (Figure 7) allows a user to customize the graph control located in the Grid Simulator Application's main window. Changes applied to the graph control do not affect the execution or result of any simulation sessions. Users may modify the following features of the graph control:

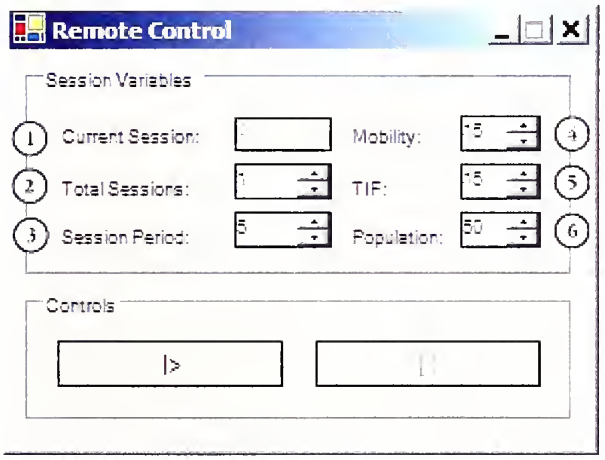
1. Display range, X/Y-Axis
2. Major tick interval, X/Y-Axis
3. Minor tick interval, X/Y-Axis
4. Graph control refresh rate
5. Graph control background color
6. Graph control foreground color

The X-Axis represents an interval of time specified in the X-Axis Range field. The Y-Axis represents the number of devices in the network population. For both axes, the major (solid) lines demarcate the value in the Major Interval field and minor (dotted) lines demarcate the value in the Minor Interval field. Supplemental graph controls can be found on the Grid Simulator Application's main window, allowing a user to track any combination of Subordinates, Initiators and total devices during a simulation session.

2.2.2. Grid Simulator System: Remote Control

The GSS-RC (Figure 8) is a supplemental application, which may be used to automate and coordinate the execution of multiple GSS-Ms that are distributed across a network. In order to do so, all participating modules must register with the GSS-RC prior to a simulation session, which is accomplished by using the "Register Module" control on the main window of each GSS-M (Figure 3). When a module registers with the GSS-RC, it submits its IP address, which allows the GSS-RC to establish a TCP connection with each module. These connections are used to synchronize the execution of all registered simulator modules.

The GSS-RC provides access to the fundamental simulator environment variables, Session Period, represented by ③ in Figure 8, *device mobility*, represented by ④, *TIF* (⑤) and *initial population size* (⑥), whose settings take precedence over those specified in any GSS-M. Additionally, the GSS-RC possesses the capacity to automate the execution of multiple consecutive simulator sessions, based on the value provided in the Total Session field, represented by ② in Figure 8. The Current Sessions field (①) displays the number of simulation sessions in the Session Queue that have been completed.


Remote Control

Session Variables

1	Current Session:		Mobility:	15	4
2	Total Sessions:	1	TIF:	15	5
3	Session Period:	5	Population:	50	6

Figure 8. Grid Simulator System – Remote Control

A session cycle for a distributed session is identical to the three-phase cycle of a stand-alone simulation session.

- *Prep*. The GSS-RC generates a "Prep Packet," which is transmitted to each registered GSS-M. Each module proceeds to perform the actions described in the stand-alone simulation prep phase. Afterwards, each module sends a READY signal to the GSS-RC.

- *Start.* When every simulation module has returned a READY signal, the GSS-RC responds with a START signal and the simulation session begins.
- *End.* When every module has completed the actions described in the stand-alone end phase an END signal is sent by each module back to the GSS-RC. If there are any sessions left to execute in the Session Queue, the distributed session cycle begins again.

As a final note, at the end of each distributed-mode simulation session in which logging is enabled, all participating simulator modules submit their individual log files to the GSS-Broker. When all modules have submitted an END signal to the GSS-RC, it retrieves the aggregated log data from the GSS-Broker and writes the logs to disk. Thus, at the end of stand-alone mode simulation sessions, logs may be found on the computer hosting the GSS-M in use. However, at the end of distributed mode simulation sessions, logs will be found on the computer hosting the GSS-RC.

2.2.3. Grid Simulator System: Broker

The GSS-Broker is a server-based administrative application whose primary purpose is to administer and coordinate resource requests in the GSS network environment. It represents the applications-server tier of the GSS and is composed of two system services, the Brokering Service and the Keep-Alive Service, which co-operate to maintain the stability and efficient operation of the simulated wireless network environment.

2.2.3.1. Brokering Service

The Brokering Service is a suite of XML Web Services that provide the business logic of the GSS-Broker. In order to make informed and effective decisions in response to resource-requests, the Brokering Service requires accurate and timely knowledge concerning the state of the grid, particularly *device state*, which describes the activity level and current configuration of each client device in the network population and *task state*, which refers to the measure of progress in the assignment and execution of initiated grid tasks. The process of tracking the relevant aspects of grid state relies on the employment of two SQL-Server database tables, which represent the data-server tier of the GSS and allow the Brokering Service to store and retrieve collected state information. These tables are the Active Agent Repository (AAR) and Task Allocation Table (TAT) and they store device and task related data respectively.

Essentially, the Brokering Service is the interface that client devices use to access the underlying grid services of the GSS, with the GSS-CA providing the primary means for communication between client devices and the GSS-Broker. In an effort to minimize the utilization of network resources, a communication channel is maintained between a GSS-CA and the GSS-Broker only as long as data is actively being requested or transmitted. These channels are built-up and torn-down automatically. The goal of the GSS-CA is to maintain maximum transparency for communication with the GSS-Broker; user intervention is required only for the initial request to join the grid, to request resources from the grid or to notify the Brokering Service of departure from the grid.

In addition to the communication channels that may be established between a GSS-CA and the Brokering Service, there is a direct-link, server-side TCP/IP communication channel that exists between the Brokering Service and Keep-Alive Service. Similar to the bandwidth-conserving behavior of GSS-CA initiated connections, the server-side communication channel is sustained only for the period of active data transmission between the Brokering Service and the Keep-Alive Service. This custom communication channel was implemented because there are no inherent mechanisms for communication between the Windows Service and Web Service technologies. There are a collection of well-defined message packets that allow for the Brokering Service and Keep-Alive Service to efficiently transmit information in response to events that may occur within the grid. A complete review of the structure of these message packets is presented in Appendix A, while a detailed presentation of the simulator network events may be found in §2.3.

2.2.3.2. Keep-Alive Service

The Keep-Alive Service is a Windows Service application that is intended to supplement the Brokering Service web application in administering the grid network. The Keep Alive Service implements a keep-alive protocol, which challenges registered devices in the grid at regular intervals for the purpose of confirming their maintained presence in the network. This is necessary due to the stateless nature of XML Web Services and the Brokering Service. After sub-tasks have been assigned by the Brokering Service, there are no inherent mechanisms in the GSS-Broker for confirming the sustained presence of active devices, i.e. Initiators and Subordinates.

Devices that do not explicitly inform the GSS-Broker of their departure, due to circumstances such as signal loss or power drain, will “silently” drop out of the Grid, causing related tasks assigned to or initiated by the device to "hang." All devices that are active in the grid, either by initiating a task or being assigned a sub-task, are added to a keep-alive collection maintained by the Keep-Alive Service. A keep-alive communication channel is established between the device and the GSS-Broker. This channel permits the GSS-Broker to immediately respond to changes in the network population. By restricting the implementation of the keep-alive protocol to active devices, the demand on the GSS-Broker's resources is kept to a minimum.

The Keep-Alive Service's unique ability to initiate communication with client devices imposes responsibilities beyond the keep-alive protocol, including:

- *Extended task-assignment functions.* The Brokering Service is only capable of making a one-time, best-effort attempt to assign initiated tasks. In the event that all sub-tasks are not assigned in this initial attempt, a message packet is created by the Brokering Service and transmitted to the Keep-Alive Service, specifying the number of sub-tasks left unassigned. Additionally, if an active Subordinate drops out of the network, its sub-task must be re-assigned by the Keep-Alive Service to ensure completion of the associated grid task.
- *Task cleanup functions.* In the event of an aborted task, whether requested by an Initiator or due to an Initiator "silently" dropping out of the grid, the Keep-Alive Service is responsible for alerting the appropriate active Subordinates to cease operation and discard their assigned sub-tasks.
- *General notification and messaging.* Any occasion requiring the GSS-Broker to communicate with a segment of the network population utilizes

the Keep-Alive Service and its ability to establish a communication channel with client devices. For example, when all active Subordinates associated with a grid task have generated and returned their results, the Brokering Service caches the results and generates a message packet to inform the Keep-Alive Service that a grid task has been completed. However, it is the responsibility of the Keep-Alive Service to inform the appropriate Initiator that results are available on the GSS-Broker.

The combined application of the Brokering Service and Keep-Alive Service creates a system capable of immediately and intelligently responding to the highly volatile topology characteristic of networks comprising mobile devices.

2.3. Simulator Events

The characteristically random behavior of virtual devices in the GSS attempts to emulate the activity of real-world networks and is one of the crucial factors in determining the quality and accuracy of data generated by simulation sessions. The behavior exhibited by virtual devices, although essentially random, occurs within a fixed framework of well-defined events. Simulator sessions take form from the random activity of devices occurring within this framework of system events and, for this reason, the remainder of this section presents a descriptive analysis of the events that may occur throughout the course of a simulator session. Events are presented categorically, according to the GSS component they are most closely associated with: client devices, the Brokering Service or the Keep-Alive Service.

2.3.1. Client Devices

The devices that constitute the network population of the GSS are the only elements of the system that are capable of truly initiating events. Events associated with the Brokering and Keep-Alive Services are ultimately a response to events invoked by client devices. There are three categories of events available to client devices, two based upon their assumed role in the network as Subordinates or Initiators and a third available to either.

The category of general client device events may be invoked by either Subordinates or Initiators. However, the particular processes involved for these events may differ based upon the role of the invoking device.

Subordinate devices, as the name suggests, take on a predominately passive role in relation to other elements in the network. Thus, events invoked by Subordinate devices primarily affect the state of the device in question and do not impose upon the resources of other devices in the network or on the network itself.

Initiators are devices that occupy a sustained, active role in the network, which is due to the utilization of network resources by its associated task; the role of Initiator persists for the lifetime of the device's associated task. Events invoked by an Initiator primarily affect the state of its task, rather than of the device itself.

2.3.1.1. Device Registration (General)

The participation of any device in the GSS network begins with registration, which is a role-independent event. A device submits a request to join the network by generating a message packet that contains the variety and amount of resources the submitting device will make available to the network. The request packet is transmitted to the Brokering Service of the GSS-Broker, which commits this information to the AAR. A unique device ID is generated and returned to the device that requested membership in the network. This indicates a successful registration and thus concludes the registration process.

2.3.1.2. Device Un-Registration (General)

Unlike registration, un-registration is a role-dependent event that signals the conclusion of a departing device's participation in the network. Upon receiving a departure notification from a client device, the Brokering Service checks the AAR to determine the role of the device. If the departing device is a Subordinate, the Brokering Service checks for an assigned sub-task which, if found, is re-assigned by the Keep-Alive Service in a process outlined in §2.3.3.2. Following the check for a sub-task and any subsequent action, the device entry is simply removed from the AAR.

If the departing device is an Initiator, the Brokering Service relays the ID of the departing device to the Keep-Alive Service. The Keep-Alive Service identifies any Subordinates assigned a task associated with the departing Initiator and transmits a small ABORT_PARTIAL_TASK packet to each device. While the Keep-Alive Service instructs the appropriate active Subordinates to abort their assigned sub-tasks, the Brokering Service removes the entry associated with the departing Initiator from the AAR. The task entries are then removed from the TAT.

2.3.1.3. Initiate Task (Subordinate)

When a client device chooses to request resources from the network, it submits a computational task, a specification for the amount of resources requested and an optional argument string (for initializing sub-tasks) to the GSS-Broker. The Brokering Service receives the task as source code, which is compiled into an executable and stored locally. A new entry in the TAT is created for the task while the Brokering Service proceeds to select the most eligible devices for task assignment from the active Subordinate population. Due to the inherent state-less nature of Web Services, the initial task assignment by the Brokering Service is a one-time, best-effort attempt.

If there remain any tasks to be assigned, the responsibility is transferred to the Keep-Alive Service, which possesses the capability and mechanisms to complete the assignment of tasks or re-assignment of sub-tasks. For each sub-task that has been assigned to a Subordinate, whether by the Brokering Service or by the Keep-Alive Service, the Keep-Alive Service assembles a small alert packet (RETRIEVE_PARTIAL_TASK) that is sent to the appropriate device. This purpose of this packet is to inform a Subordinate that it has a pending sub-task to be retrieved from the Brokering Service.

2.3.1.4. Retrieve Partial Task (Subordinate)

Upon receipt of a RETRIEVE_PARTIAL_TASK alert packet from the Keep-Alive Service, a Subordinate will relay its network ID to the Brokering Service and request the pending sub-task. Using the device ID, the Brokering Service determines the ID of the associated task and assembles a Task object from the information contained in the TAT and returns the task ID to the device. The information delivered to a Subordinate device via the Task object consists of the task ID, a string indicating the local directory to which results will be written, an array of bytes that comprise the task executable and an (optional) argument string.

2.3.1.5. Return Partial Task (Subordinate)

Computational tasks should include provisions for writing task results to disk as an array of bytes. Because the computational devices being discussed in this work typically possess significantly greater amounts of disposable disk memory versus physical memory, it is recommended that, in the event of large result sets, task results be written to the local disk by a Subordinate before contacting the Brokering Service. It is further recommended that all results be stored generically as an array of bytes on the Subordinate device, before transmission to the Brokering Service for storage and retrieval by the Initiator.

The Brokering Service should not be required to be aware of the nature of results; it is merely responsible for receiving partial results from Subordinates and storing them for retrieval by Initiators. We implemented the aforementioned recommendations in order to circumvent the complexity and processing overhead of storing type-specific results in the TAT.

Thus, when an active Subordinate completes execution of a sub-task in the GSS, it writes the results as an array of bytes to the local disk. The device then contacts the Brokering Service, and submits the results and the ID of the assigned task. The Brokering Service uses the task ID to create an entry in the TAT to store the partial result being returned. After all sub-tasks have completed execution and all partial results have been submitted to the GSS-Broker, the Keep-Alive Service assembles a RETRIEVE_PARTIAL_RESULTS packet to alert the appropriate Initiator that it has pending results for collection on the GSS-Broker.

2.3.1.6. Retrieve Partial Result (Initiator)

Upon receipt of a RETRIEVE_PARTIAL_RESULTS packet from the Keep-Alive Service, an Initiator is expected to contact the Brokering Service, providing its assigned device ID in an attempt to request delivery of pending sub-task results. Using the submitted device ID, the Brokering Service aggregates the results from the TAT into a ResultCollection object, which is simply an array of byte arrays. This ResultCollection is returned to the inquiring Initiator, signifying the completion of the computational task and the satisfactory fulfillment of a resource request.

2.3.1.7. Abort Task (Initiator)

There are two actions that may result in the abortion of an initiated task: (1) an Initiator may voluntarily choose to prematurely terminate the execution of its computational task, or (2) an Initiator may unexpectedly drop out of the network due to unpredictable factors such as power loss or signal interference.

Should an Initiator elect to abort a task, it alerts the Brokering Service by calling the appropriate method and submitting its unique device ID. The Brokering Service proceeds to remove relevant entries from the TAT and informs the Keep-Alive Service that active Subordinates working on associated sub-tasks should abort the tasks and discard any partial results obtained. The Keep-Alive Service distributes this message to the appropriate Subordinates in the form of an ABORT_PARTIAL_TASK packet that is delivered by "piggy-backing" on the keep-alive connections maintained by the Keep-Alive Service with all Initiators and active Subordinates.

If an Initiator unexpectedly drops out of the grid network, the Keep-Alive Service will immediately detect the connection failure via the keep-alive protocol. It then transmits an ABORT_PARTIAL_TASK packet to all Subordinates occupied with the absent Initiator's task and concurrently informs the Brokering Service of the Initiator's departure. This allows the Brokering Service to update the AAR and TAT accordingly, reflecting the departure of the Initiator and the availability of the previously occupied Subordinates.

2.3.2. Brokering Service

The Brokering Service is a component of the GSS-Broker that has a reactionary role in the GSS. Action is taken by the Brokering Service in response to those events initiated by the device population as described previously. While the events of the Brokering Service arise from methods that are primarily intended to provide network clients with access to the underlying services of the grid network, they are more closely associated with the secondary purpose of those methods, which allow the Brokering Service to manage and maintain an accurate record of the overall state of the grid.

2.3.2.1. Register Device

The initial point of contact for devices that wish to join the Grid is the Brokering Service. By means of the GSS-CA, devices submit a request to join the network by presenting the Brokering Service with a list of their salient characteristics, encapsulated in a Device object. A Device object contains the following properties of a client device:

- Device Type
- Processor Speed
- Physical Memory
- Disk Space
- Power Level
- IP Address
- Port

The Brokering Service commits this information to the AAR and, if a record entry is successfully created, generates a unique device ID that is returned to the device. The ID issued to the device is used to distinguish the device in any future network transactions that it may participate in; its issuance concludes the registration process.

2.3.2.2. Un-Register Device

A device may be un-registered from the grid voluntarily or involuntarily. A voluntary un-registration occurs when a device informs the Brokering Service. This allows the Brokering Service to immediately update the record of the network state by removing the device's entry from the AAR. However, due to unpredictable factors such as power loss or signal interference, devices are not always capable of alerting the GSS-

Broker of their departure. In cases of involuntary departure, the system will remain unaware of a device's departure until the keep-alive protocol detects the absence.

If the departing device is an Initiator, it will be engaged in the keep-alive protocol. So, when it fails to respond, the Keep-Alive Service will take appropriate measures to recover network resources, i.e. active Subordinates, by instructing the appropriate devices to abort ongoing processing and discard any results related to their assigned tasks. If the departing device is an inactive Subordinate, it will "silently" drop out of the network until the Brokering Service attempts to assign a task to it. When the Keep-Alive Service attempts to establish a keep-alive connection with the device and gets no response, the Brokering Service will be alerted. If the departing device is an active Subordinate, the keep-alive protocol will detect the involuntary departure. The Keep-Alive Service will re-assign its sub-task, and the Brokering Service will be notified of the appropriate changes in resource allocation.

2.3.2.3. Initiate Task

When a client device wishes to request resources from the grid, it does so by submitting a computational task to the GSS-Broker for distribution and execution. The task initiation method of the Brokering Service requires that a device provide the following information:

- The task source code, from which the Brokering Service will compile an executable for distribution.
- An argument list, to be used in the initialization of sub-tasks.

- A specification of the number of sub-tasks for the Brokering Service to generate.

This information is encapsulated in a TaskInfo object, submitted to the Brokering Service via a GSS-CA. An entry is created in the TAT for the new task and it is assigned a unique task ID that is identical to the device ID of the associated Initiator. Following this, an attempt is made by the Brokering Service to assign each sub-task to an inactive Subordinate. However, because XML Web Services are a stateless technology, this is a one-time attempt.

If all sub-tasks cannot be assigned, the responsibility is passed onto the Keep-Alive Service, which contains a mechanism capable of completing the assignment of sub-tasks. This mechanism is referred to as the CompletionStateManager. When a sub-task is assigned to an inactive Subordinate, the task ID is recorded in a field in the Subordinate's AAR record. Regardless of which component of the GSS-Broker assigns the sub-tasks, the Keep-Alive Service concludes the process of initiation by assembling small RETRIEVE_PARTIAL_TASK packets, which are delivered to the appropriate Subordinates, informing them of sub-tasks on the Brokering Service.

2.3.2.4. Abort Task

Task abortion may occur voluntarily or involuntarily, like device un-registration. If an Initiator concludes that a particular task is no longer relevant or is satisfied with the number of (intermediate) results generated, then the decision may be made to voluntarily abort the task prematurely. In this case, the appropriate method is invoked on the Brokering Service via the Initiator's GSS-CA and the abortion process begins. The local copy of the task located on the server hosting the GSS-Broker, is deleted and the appropriate task entry is removed from the TAT. The Brokering Service then informs the Keep-Alive Service of the task abortion, which assembles ABORT_PARTIAL_TASK packets and transmits them to active Subordinates executing associated sub-tasks.

However, if an Initiator unexpectedly drops out of the network, the Keep-Alive Service detects the departure and assembles ABORT_PARTIAL_TASK packets in response to this event, informing the appropriate Subordinates to abort their assigned sub-tasks and discard any results. It also informs the Brokering Service of the Initiator's departure. This allows the Brokering Service to update the AAR and TAT to reflect the changes in the state of the grid by removing the Initiator and its associated task's entries from their respective tables.

2.3.3. Keep-Alive Service

Like the Brokering Service, the Keep-Alive Service is a component of the GSS-Broker that is incapable of initiating events in the network. The Keep-Alive Service is primarily responsible for executing the keep-alive protocol during resource request sessions and for facilitating messaging across the network. It is the only component of the GSS-Broker that is capable of initiating a connection with client devices. The Brokering Service is merely capable of communicating with client devices as a response to method invocation.

2.3.3.1. Re-Assign Sub-Task

The assumption is made that all devices in the network population are susceptible to the possibility of unexpectedly dropping out of the grid. Recovery mechanisms were briefly discussed in §2.3.2.2, summarizing the events following the departure of a Subordinate or Initiator. Here we will discuss the recovery mechanism for the unexpected departure of active Subordinates in greater detail, with particular attention to the process of re-assigning its associated grid sub-task.

If an inactive Subordinate suddenly drops its connection to the network, the process is relatively straightforward. No action is taken until the Brokering Service attempts to assign a sub-task to the missing device. When the Keep-Alive Service attempts to establish a keep-alive connection with the absent device and receives no response, it notifies the Brokering Service of the failure, which then removes the device's entry from the AAR. The Keep-Alive Service then re-assigns that particular sub-task.

However, if an active Subordinate suddenly drops its connection to the network, the device's absence will immediately be detected because of the established keep-alive connection. The Keep-Alive Service notifies the Brokering Service to remove the device's entry from the AAR. In the process of removing the device's entry, the Brokering Service determines the ID of the sub-task assigned and makes an attempt to re-assign it to another inactive Subordinate. Like the initial task assignment process, the Brokering Service makes a one-time, best-effort attempt to re-assign the sub-task. If no eligible devices are available for assignment, the responsibility is handed over to the Keep-Alive Service, which employs the CompletionStateManager to finish the process.

3. Experimental Process And Analysis Of Results

The central purpose of the GSS is to provide the means for researching the direct effects, if any, of mobile devices on resource-sharing in wireless grid environments. After developing the framework for a simulator to demonstrate the effects of mobility on resource-sharing grids, we implemented a series of experiments designed to test the simulator in different scenarios. The remainder of this section describes the setup and configuration of these experiments and provides analysis and inferential observations based on the simulation results.

3.1. Process Overview

The GSS was designed to serve as the vehicle for experiments by providing a non-deterministic environment to conduct testing and the mechanisms for selectively controlling aspects of interest in the network environment. The GSS allowed us to test the impact of mobility on resource-sharing under various conditions, thus approximating the behavior of real-world mobile devices participating in a wireless grid. The experimental variables were introduced in §2.2.1 and are summarized here. *Device mobility* is the only customizable device property and applies to the device population as a whole. It is used in calculating the number of arriving and departing devices in a given period of time based on a Poisson distribution, a discrete probability distribution function used to determine the likelihood that an event will occur some number of times in a given period. The formula used to determine the probability of events occurring with a Poisson distribution takes the form:

$$P(x,\lambda)=\frac{e^{-\lambda}(\lambda)^x}{x!}$$

Equation 1. Formula for Poisson distribution [8].

where:

- x is a natural number representing some number of occurrences before time *t*,
- *e* is the base of the natural logarithm, and
- λ is a positive real number representing the average number of occurrences, *x/t*.

Values related to the behavior algorithm that is implemented to simulate device mobility are specified in the Environment Settings panel of a GSS-M. The λ component of the Poisson distribution formula is represented by the form field for *device mobility*, indicating some percentage of 100 devices as the number of occurrences, i.e. device departures/arrivals, in a given time interval. The *t* component of the Poisson distribution formula is represented by the form field for Period, indicating the frequency with which mobility-related events are re-calculated during a simulation session.

The network environment's experimental variables include *device population* and *task initiation frequency* (*TIF*) and may be specified in the same manner as *device mobility*. *Device population* simply indicates the number of devices that the GSS-M should generate prior to the start of a simulation session. *TIF* indicates the frequency with which resources are requested from the grid by clients in the network. Like *device mobility*, *TIF* is simulated by means of an algorithm based on the Poisson distribution. However, with regard to *TIF*, the λ component of the formula is represented by the value in the form field for *TIF*, indicating a percentage of 100 devices as the number of occurrences, i.e. devices, requesting resources from the grid, in a given time interval.

Similar to *device mobility*, the *t* component of the formula is represented by the form field for *period*.

3.2. Experiment Parameters

The initial series of simulation experiments were designed to determine which experiment variables, or factors, exhibited the most significant effect on the time taken to complete a grid task, as well as any interaction effects exhibited by the test factors. This was accomplished by treating each factor as an independent variable and testing it against all combinations of the other factors involved. The test values selected for each experimental variable are presented in Table 1.

Table 1. Range of values for experiment variables.

Simulation Series I		
Population	100	300
Mobility	10%	50%
TIF	10%	35%

The execution times for all successfully completed Grid Tasks initiated during the course of a simulation session were averaged and the resulting value was used to characterize that particular session configuration. These averaged results were used to perform a three-way factorial analysis of variance, in order to determine the main effects and interaction effects of the three test factors.

Table 2. Formulae for determining main effects of experimental factors.

3-Way Factorial Analysis of Variance, Main Effects	
e1 (P)	$((R_2 - R_1) + (R_4 - R_3) + (R_6 - R_5) + (R_8 - R_7)) / 4$
e2 (M)	$((R_3 - R_1) + (R_4 - R_2) + (R_7 - R_5) + (R_8 - R_6)) / 4$
e3 (T)	$((R_2 - R_1) + (R_4 - R_3) + (R_6 - R_5) + (R_8 - R_7)) / 4$

Table 3. Formulae for determining interaction effects of experimental factors.

3-WAY FACTORIAL ANALYSIS OF VARIANCE, INTERACTION EFFECTS	
e12 (PM)	$(R_1 - R_2 - R_3 + R_4 + R_5 - R_6 - R_7 + R_7) / 4$
e13 (PT)	$(R_1 - R_2 + R_3 - R_4 - R_5 + R_6 - R_7 + R_8) / 4$
e23 (MT)	$(R_1 + R_2 - R_3 - R_4 + R_5 - R_6 + R_7 + R_8) / 4$
e123 (PMT)	$(-R_1 + R_2 + R_3 - R_4 + R_5 - R_6 - R_7 + R_8) / 4$

The factor shown to have the most substantial main effect was *device mobility*. Another series of experiments was conducted to further test *device mobility* as an influential factor with finer granularity in an effort to better visualize the nature of the relationship between mobility and the time taken to complete a grid task, the latter being the metric used to gauge network efficiency. The extended range of values utilized in the second series of simulation sessions is presented in Table 2.

Table 4. Range of extended values for device mobility.

SIMULATION SERIES II					
Population	100	-	200	-	300
Mobility	0.10	0.20	0.30	0.40	0.50
TIF	0.10	-	0.25	-	0.35

3.3. Results & Analysis

We present the raw results (*average task execution time*) for all simulation sessions first, followed by inference and analysis based on these results at the end of this section. The initial series of experiments comprise 8 distinct simulation session configurations, each resulting in an average task execution time for that particular configuration. The raw results for the first series of simulation sessions are presented below, in Table 5.

Table 5. First simulation series results.

N	Population (P)	Mobility (M)	TIF (T)	Result (R_N)
1	100	10%	10%	21.25000
2	300	10%	10%	23.18462
3	100	50%	10%	37.53333
4	300	50%	10%	34.45238
5	100	10%	35%	21.82759
6	300	10%	35%	29.23741
7	100	50%	35%	50.10811
8	300	50%	35%	38.54430

After collecting these results, we determined the varying magnitudes of main and interaction effects of the independent variables (*factors*) on the dependent variable (*average task execution time*). By inserting the results from Table 5 into the appropriate formulas from Tables 2 and 3, the effects of the experiment's independent variables were calculated, resulting in the values listed in Table 6.

Table 6. Magnitude of factorial main and interaction effects.

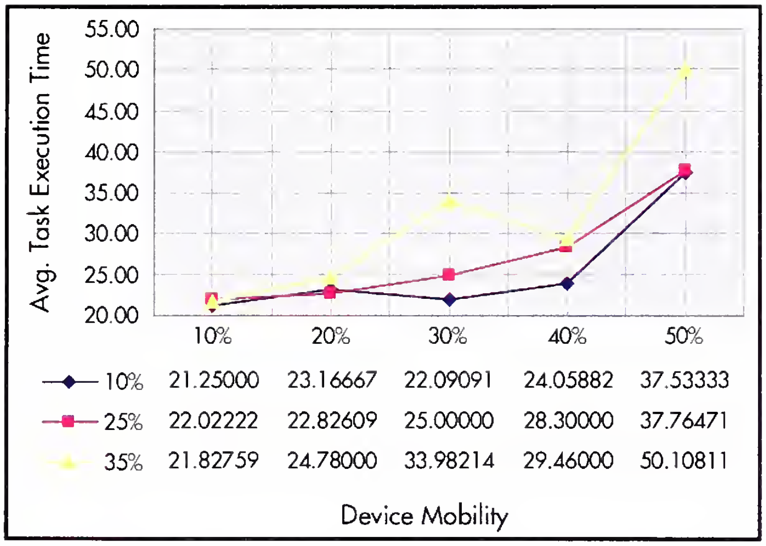
e	Result
e1 (P)	-01.32508
e2 (M)	16.28463
e3 (T)	05.82427
e12 (PM)	-05.99730
e13 (PT)	-00.75191
e23 (MT)	02.50907
e123 (PMT)	-03.48952

As illustrated above, *device mobility* is clearly shown to be the single factor which possesses the greatest effect on *average task execution time*. In order to more clearly establish *device mobility*'s relationship with *average task execution time*, a second series of simulation sessions were arranged, in which the factor was tested for additional intermediate values presented in Table 4. Raw results for the second series of simulation sessions are presented in Table 7, below. These show a finer granularity of the critical effect of *device mobility* on the *average task execution time*. Higher *device mobility* generally corresponds to higher *average task execution time* (for the same *TIF* and *initial population size*).

Table 7. Results for the second series of simulation sessions.

Mob	Tif	Pop (100)	Pop (200)	Pop (300)
10	10	21.25000	22.09677	23.18462
20	10	23.16667	28.54545	31.93878
30	10	22.09091	28.73171	24.53488
40	10	24.05882	24.26316	30.67500
50	10	37.53333	29.96429	34.45238
10	25	22.02222	23.43210	22.15789
20	25	22.82609	32.21622	37.42857
30	25	25.00000	34.12162	35.07865
40	25	28.30000	39.64286	40.65333
50	25	37.76471	38.53704	39.33871
10	35	21.82759	31.87629	29.23741
20	35	24.78000	33.00000	37.25203
30	35	33.98214	32.06422	43.66667
40	35	29.46000	40.58462	41.50000
50	35	50.10811	43.64151	38.54430

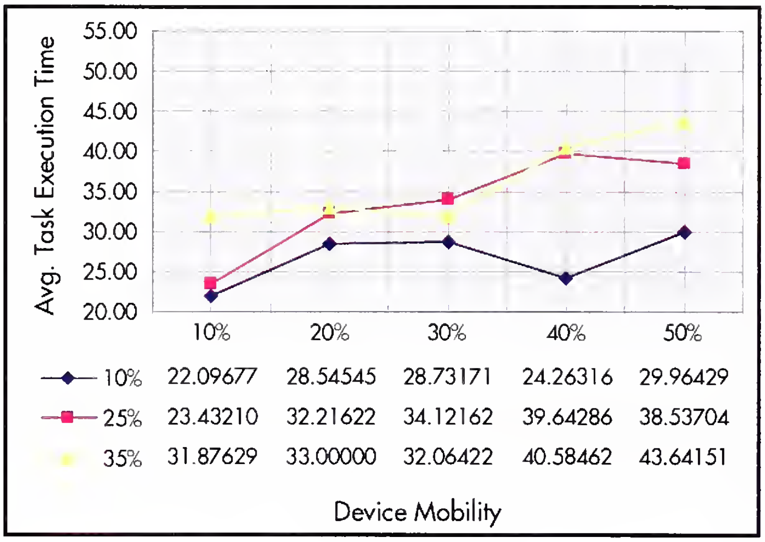
Based on these results, we prepared the following graphs to illustrate the relationship between *device mobility* and *average task execution time*. Each graph represents a data series from one of the three *initial population size*s tested. The first graph illustrates the effect of increasing *device mobility* against three values of *TIF* in a network with an *initial population size* of 100 devices.



	10%	20%	30%	40%	50%
10%	21.25000	23.16667	22.09091	24.05882	37.53333
25%	22.02222	22.82609	25.00000	28.30000	37.76471
35%	21.82759	24.78000	33.98214	29.46000	50.10811

Figure 9. Mobility vs. task execution time for an initial population of 100 devices.

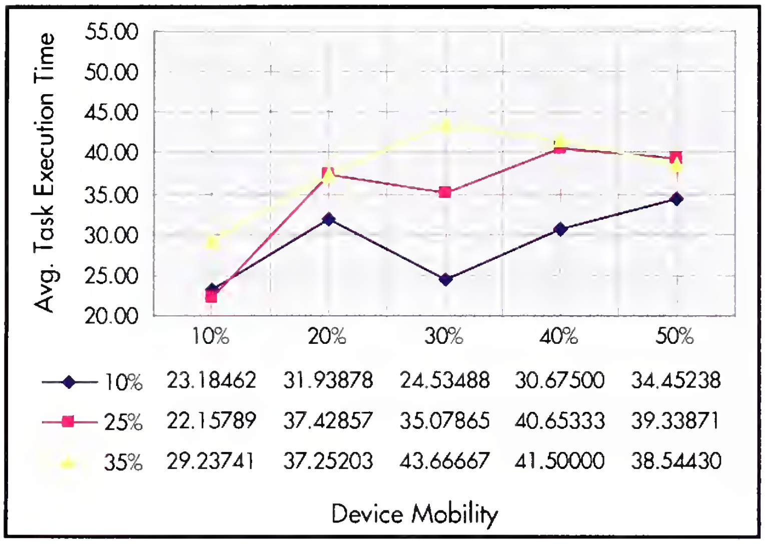
The next graph illustrates the effect of increasing *device mobility* against three values of *TIF* in a network with an *initial population size* of 200 devices.



	10%	20%	30%	40%	50%
10%	22.09677	28.54545	28.73171	24.26316	29.96429
25%	23.43210	32.21622	34.12162	39.64286	38.53704
35%	31.87629	33.00000	32.06422	40.58462	43.64151

Figure 10. Mobility vs. task execution time for an initial population of 200 devices.

The final graph illustrates the effect of increasing *device mobility* against three values of *TIF* in a network with an *initial population size* of 300 devices.



	10%	20%	30%	40%	50%
◆ 10%	23.18462	31.93878	24.53488	30.67500	34.45238
■ 25%	22.15789	37.42857	35.07865	40.65333	39.33871
▲ 35%	29.23741	37.25203	43.66667	41.50000	38.54430

Figure 11. Mobility vs. task execution time for an initial population of 300 devices.

In general, we infer from the preceding graphs that higher degrees of *device mobility* will generally impede the performance of the grid network, regardless of *initial population size* or *TIF*. However, there are plot points that deviate from this reading of the data sets; thus, this interpretation of the relationship between *device mobility* and *average task execution time* is not completely consistent with the graphical representations.

In order to clarify the patterns, trend-lines were generated for each of the data series in the first graph. The following graphs present 2nd order polynomial regression trend-lines for each series of simulation sessions with an *initial population size* of 100 devices, each graph organized by *TIF*. The first graph depicts a polynomial trend-line for five simulation sessions with an *initial population size* of 100 devices, a *TIF* of 10% and varying degrees of *device mobility*, from 10% to 50%. The R^2 value of the trend-line is 0.884, indicating a good fit for the data.

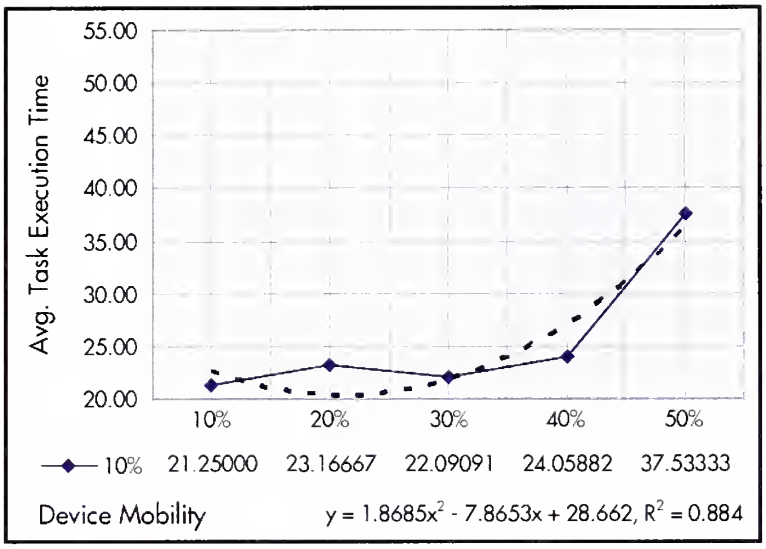


Figure 12. Polynomial regression trend-line. Initial population: 100 devices, TIF: 10%.

The next graph depicts a polynomial trend-line for five simulation sessions with an *initial population size* of 100 devices, a *TIF* of 25% and varying degrees of *device mobility*, from 10% to 50%. The R^2 value of the trend-line is 0.9835, indicating an excellent fit for the data.

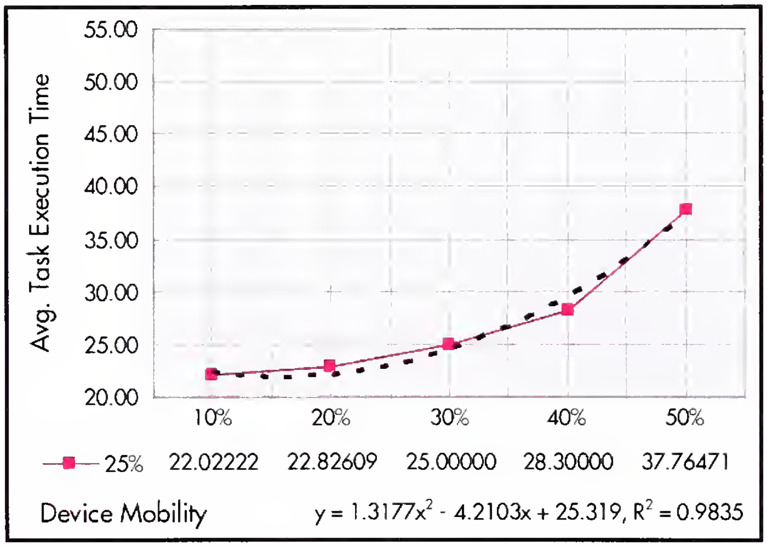


Figure 13. Polynomial regression trend-line. Initial population: 100 devices, TIF: 25%.

The final graph depicts a polynomial trend-line for five simulation sessions with an *initial population size* of 100 devices, a *TIF* of 35% and varying degrees of *device* *mobility*, from 10% to 50%. The R^2 value of the trend-line is 0.8273, indicating a good fit for the data.

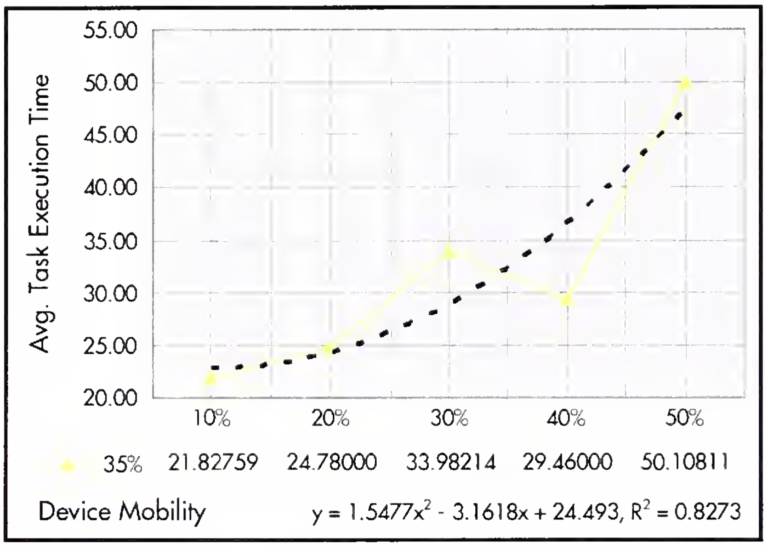


Figure 14. Polynomial regression trend-line. Initial population: 100 devices, TIF: 35%.

The formulae of the three trend-lines are similar enough to suggest a predictable and consistent effect of *device mobility* on *average task execution time* for the representative conditions. Unfortunately, this pattern simply does not seem to carry over to simulation sessions with *initial population size*s of 200 or 300 devices. Trend-lines generated for the latter two data sets are less uniform and predictable when compared to the trend-lines generated for the data sets obtained by tests on a grid with a population of 100 devices This is presumably due to some critical amount of drain on network or workstation resources resulting from the larger population sizes, however further speculation on the matter is reserved until the end of this section..

In an attempt to determine the accuracy and reliability of the simulation results, a linear regression model was created using the values listed in Table 6. A factorial analysis of the results from the first series of experiments (Table 5) produced the values in Table 6, which represent coefficients for a linear formula that would ideally yield relatively accurate predictions for *average task execution time* by plotting it as a function of *initial population size* (P), *device mobility* (M) and *TIF* (T), with some anticipated error, ε.

$$f(P,M,T) = -1.33P + 16.28M + 5.82T - 6.00PM - 0.75PT + 2.51MT - 3.49PMT + \varepsilon$$

Equation 2. Multiple linear regression formula.

By substituting the appropriate parameters tested in the second simulation session, the following table was compiled, presenting a side-by-side comparison of actual vs. predicted simulation results.

Table 8. Predicted vs. actual simulation results.

Pop (P)	Mob (M)	Tif (T)	Predicted	Actual
100	0.10	0.10	-201.2536393	21.25000
100	0.20	0.10	-263.0626056	23.16667
100	0.30	0.10	-324.8715719	22.09091
100	0.40	0.10	-386.6805382	24.05882
100	0.50	0.10	-448.4895045	37.53333
100	0.10	0.25	-216.8552928	22.02222
100	0.20	0.25	-283.8609030	22.82609
100	0.30	0.25	-350.8665133	25.00000
100	0.40	0.25	-417.8721235	28.30000
100	0.50	0.25	-484.8777338	37.76471
100	0.10	0.35	-227.2563951	21.82759
100	0.20	0.35	-297.7264346	24.78000
100	0.30	0.35	-368.1964742	33.98214
100	0.40	0.35	-438.6665137	29.46000
100	0.50	0.35	-509.1365533	50.10811
200	0.10	0.10	-404.7432593	22.09677
200	0.20	0.10	-530.0147456	28.54545
200	0.30	0.10	-655.2862319	28.73171
200	0.40	0.10	-780.5577182	24.26316
200	0.50	0.10	-905.8292045	29.96429
200	0.10	0.25	-436.8578428	23.43210
200	0.20	0.25	-572.5602530	32.21622
200	0.30	0.25	-708.2626633	34.12162
200	0.40	0.25	-843.9650735	39.64286
200	0.50	0.25	-979.6674838	38.53704
200	0.10	0.35	-458.2675651	31.87629

200	0.20	0.35	-600.9239246	33.00000
200	0.30	0.35	-743.5802842	32.06422
200	0.40	0.35	-886.2366437	40.58462
200	0.50	0.35	-1028.893003	43.64151
300	0.10	0.10	-608.2328793	23.18462
300	0.20	0.10	-796.9668856	31.93878
300	0.30	0.10	-985.7008919	24.53488
300	0.40	0.10	-1174.434898	30.67500
300	0.50	0.10	-1363.168905	34.45238
300	0.10	0.25	-656.8603928	22.15789
300	0.20	0.25	-861.2596030	37.42857
300	0.30	0.25	-1065.658813	35.07865
300	0.40	0.25	-1270.058024	40.65333
300	0.50	0.25	-1474.457234	39.33871
300	0.10	0.35	-689.2787351	29.23741
300	0.20	0.35	-904.1214146	37.25203
300	0.30	0.35	-1118.964094	43.66667
300	0.40	0.35	-1333.806774	41.50000
300	0.50	0.35	-1548.649453	38.54430

There is a substantial discrepancy between the two columns, which cannot be accounted for by some predictable degree of error. The erratic and unpredictable nature of these results implied the presence of error either in the design of the experiments or the design of the simulator itself. To better judge which was the case, we carried out further analyses of the log data generated throughout the course of the simulator sessions.

3.4. Further Analysis

Log entries (discussed in §2.2.1) were generated at intervals of 1 second for the 10 minute duration of simulator sessions, providing a detailed record of the system state throughout the course of each experiment. Based on calculations derived from values in the session logs, we were able to track seven characteristics of the simulated network in order to uncover any relevant performance issues responsible for the unpredictable nature of session results. They are presented throughout the remainder of this section as the following graphs:

1. *Initiator Population.* Displays the percentage of the total device population registered as an Initiator at each time interval.
2. *Active Subordinate Population.* Displays the percentage of the total device population registered as an active Subordinate at each time interval.
3. *Tasks Completed.* Displays the percentage of total initiated tasks (from the beginning of the session) that have been successfully completed at each time interval.
4. *Initiators Dropped.* Displays the percentage of total devices dropped (from the beginning of the session) which were Initiators at the time of departure.
5. *Active Subordinates Dropped.* Displays the percentage of total devices dropped (from the beginning of the session) which were active Subordinates at the time of departure.
6. *Device add/drop ratio.* Displays the ratio of total devices added to total devices dropped (from the beginning of the session) at each time interval.
7. *Result Execution Times.* Displays the time-to-completion for all grid tasks successfully executed throughout the course of a simulator session. Result times are listed in chronological order of completion.

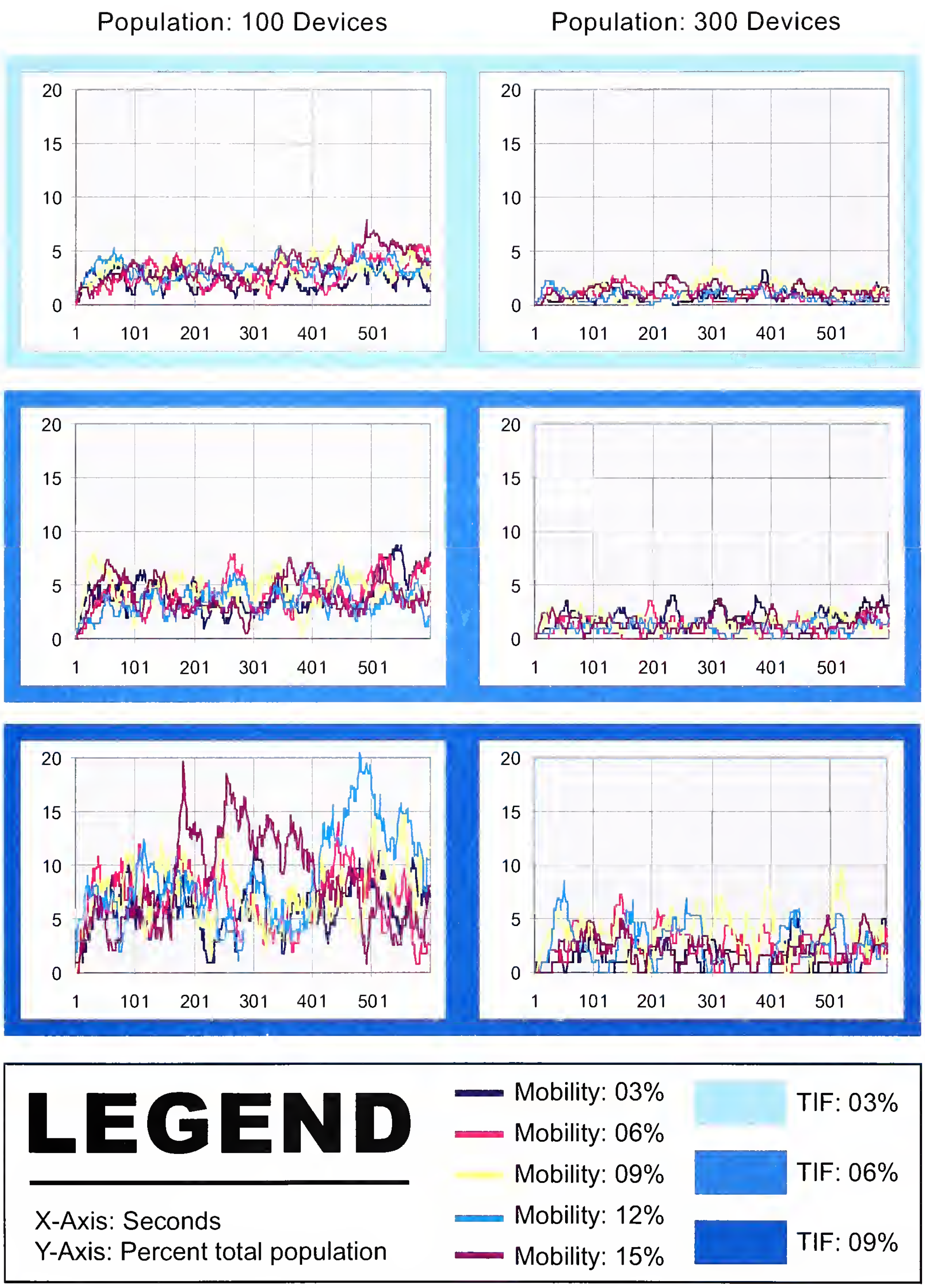


Figure 15. Initiator Population.

The number of resource requests in a given interval, i.e. once per second for these experiments, was calculated as a pseudo-random Poisson distribution value based on a specified percentage of 100 devices. For example, a *TIF* of 6% indicates that some number of Subordinates equal to a Poisson distribution value based on (100 devices * 0.06) = 6 devices would attempt to initiate a grid task every time the GSS behavior algorithms were invoked. Prior to running these experiments, we determined that a grid task could successfully execute in approximately 20 seconds, provided ideal network conditions. Thus, given the rate of resource requests and the ideal time taken for a grid task to be completed, we can anticipate reasonable values for the Initiator population, e.g. an *initial device population* of 100 devices with *TIF* of 6% should yield an Initiator population of around 6%. In Figure 15 (pg. 59), we can see that while the majority of sessions produced and maintained a reasonable number of Initiators, the simulation session configured with an initial population size of 100 devices and a *TIF* of 9% produced an inexplicably large population of Initiators.

All grid tasks that were submitted by Initiators during these experiments were configured to request a constant and consistent amount of resources from the grid network. Therefore, the general assumption is that the number of active Subordinates in the Grid at any moment would be a multiple of the number of Initiators in the Grid. Referring to Figure 16 (pg. 61), we can see that this largely seems to be the case, except for the sessions with an initial population size of 100 devices and a *TIF* of 6% and 9%. The active Subordinate populations in these sessions are showing a slow and steady trend upwards, indicating that resource requests are not being satisfied; instead, they appear to be accumulating in the grid.

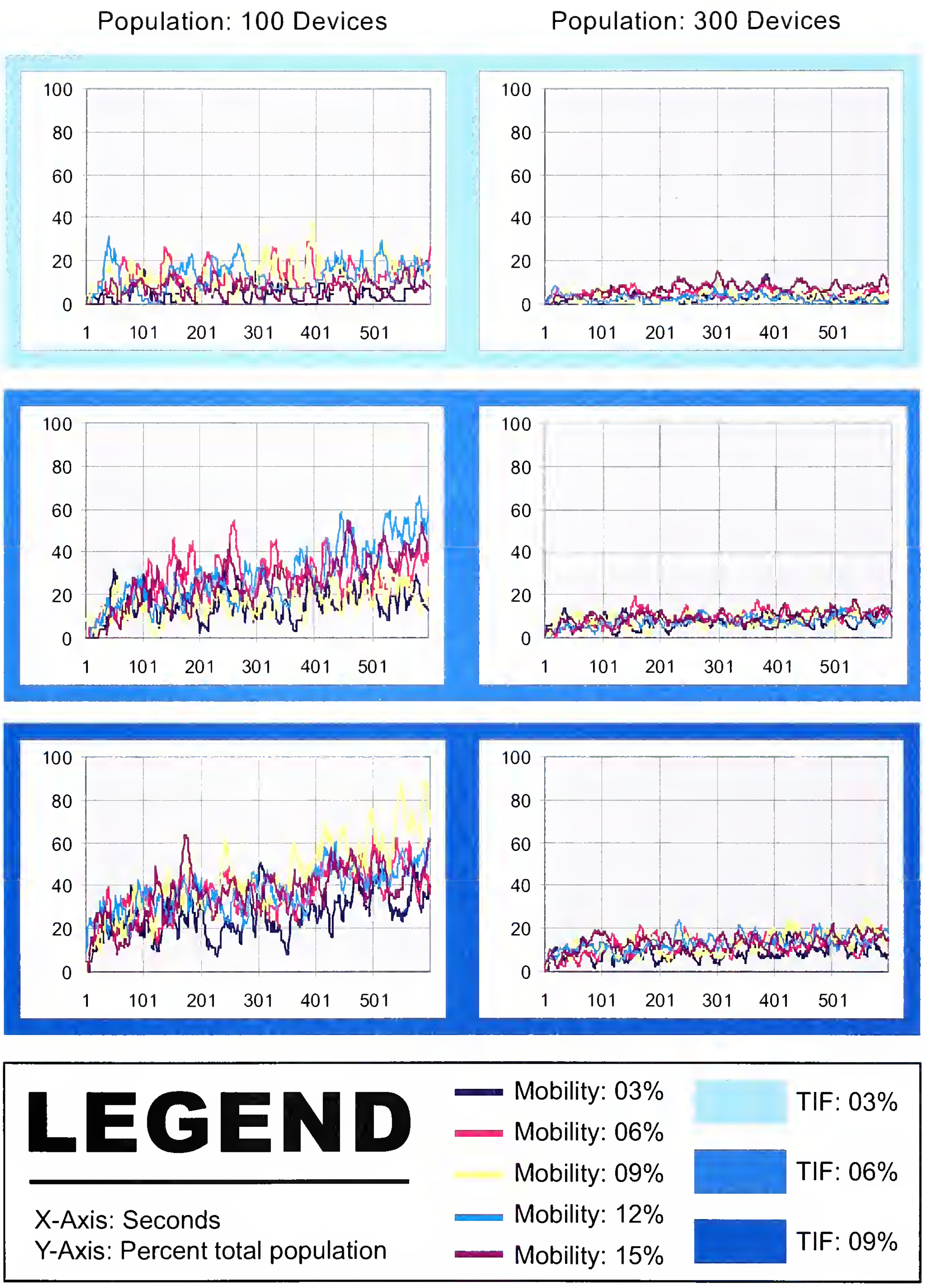


Figure 16. Active Subordinate population.

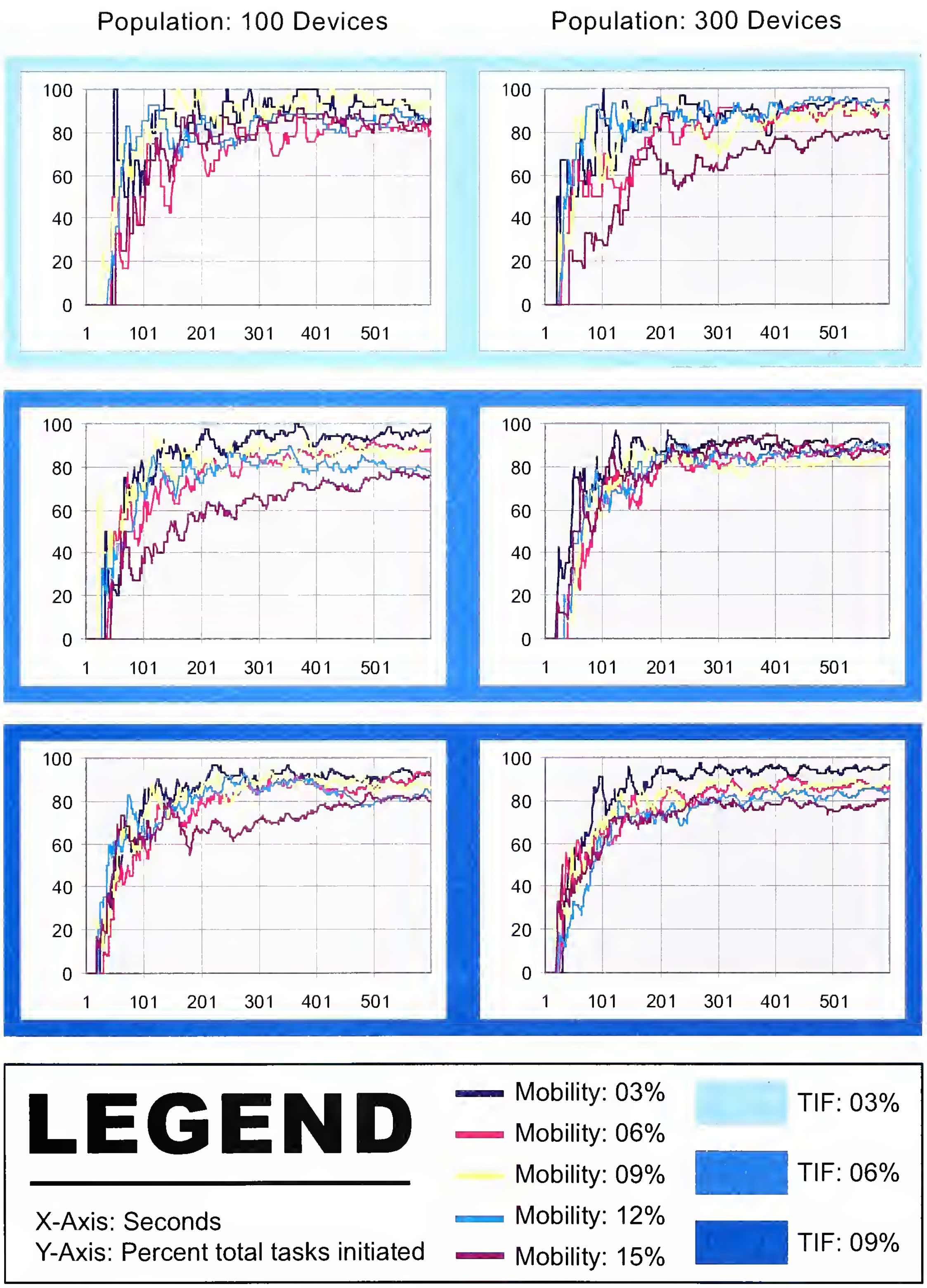


Figure 17. Tasks completed.

The graphs in Figures 15 and 16 suggest that smaller populations appear to have a negative impact on network efficiency, which can be inferred from the increasing Initiator and active Subordinate populations, indicating an accumulation of grid tasks. This inference is supported by the graphs in Figure 17 (pg. 62), which show the cumulative percentage of completed grid tasks throughout simulation sessions. In ideal network conditions, the graph values would constantly be at or approaching 100%. All the graphs are generally characterized by the same parabolic shape as the first few tasks of any simulation session are completed and the task completion ratio quickly approaches 100%. However, certain cases, such as those with initial population sizes of 100 devices and higher values of *TIF*, begin to deviate from this characteristic shape as configurations with higher *device mobility* settings begin to lag from resource requests being introduced into the system more quickly than they can be satisfied.

Additionally, fewer completed tasks in the network could be the result of Initiators or active Subordinates prematurely departing from the network. Active Subordinates that drop out of the network prior to completing their assigned sub-task would require the GSS-Broker to re-assign the sub-task. Unfortunately, in an unstable network topology, a device receiving a re-assigned sub-task may drop out as well, resulting in a condition where a sub-task could be juggled within the system indefinitely. In cases where an Initiator leaves the network, the associated task is simply aborted and network resources, i.e. active Subordinates, are released. To ascertain the degree to which either of these scenarios contributed to the poor task completion ratios of the sessions with small *device populations* and high *TIF*, we present Figures 18 and 19, which display the percentage of total devices dropped as either Initiators or active Subordinates.

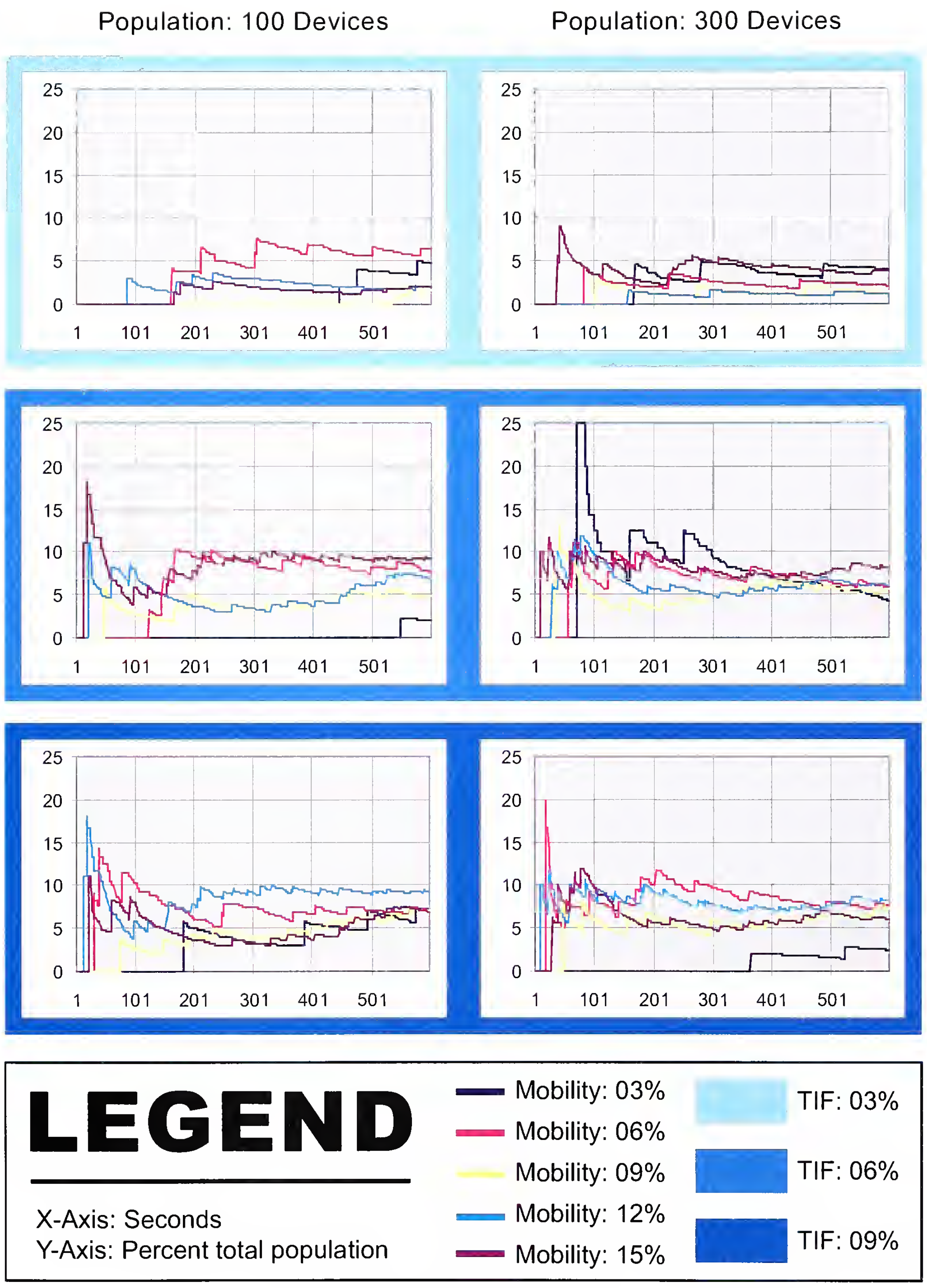


Figure 18. Initiators dropped.

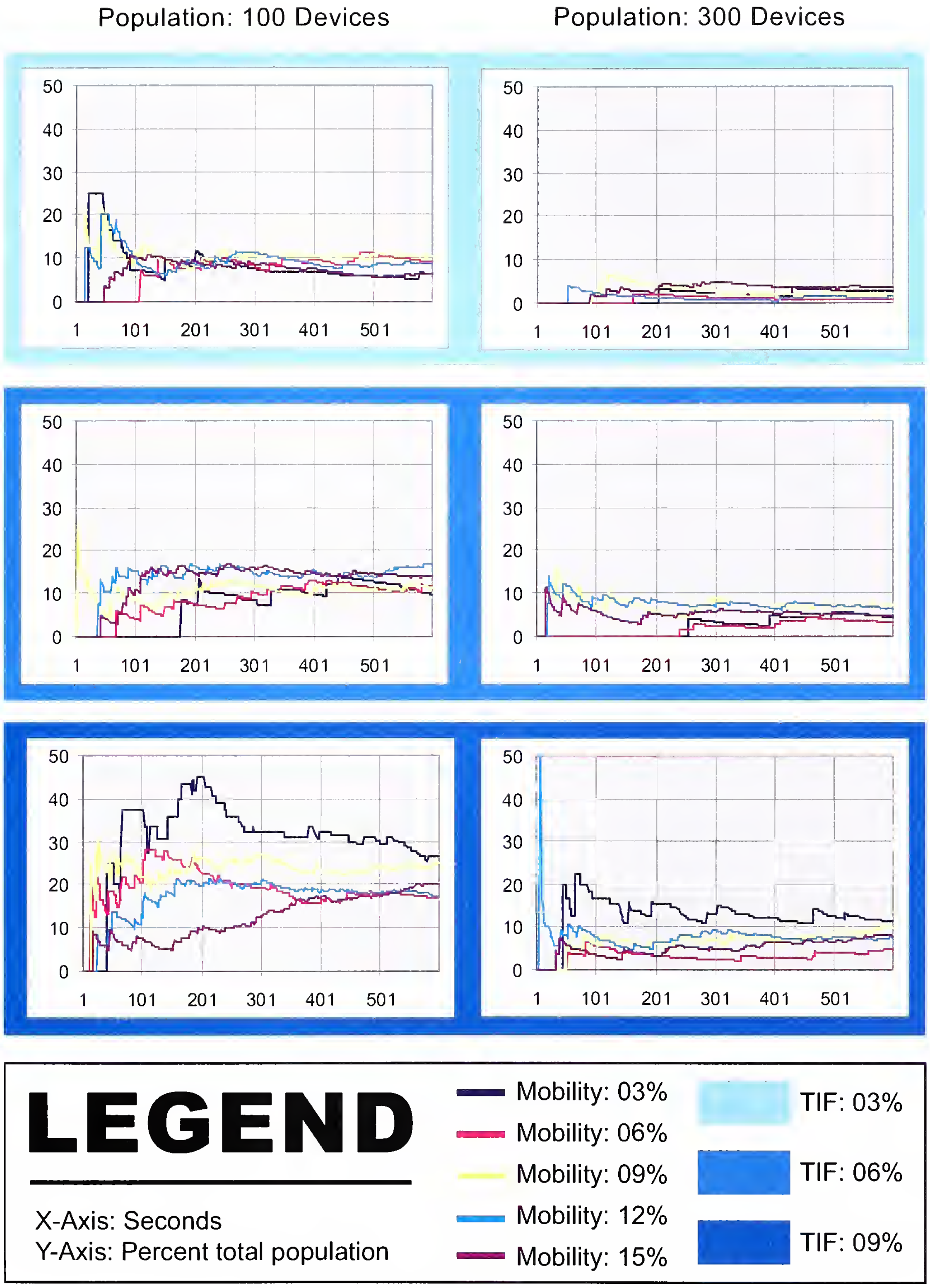


Figure 19. Active Subordinates dropped.

Based on the values presented in Figure 18 (pg. 64), Initiators account for a consistent percentage of dropped devices, typically between 5 and 10%. Likewise, active Subordinates (Figure 19, pg. 65) appear to have been dropped from the network at a consistent rate, accounting for 10% ± 5% of all dropped devices. One exception of note is in the session configured with an *initial device population* of 100 devices and a *TIF* of 9%, in which active Subordinates constitute 20% ± 5% of all dropped devices. Initiators comprise a minimal portion of all dropped devices, which suggests that most of the incomplete grid tasks in Figure 17 are not the result of Initiators departing before their resource requests are satisfied, i.e. task abortion, but more likely the result of the GSS being overwhelmed by resource requests or inefficiently recovering from aborted sub-tasks.

To make certain that the algorithms responsible for emulating *device mobility* were operating as expected, we calculated and graphed the values presented in Figure 20 (pg. 66), which show the ratio of devices added to devices dropped throughout a simulation session. A growing population is represented by values greater than 1; a shrinking population is represented by values less than 1; and stasis is represented when the line is at 1. In most cases, the population appears to begin settling into equilibrium around the 2 – 3 minute mark. This behavior seems reasonable, considering that the same pseudo-random function is invoked at identical intervals to emulate the arrival and departure of devices from the network.

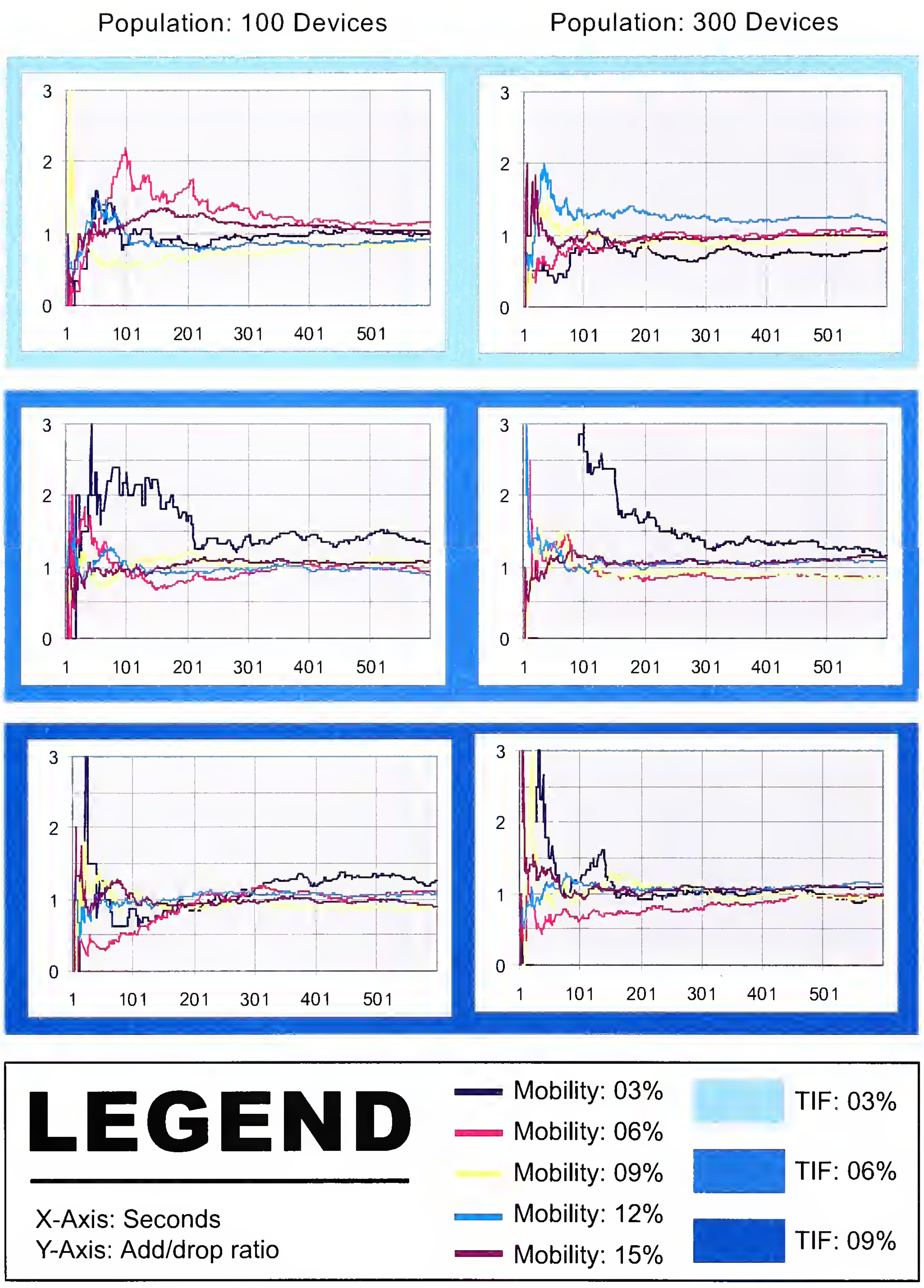


Figure 20. Device add/drop ratio.

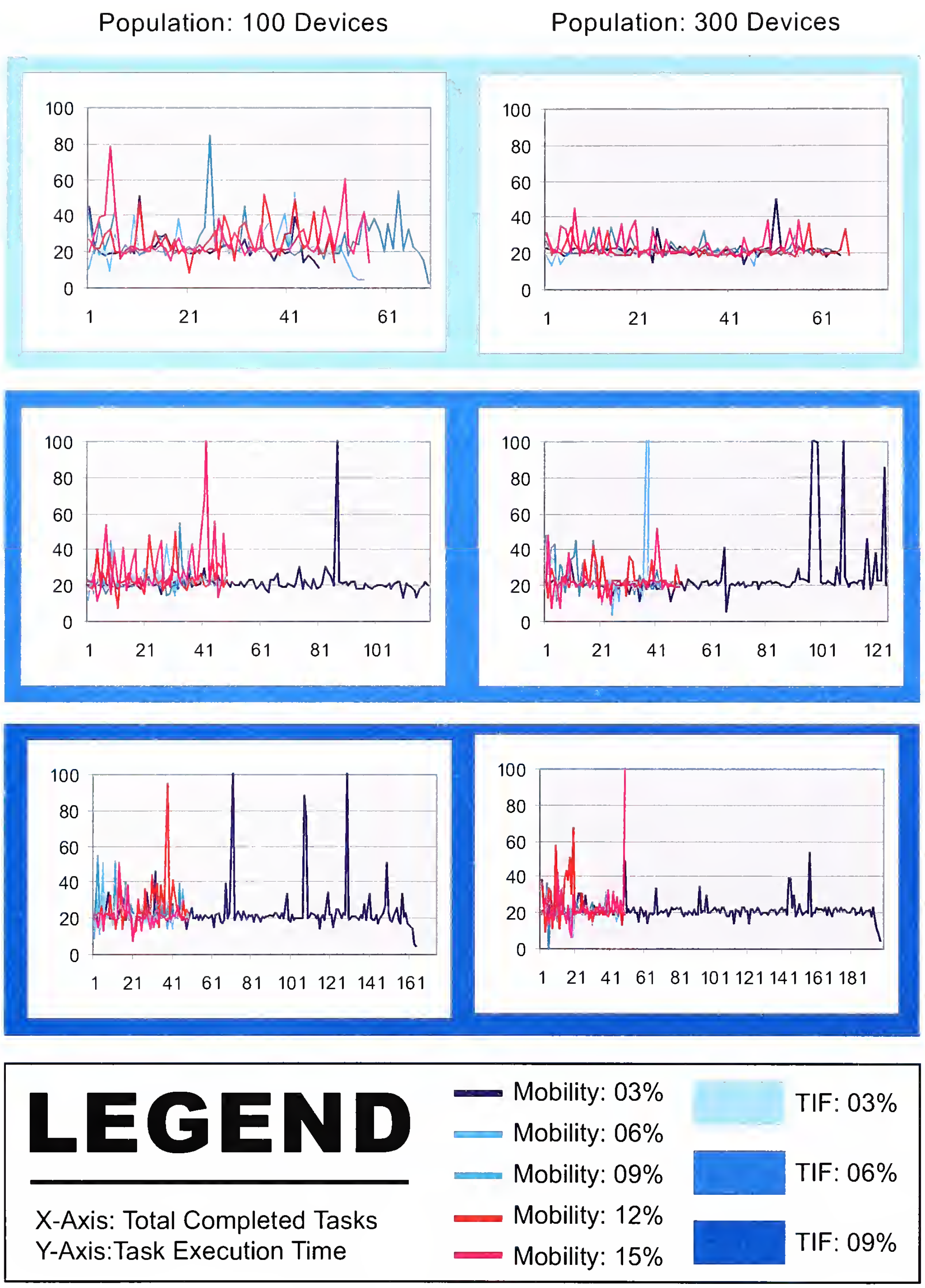


Figure 21. Task execution times.

Finally, we examined the execution times for all successfully completed grid tasks, presented in Figure 21 (pg. 68). Accounting for real-world factors, such as network conditions and the unexpected departure of Initiators and active Subordinates, we assume a range of task completion times, from 20 – 60 seconds, to be reasonable. However, some of the graphs in Figure 21 show tasks with inexplicably large completion times, in excess of 100 seconds. Another anomaly is the disproportionately large number of completed tasks for cases with *device mobility* of 3%. Such a large disparity in the number of completed tasks for cases with similar *TIF* indicates that there is a very low performance threshold for *device mobility* beyond which the GSS begins to suffer. Figure 18 reveals that Initiators are a minor constituent of total devices dropped, thus we can infer that the debilitating impact on task completion mainly derives from Subordinates exhibiting a higher degree of mobility. This suggests that the GSS-Broker experiences difficulty with re-assigning sub-tasks as active Subordinates are dropped from the network population.

Based on the preceding graphs and our observations of the simulator's performance, we provide the following list of conclusions and recommendations for future development of the GSS:

1. *Physical limitations of the host environment influenced the accuracy of the results.* There may have been hidden bottlenecks in the hosting environment that adversely affected the execution of simulation sessions. The web server hosting the GSS-Broker may have been unable to cope with the strain of administrating larger device populations, for example, 300 devices. Additionally, client machines that hosted a GSS-M may have been unable to cope with administrating the behavior and functionality of

devices generated for simulation tests. Solutions include monitoring processor, memory and network usage of the server hosting the GSS-Broker and of any machines hosting a GSS-M during simulation sessions to ensure they are not being taxed too heavily.

2. *An alternative/amended analysis of the logged data is required.* There appears to be a configuration threshold beyond which the performance of the GSS beings to seriously degrade, i.e. session configurations with a low initial *device population*, high *TIF* and/or high *device mobility*. The data represented in the graphs may require more sophisticated interpretations than those we have supplied to determine what the cause of this poor performance may be. Additionally, there may be more salient aspects of the GSS that we failed to include in the session logs. Recommendations include conducting further simulation tests and executing each simulation scenario multiple times to determine if the performance trend in each graph remains consistent. Additionally, session logs should be implemented on the GSS-Broker to track its performance during simulation sessions.
3. *The execution parameters selected for simulation sessions are inappropriate.* The values that were selected and tested for each of the three independent variables were chosen as "best-guess" values. The possibility exists that they could have been chosen to cover a greater or lesser range to reveal any latent trends in the operating efficiency of the GSS-Broker. A method to resolve this possibility would be to run two

small series of tests, with exaggerated values chosen for the independent variables, to determine if there is greater consistency in the trend of the results for either series.

4. *Better or additional metrics could have been chosen to evaluate network performance.* Rather than relying on average task completion time as the sole indicator of network performance, it may also be worthwhile to track other factors, such as recovery time for the GSS-Broker when aborting grid tasks or re-assigning sub-tasks. Also, interaction between the chosen environment parameters, i.e. *initial population size*, *device mobility* and *TIF*, may not as strong as with some other unknown factor. Consideration should be made for other simulation parameters that may be included in addition or in lieu of current ones.

4. CONCLUSIONS AND FUTURE WORK

The GSS infrastructure presents a modular and extensible framework for simulating a resource-sharing wireless grid comprised of mobile devices. The features provided by the framework are minimal because the emphasis of the current system is on the provision of a strong foundation for future development rather than that of a complete, self-contained production-quality implementation. That foundation, as outlined in this thesis, is sufficiently established to effectively permit limited resource-sharing among a population of virtual devices. In addition, the simulator proves our initial concept that a wireless grid comprised of resource-sharing mobile devices can successfully execute a computationally intensive task.

Due to time constraints, there remain a few unimplemented features that were included in the initial draft of the GSS infrastructure but not included in our implementation of the simulator. The following recommendations address the extension of the GSS address as work that remains to be developed by future researchers.

1. Extend the collection of device properties and device types.
2. Develop more sophisticated recovery mechanisms for connection loss.
3. Add support for sharing a greater variety of resources.

The first recommendation refers to the Device object model currently utilized by the GSS, consisting of the properties outlined in §2.3.2.1. Of the properties listed, Processor Speed, Physical Memory and Power Level are not yet fully implemented as intended. Processor Speed and Physical Memory are currently used by the Brokering Service in an algorithm used to calculate the relative eligibility of inactive Subordinates when assigning sub-tasks. However, these properties can also serve as a modifier to influence the relative speed with which devices accomplish the execution of sub-tasks.

Furthermore, Power Level is included in the Device object property collection, but has not been implemented in the current version of the GSS. Power Level is intended to indicate the level of charge remaining in the battery supply of mobile devices in the network. By incorporating an algorithm in the Device object to model different drain rates on the Power Level based on device activity, we can observe the effects of low battery on the stability of devices in the grid, making the simulation more realistic.

The second recommendation addresses the fairly rigid set of choices available on the GSS when Initiators unexpectedly drop out of the network. When Initiators leave the grid without warning, their associated tasks are aborted and results discarded. However, this fixed response does not effectively consider the ephemeral nature of wireless connections and their susceptibility to brief or momentary disconnects; rather, this solution treats every disconnection as a critical and unrecoverable event. Alternative measures should be implemented on the GSS to recover from the unexpected departure of Initiators. For example, instead of deleting grid tasks and discarding any accumulated results, mechanisms could be developed for "pausing" the execution of a task, for storing the results temporarily, and for deleting tasks only if the absent Initiator is unable to re-register with the GSS within a fixed period of time.

The third recommendation for improving the GSS is that the system extends support for the sharing of a greater variety of resources. The current system is only capable of managing requests for processing power. However, the infrastructure for the GSS should consist of services that view a resource in abstract terms, with an API instituted for future developers to extend the GSS, permitting it to handle specific types of resources in the future, such as data storage systems or remote applications. Further experiments could then be conducted on the effects of mobility in resource-sharing networks, with special emphasis on different types of resources, such as the sharing of processing power or the sharing of disk space.

5. Summary

We have described our work in determining the effects of device mobility on resource-sharing in grid networks. The first section of this work presented an introductory survey of technologies relevant to our work, such as grid networks, grid protocols and services, mobile computing and the history of wireless communication.

Then we described the design and architecture of a wireless grid system comprised of resource-deficient mobile devices. In order to be compatible with current wired grids and internetworking systems, the design was developed on well-known standards and uses popular technologies.

In the second section of this paper, we discussed the architecture of the simulation system used for experimentation and testing. The process and parameters of experimentation are presented in the third section. Descriptions of the experiments, results collected and an analysis of the results was also discussed. An analysis of the results clearly indicates that the presence of mobility in a resource-sharing network impedes its performance, as measured by average task completion time. Since graphical representations showed an inconsistent relationship between mobility and network performance, we attempted to explain the possible reasons for this apparent anomaly. We concluded with recommendations for future research to extend our work.

Annotated Bibliography

[1] P. Avery and I. Foster, "The GriPhyN Project: Towards Petascale Virtual Data Grids," 2001; http://www.griphyn.org. *The Grid Physics Network (GriPhyN) project is a grid network implementation that provides a distributed and collaborative environment capable of supporting Petascale data processing and analysis for computationally intensive scientific research. The requirements which initially shaped the GriPhyN arose from four NSF-funded frontier physics projects associated with the GriPhyN project.*

[2] M. Barbeau, "Mobile, Distributed, and Pervasive Computing," in: I. Stojmenovic, Chapter 27 - Handbook of Wireless Networks and Mobile Computing, John Wiley and Sons, Inc., 2002. *Excerpted from a larger work on mobile computing that provides a good introduction to the paradigm of pervasive computing. Technologies such as mobile computing, service discovery and distributed computing are discussed as elements necessary for successfully establishing a pervasive computing environment.*

[3] D. Bernholdt, *et al.*, "The Earth System Grid: Supporting the Next Generation of Climate Modeling Research," Proceedings of the IEEE, Vol. 93, No. 3, March 2005. *Building upon the Globus toolkit, the Earth Systems Grid is a grid project that seeks to establish a collaborative grid environment, allowing scientists to manage, discover, access and analyze potentially Petascale distributed datasets, resulting from global climate model simulations.*

[4] V. Berstis, "Fundamentals of Grid Computing," Technical Report, IBM Redbooks, November 11, 2002. *A comprehensive primer on grid computing,*

presenting grid concepts from the perspectives of a grid user, administrator and application developer.

[5] Bluetooth Special Interest Group, "Specification of the Bluetooth System, Core, v. 1.1," Joint specification by Agere, Ericsson, IBM, Intel, Microsoft, Motorola, Nokia and Toshiba, February 22, 2001; https://www.bluetooth.org/spec/. *The industry specification for Bluetooth wireless technology, which is both precursor and peer to the current ad-hoc standard of the IEEE 802.11x family of protocols.*

[6] D. Booth, *et al.*, Ed., "Web Services Architecture," World Wide Web Consortium (W3C) note, February 11, 2004; http://www.w3.org/TR/ws-arch/. *Web services are a fundamental component of grid technologies and this W3C note provides a thorough overview of the W3C recommended standard of Web Services Architecture.*

[7] R. Buyya, D. Abramson and J. Giddy, "A Case for Economy Grid Architecture for Service Oriented Grid Computing," Proceedings of the 10th Heterogeneous Computing Workshop, April 23 – 27, 2001. *The case is made that a computational economy for regulating the demand and supply of grid resources is necessary for creating a truly scalable grid system. This grid economy ensures incentive for grid resource owners to participate in the grid and encourages resource consumers to optimize their usage of grid resources.*

[8] C. Croarkin and P. Tobias, Eds., "NIST/SEMATECH e-Handbook of Statistical Methods," NIST/SEMATECH, March 2005; http://www.itl.nist.gov/div898/handbook/. *An e-book published by the National*

Institute for Standards and Technology, which is a publicly available reference for fundamental methods of statistical analysis.

[9] B.P. Crow, *et al.*, "IEEE 802.11 Wireless Local Area Networks," *IEEE Communications Magazine*, Vol. 35, No. 9, September 1997. *This article presents a summary overview at the initial draft of the IEEE 802.11 specification with particular attention paid to the Medium Access Control (MAC) sub-layer.*

[10] L. Ferreira, *et al.*, "Grid Computing in Research and Education," Technical Report SG24-6649-00, IBM Redbooks, April 2005. *Details an extensive selection of potential grid applications based on real-world grid implementations from the scientific and educational communities.*

[11] L. Ferreira, *et al.*, "Grid Services Programming and Application Enablement," Technical Report SG24-6100-00, IBM Redbooks, May 2004. *Presents concepts related to the Open Grid Services Architecture (OGSA), Open Grid Service Infrastructure (OGSI) and the Globus Toolkit v. 3.0 with supplemental examples for developing a grid service application.*

[12] L. Ferreira, *et al.*, "Introduction to Grid Computing with Globus," Technical Report SG24-6895-01, IBM Redbooks, September 2003. *Presents introductory concepts of grid computing, potential applications of grid technology and some examples of real-world grid implementations. The text is supplemented by an introduction to the Globus Toolkit and a walk-through demonstration of a grid application.*

[13] A. Fog, "Non-Uniform Random Number Generators," July 13, 2004; http://www.agner.org/random/. *Provides access to software libraries for the*

generation of uniform and non-uniform random numbers with various distribution methods. These libraries are available in C++ and assembly language.

[14] G. Forman and S. Zahorjan, *The Challenges of Mobile Computing*, Technical Report 93-11-03, University of Washington, Department of Computer Science & Engineering, March 1994. *Surveys the paradigm of mobile computing and the challenges that arise from influential design factors, such as the use of wireless networking, device mobility and device portability.*

[15] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid," International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001; http://www.globus.org/research/papers/anatomy.pdf. *Describes the need for grid technology and proposes an extensible, open grid architecture, as well as inter-grid protocols, ensuring successful inter-operability amongst various grid implementations.*

[16] I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," Proceedings of the Workshop on Environments and Tools for Parallel Scientific Computing, SIAM, Lyon, France, August 1996. *Proposes a low-level developer toolkit (the "Globus Toolkit") for creating networked virtual super-computers capable of exploiting geographically diverse, heterogeneous resources for high-performance computing.*

[17] I. Foster, *et al.*, "The Physiology of the Grid," Open Grid Service Infrastructure WG, Global Grid Forum, June 22, 2002; http://www.globus.org/research/papers/ogsa.pdf. *Proposes an Open Grid Services Architecture (OGSA) capable of integrating the grid services made available by*

disparate "virtual organizations," enabling the large-scale aggregation of geographically distributed, heterogeneous resources characteristic of grid networks.

[18] N. Golmie, N. Chevrollier, and O. Rebala, "Bluetooth and WLAN Coexistence: Challenges and Solutions," IEEE Wireless Communications Magazine, December 2003. *Examines the interference effects that arise from the presence of Bluetooth and WLAN transmissions in close proximity and proposes techniques for circumventing these time and frequency collisions.*

[19] Hewlett-Packard, "Understanding Wi-Fi," Technical Report, Hewlett-Packard, January 2002; http://www.hp.com/rnd/library/pdf/understandingWiFi.pdf. *Survey on the evolution of the IEEE 802.11 family of protocols and introduction to the concept of radio frequency (RF) propagation. Includes a description of FCC regulations governing the use of the RF spectrum and technologies used to enforce these regulations.*

[20] *IEEE Standard 802.11, Wireless LAN Medium Access Control, (MAC) and Physical Layer (PHY) Specifications*, IEEE, September 1999. *The original IEEE recommendation for wireless telecommunications and information exchange in metropolitan (WMAN) and local area networks (WLAN) occupying the unlicensed Industrial, Scientific and Medical (ISM) frequency band of 2.4GHz . Includes the specifications for the Medium Access Control (MAC) and Physical (PHY) Layers of compliant architectures.*

[21] *IEEE Standard 802.11a, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications High-speed Physical Layer in the 5 GHz*

Band, IEEE, September 1999. *Amendment to the original IEEE 802.11 recommendation specifying PHY sub-layer operation in the unlicensed national information insfrastructure (U-NII) band of 5GHz, allowing for greater communications bandwidth than that which is provided by the IEEE 802.11 or 802.11b specifications.*

[22] *IEEE Standard 802.11b, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*, IEEE, September 1999. *Amendment to the original IEEE 802.11 recommendation with extended specifications for the PHY sub-layer in WLANs occupying the ISM band of 2.4GHz, allowing for greater communications bandwidth than that which is provided in the IEEE 802.11 specification.*

[23] *IEEE Standard 802.11g, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 4: Further Higher Data Rate Extension in the 2.4 GHz Band*, IEEE, September 2003. *Amendment to the original IEEE 802.11 recommendation with extended specifications for the PHY sub-layer in WLANs occupying the ISM band of 2.4GHz, allowing for even greater communications bandwidth than that which is provided in the IEEE 802.11b specification.*

[24] W. E. Johnston, D. Gannon and B. Nitzberg, "Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid," Presented at the 8th IEEE International Symposium on High Performance Distributed Computing, August 3 – 6, 1999. *Based on the Globus meta-computing system, NASA's Information Power Grid (IPG) is a project focused on developing*

a networking infrastructure capable of locating, aggregating, integrating and managing computer-based resources throughout the NASA enterprise.

[25] S. Kurkovsky and Bhagyavati, "Agent-Based Distributed IDA* Search Algorithm for a Grid of Mobile Devices," in Proceedings of The 7th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI-2003), Orlando, FL., July 27-30, 2003. *Proposes a wireless grid network architecture modeled on cellular network systems and an IDA*-based search algorithm that is designed to operate in such distributed environments.*

[26] A. Natrajan, *et al.*, "The Legion Grid Portal," Submitted to Grid Computing Environments, Concurrency and Computation: Practice and Experience, 2001. *Describes the Legion Grid Portal, which provides an intuitive interface to an underlying grid system and is structured to accommodate diverse grid implementations. The portal implementation discussed is structured using current web technologies to operate over the Legion grid infrastructure.*

[27] P. Nedeltchev, *Wireless Local Area Networks and the 802.11 Standard*, Cisco Systems, Technical Report, March 31, 2001; http://www.cisco.com/warp/public/784/packet/jul01/pdfs/whitepaper.pdf. *Survey and analysis of the IEEE 802.11, 802.11a and 802.11b specifications. Discusses various aspects of these standards, as well as documented issues and developmental direction of the 802.11 specification.*

[28] J. Oraskari, *Bluetooth versus WLAN IEEE 802.11x*, Technical Report, Department of Computer Science and Engineering, Helsinki University of Technology, October 27, 2000. *Technical analysis of the Bluetooth and 802.11 protocols and*

comparison of their respective features with regard to home-WLAN market technologies.

[29] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," Fifteenth ACM Symposium on Principles of Distributed Computing, May 1996. *Survey of issues relevant to the paradigm of mobile computing and presentation of potential research topics related to developments in the field.*

[30] B. Spencer, Jr., *et al.*, "NEESGrid: A Distributed Collaboratory For Advanced Earthquake Engineering Experiment And Simulation," 13th World Conference on Earthquake Engineering, August 1 – 6, 2004. *The Network for Earthquake Engineering Simulation Grid (NEESGrid) is the central component of the NEES project, which allows geographically diverse teams of earthquake researchers to collaborate on increasingly complex earthquake simulations using leading-edge computing resources and research equipment*

[31] W-S. Tan, Ed., "Infrared Data Association Serial Infrared Physical Layer Specification," Joint specification by Hewlett-Packard, IBM, Vishay and Sharp, February 6, 2001. *Presents the PHY Layer specifications for serial infra-red (SIR) data transmissions enabling information exchange between electronic devices, such as computers and computer peripherals, as ratified by the Infra-red Data Association (IrDA).*