CONTEXT AWARE ENERGY CONSERVATION IN PERVASIVE COMPUTING ENVIRONMENTS

by

PRATHIBA JOSEPH

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON
AUGUST 2006

To my parents, family and friends

ACKNOWLEDGEMENTS

I would like to extend my gratitude to my advisor, Dr. Mohan Kumar, for giving me an opportunity to work on this challenging research topic and providing me the right guidance and support through the course of this research. I am grateful to Mr. Mike O'Dell and Dr. Yonghe Liu for serving on my thesis committee.

I am very grateful to my parents and family who have been a constant source of inspiration during my entire academic career. I would like to thank Sukruth and Shreyas for their invaluable help and advice all the way. I would like to extend my thanks to my sister, Suchitha, for her constant encouragement and moral support.

A word of appreciation to Derek, Sunil, Susanth and my friends at the PICO lab for their help during the implementation of this work.

June 20, 2006

ABSTRACT

CONTEXT AWARE ENERGY CONSERVATION IN PERVASIVE COMPUTING ENVIRONMENTS

Publication No._____

Prathiba Joseph, M.S.

The University of Texas at Arlington, 2006

Supervising Professor: Mohan Kumar

Extending battery lifetime is one of the most critical and challenging problems in mobile systems. The greatest utility of mobile devices is their ability to be used anywhere, and at anytime. But power limitations of these devices seem to hinder this goal. The ever-growing needs of mobile users for increased lifetime of wireless devices imply that emerging wireless systems must be more energy-efficient than ever before.

Cyber foraging or remote resource exploitation may be an efficient way to deal with this problem in a pervasive computing environment. Mobile devices can save battery power by migrating tasks to a nearby wired infrastructure or to other wireless devices in the environment with higher battery capacity and processing power. However, this process requires considerable automation to minimize energy consumption and user distraction. Also, a pervasive computing environment is highly dynamic and the contexts in the environment change rapidly as devices enter and leave the network. Thus, it is important that the devices are aware of the changing context and adapt to these changes accordingly. This leads to the challenge of how the devices would detect these changes and secondly how they would adapt to these changes after they are detected.

A middleware service framework has been developed and deployed over a network of machines that exploit remote resources. The framework adopts a context-aware approach to make intelligent decisions on task migration and uses the achievable throughput in an end-to-end path as the context. The throughput achievable to each of the remote devices is measured and stored, and this is used to examine the trade-off between communication power expenditures and the power cost of local processing.
For a set of tasks, energy savings of up to 43% are achieved through this process of context aware energy conservation.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv

ABSTRACT v

LIST OF FIGURES x

1 INTRODUCTION 1

1.1 Motivation 2

1.2 Contribution and Scope 3

1.3 Organization 4

2 BACKGROUND AND LITERATURE REVIEW 5

2.1 Pervasive Computing 5

2.1.1 History 6

2.1.2 Related Fields 8

2.1.3 Challenges in Pervasive Computing 10

2.1.4 Ongoing Work 11

2.2 Energy Consumption 13

2.2.1 Sources of power consumption 14

2.2.2 Energy Conservation Techniques 15

2.3 Resource Exploitation or Cyber Foraging 20

2.3.1 Challenges 20

2.3.2 Related Work 21
2.4 Context Awareness 22
2.4.1 Context-Aware Systems 23
2.4.2 User Intent 23
2.4.3 Context awareness 23
2.4.4 Adaptation 24
3 PROPOSED SCHEME: ConAEC 26
3.1 Context Aware Energy Conservation 26
3.1.1 Related Work 27
3.1.2 Context Aware Energy Conservation Techniques 28
3.2 Cyber foraging and Energy Conservation 29
3.2.1 Determining the set of tasks that can be migrated 29
3.2.2 Measuring the energy spent in communication 29
3.2.3 Measuring the energy consumed by the local execution of a task 30
3.2.4 Related Work 30
3.3 Context Aware Energy Conservation with Cyber foraging 31
3.4 Energy Model 33
3.4.1 Decision making 33
3.4.2 Checking for correctness of the decision 33
3.4.3 Predicting Energy Consumption 35
3.4.4 Estimating t_{rproc}, t_{proc} and τ 37
3.5 Algorithms 37

3.5.1 Procedure to make an offload decision 37

3.5.2 Procedure to estimate the energy for local execution 39

3.5.3 Procedure to estimate the energy for remote execution 40

3.6 Mobility and Service Discovery 40

4 SYSTEM MODEL 42

4.1 Pervasive Information Community Organization (PICO) 42

4.1.1 Architecture of the PICO framework 42

4.2 ConAEC 44

4.2.1 ConAEC Agents 44

4.2.2 Control flow 46

4.2.3 Updating Parameters 49

4.3 Environment 50

4.4 Software 51

5 IMPLEMENTATION AND RESULTS 52

5.1 Devices and Environment 52

5.2 Power Specifications 53

5.3 Measurement of actual energy consumption 53

5.4 Assumptions and Limitations 54

5.5 Results 54

6 CONCLUSIONS AND FUTURE WORK 64

REFERENCES 65

BIOGRAPHICAL STATEMENT 72

LIST OF FIGURES

Figure Page

2.1: PICO’s layered structure..13

4.1: A Device Tuple..43

4.2: A Delegent Tuple..43

4.3: ConAEC Framework..46

4.4: Making a service request for an offload decision..47

4.5: Offloading the task to a remote device..48

4.6: Executing the task on the local device..49

4.7: Updating parameters..50

5.1: Remaining battery energy with time..55

5.2: Energy savings for ConAEC..56

5.3: Energy consumption at different throughput values..57

5.4: Energy savings for ConAEC at different throughput values........................58

5.5: Adaptation in ConAEC..59

5.6: Adaptation in the static scheme..60

5.7: Energy savings for different task sizes..61

5.8: Overhead due to ConAEC..63

CHAPTER 1

INTRODUCTION

In the past fifty years of computing history, there have been two evolutionary trends that have altered the place of technology in our lives; the mainframe and the PC. Today the internet has brought a wave of technological change that is unprecedented in its sweep. The development in wireless technology has made revolutionary changes in the last decade. Laptops, Palmtops, smart phones and other handheld devices enabled with wireless capabilities are used for communication. Computing has truly accelerated the speeds at which mankind works, plays, and relaxes. Since its onset, computing technology continues to advance at an exponential rate.

Over the years, ever smaller, portable computers came on the market and wireless networking technology evolved, leading to the emergence of mobile and wireless computing. Embedded computing gave us small devices, sensors and actuators, with increased communication capabilities. The technological potential for pervasive computing became apparent to researchers from the early 1990s.

Pervasive computing is the trend towards increasingly connected computing devices in the environment, a trend being brought about by a convergence of advanced electronic - and particularly, wireless technologies and the Internet. It is the idea of integrating computers seamlessly into the world such that people cease to become aware of it. Tremendous developments in such technologies as wireless communications and networking, mobile computing and handheld devices, embedded systems, and the like have led to the evolution of pervasive computing platforms. Two distinct earlier steps in this evolution are distributed systems and mobile computing. Distributed systems emerged at the intersection of personal computers and local area networks. It essentially is parallel computing using multiple independent computers communicating over a network to accomplish a common objective or task. Mobile Computing is distributed computing with mobile devices Thus, the design principles of distributed computing apply with constraints introduced due to mobility.

Users in pervasive computing environments can be mobile and have computing sessions distributed over a range of devices. The infrastructure's role with respect to users should be to maintain knowledge of their context and to manage tasks related to their mobility.

The shrinking size and increasing density of next generation wireless devices imply reduced battery capacities, meaning that emerging wireless systems must be more energy-efficient than ever before.

1.1 Motivation

The motivation of this thesis stems from the fact that power management is one of the most challenging problems in mobile systems. The fraction of the time a wireless device remains usable to the user is constrained by limited battery capacity and the user may have to plug in his device to recharge the battery regularly, which hampers the use of wireless devices. Advances in battery technology alone cannot deal with this problem. It is very important that the higher levels of the system are actively involved to ensure that the available energy is conserved and utilized in the most efficient way.

Also, today's systems are poor at capturing and exploiting context information and in adapting to changes in context. A pervasive computing system must be cognizant of its user's state and surroundings, and must modify its behavior based on this information. A pervasive computing environment is highly dynamic and the contexts in the environment change rapidly as users enter and leave the network. It is important that the devices are aware of the changing context and adapt to these changes accordingly. The need to make mobile devices smaller, lighter and have longer battery life may require that their computing capabilities be compromised. But meeting the ever-growing needs of mobile users may require high computing capabilities in lightweight devices. Reconciling these contradicting needs can be difficult. Cyber foraging provides an effective solution to this problem. The idea of cyber foraging is to dynamically augment the resources of a wireless mobile device by exploiting the resources of a nearby-wired infrastructure. Task migration also provides an effective solution to the problem of power management, but is dependent on environment conditions and other system and user contexts.

Most systems today, do not adapt to dynamically changing context data to make decisions on task migration. Thus, they are not able to fully exploit the benefits of remote execution.

1.2 Contribution and Scope

This thesis work investigates the criteria necessary to make intelligent choices on behalf of human users based on user preferences and system conditions. It addresses the need to automate performance enhancement techniques.

The contribution of this thesis includes the development of a middleware service framework, ConAEC (Context Aware Energy Conservation), deployed over a network of machines that exploit remote resources. The framework adopts a context-aware approach to make decisions on task migration to maximize energy savings on the local device. The developed framework is novel, in that it consists of intelligent software agents that are capable of:

- capturing and storing context information
- processing and analyzing this information
- adapting to dynamically changing contexts
- automatically making decisions and performing tasks on behalf of the user
- performing these activities in a way that is transparent to the user.

Most systems today do not use context information, do not consider the network environment context while making a decision to offload the task, or use static context information like size of the task to make an offload decision.

The framework proposed in this thesis work combines concepts from cyber foraging and context awareness to enable an adaptive context aware energy conservation scheme that uses cyber foraging to address the problem of power management.

1.3 Organization

The rest of the thesis is organized as follows. Chapter 2 gives a background of the related work in this area and covers pervasive computing, energy conservation, cyber foraging and context awareness. In Chapter 3, we discuss context aware energy conservation, energy conservation through cyber foraging, and the proposed scheme: context aware energy conservation using cyber foraging. In Chapter 4, we present the system model for the proposed middleware framework. Chapter 5 contains the implementation details and a discussion of the results. Finally, Chapter 6 draws conclusions and discusses future work.

CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

The vision of pervasive computing and ubiquitous computing was set forth in the early 90's. Mark Weiser, known as the 'father of ubiquitous computing', described a vision that consisted of environments saturated with computing and communication capability, yet gracefully integrated with human users. The goal is to create a system that is pervasively and unobtrusively embedded in the environment, completely connected, intuitive, effortlessly portable, and constantly available.

2.1 Pervasive Computing

Pervasive computing is an emerging field of research that brings in revolutionary paradigms for computing models in the 21st century. It is the trend towards increasingly connected computing devices in the environment, a trend being brought about by a convergence of advanced electronic - and particularly, wireless technologies and the Internet. It is the idea of integrating computers seamlessly into the world such that they, as Mark Weiser put it, 'weave themselves into the fabric of everyday life until they are indistinguishable from it'. According to Liu et al. [15] a pervasive computing system should be:

- Pervasive: be everywhere, with every portal reaching into the same information base.
- Embedded: live in our world, sensing and affecting it.
- Nomadic: allow users and computations to move around freely, according to their needs.
- Adaptable: provide flexibility and spontaneity, in response to changes in user requirements and operating conditions.

- Powerful, yet efficient: free itself from constraints imposed by bounded hardware resources, addressing instead system constraints imposed by user demands and available power or communication bandwidth.
- Intentional: enable people to name services and software objects by intent.
- Eternal: never shut down or reboot; it must be available all the time.

Although new technologies are emerging, the most crucial objective is not, necessarily, to develop new technologies but is largely focused on finding ways to integrate existing technologies with a wireless infrastructure.

2.1.1 History

The late Mark Weiser wrote what are considered some of the seminal papers in Ubiquitous Computing beginning in 1988. Weiser was influenced in a small way by the dystopian Philip K. Dick novel Ubik, which envisioned a future in which everything from doorknobs to toilet-paper holders, were intelligent and connected. Currently, the art is not as mature as Weiser hoped, but a considerable amount of development is taking place. The history of pervasive computing spans three decades while research in pervasive computing is very much ongoing. There are three key individuals that are responsible for shaping the concepts of pervasive computing.

Mark Weiser is considered to be the father of Ubiquitous computing. He wrote what are considered some of the seminal papers in Ubiquitous Computing beginning in 1988. Weiser's research paper [2] described computing as a progression of three waves of human-computer interaction. The first wave was mainframes in which one machine was shared by many human users. The second wave described today's common personal computing. In the second wave one personal computer is used by one human user. Weiser described the third wave as ubiquitous computing, in which one person will be shared by many computers.

Terry Allen Winograd is notable for his prediction of the future of computer science. In his article, “From Computing Machinery to Interface Design” [34] Winograd describes three directions of change for Computer Science. They are computation to communication, machinery to habit, and aliens to agents. He describes how data computing machines and applications have evolved to become data communicating applications. The second direction of change noted by Winograd is from machinery to habit. There is a shift in computer science from how to give instructions to computing machinery to how to perform high-level operations. There also is a shift from what operating system and software is running on a machine to what experience web enabled applications give to a user.

Mahadev Satyanarayanan is most notable for his 2001 article, “Pervasive Computing: Vision and Challenges” [3]. He identifies four new research areas brought about by pervasive computing; effective use of smart spaces, invisibility, localized scalability, and masking uneven conditioning. By embedding computing infrastructure in building infrastructure, a smart space brings together two worlds that have been disjoint until now. The fusion of these worlds enables sensing and control of one world by the other. Smartness may also extend to individual objects, whether located in a smart space or not. Satyanarayanan points out that the ideal expressed by Weiser is a complete disappearance of pervasive computing technology from a user’s consciousness. In practice, a reasonable approximation to this ideal is minimal user distraction. If a pervasive computing environment continuously meets user expectations and rarely presents him with surprises, it allows him to interact almost at a subconscious level. As smart spaces grow in sophistication, the intensity of interaction between a user’s personal computing space and his surroundings increases. This has severe bandwidth, energy and distraction implications for a wireless mobile user. The fourth thrust is the development of techniques for masking uneven conditioning of environments. The rate of penetration of pervasive computing technology into the infrastructure will vary considerably depending on many non-technical factors such as organizational structure, economics and business models. He also predicts a shift toward research areas outside mainstream computer science.

2.1.2 Related Fields

Tremendous developments in such technologies as wireless communications and networking, mobile computing and handheld devices, embedded systems, and the like have led to the evolution of pervasive computing platforms. Pervasive computing represents an evolutionary step in this line of work. Two distinct earlier steps in this evolution are distributed systems and mobile computing.

2.1.2.1 Distributed computing

Distributed systems emerged at the intersection of personal computers and local area networks. It essentially is parallel computing using multiple independent computers communicating over a network to accomplish a common objective or task. The type of hardware, programming languages, operating systems and other resources may vary drastically. Distributed computing spans many areas such as,

- remote communication, including protocol layering, remote procedure call, the use of timeouts, and the use of end-to-end arguments in place of functionality.
- fault tolerance, including atomic transactions, distributed and nested transactions, and two-phase commit
- high availability, including optimistic and pessimistic replica control, mirrored execution, and optimistic recovery.
- remote information access, including caching, function shipping, distributed file systems, and distributed databases.
- security, including encryption-based mutual authentication and privacy.

This body of knowledge is foundational to pervasive computing [2].

2.1.2.2 Mobile computing

Another important concept for pervasive computing is mobile computing. Mobile Computing can be defined as distributed computing with devices. Thus, the design principles of distributed computing apply with constraints introduced due to mobility. The key challenges include unpredictable variation in network quality, lowered trust and robustness, limitations on local resources imposed by weight and size constraints, and concern for battery power consumption of mobile elements.

Mobile computing is still a very active and evolving field of research and includes:

- Mobile networking, including Mobile IP, ad hoc protocols, and techniques for improving TCP performance in wireless networks.
- Mobile information access, including disconnected operation, bandwidth-adaptive file access, and selective control of data consistency.
- Support for adaptive applications, including transcoding by proxies and adaptive resource management.
- System-level energy saving techniques, such as energy aware adaptation, variable-speed processor scheduling, and energy-sensitive memory management.
- Location sensitivity, including location sensing and location-aware system behavior [2].

2.1.2.3 Missing Capabilities

Satyanarayanan [2] identifies two main capabilities, missing from distributed and mobile computing, that enable pervasive computing. The first is pro-activity which involves a system's ability to foresee a user's need and the system's capability to act upon that fore knowledge. The second is self-tuning or automatically adjusting to dynamically changing environmental conditions.

2.1.3 Challenges in Pervasive Computing

Pervasive computing provides a wealth of ubiquitous services and applications that allow users, machines, data, applications, and physical spaces to interact seamlessly with one another. Some of the technical problems in pervasive computing correspond to problems already identified and studied earlier in the evolution. In some of those cases, existing solutions apply directly; in other cases, the demands of pervasive computing are sufficiently different that new solutions have to be sought.

A pervasive computing environment may consist of a wide variety of devices such as wired devices, wireless devices, handheld devices, sensors and so on. Each of these devices has different requirements and capabilities and hence it is important that these heterogeneous devices communicate with each other efficiently with minimum user interaction. The practical realization of pervasive computing is hindered by a number of challenges. The software components of a pervasive computing system would need to support application requirements such as context awareness, dynamic adaptation, mobility and distribution, interoperability, rapid development and deployment of software components, providing resource discovery services and scalability.

2.1.3.1 Heterogeneity

One major challenge is that posed by devices operating in the environment: heterogeneity in the devices and the problems caused by device mobility. Heterogeneous devices will be required to interact seamlessly, despite wide differences in hardware and software capabilities.

2.1.3.2 Mobility

Mobility introduces problems such as the maintenance of connections as devices move between areas of differing network connectivity, and the handling of network disconnections.

2.1.3.3 Scalability

As smart spaces grow in sophistication, the intensity of interactions between a user's personal computing space and its surroundings increases. This has severe bandwidth, energy and distraction implications for a wireless mobile user. Scalability, in the broadest sense, is thus a critical problem in pervasive computing. Like the inverse square laws of nature, good system design has to achieve scalability by severely reducing interactions between distant entities.

2.1.3.4 Consistency

Users in pervasive computing environments can be mobile and have computing sessions distributed over a range of devices. The infrastructure's role with respect to users should be to maintain knowledge of their context and to manage tasks related to their mobility.

2.1.3.5 Security and Privacy

As a pervasive computing system becomes more knowledgeable about the user's movements, behavior and habits, it becomes more important that the use of this information is strictly controlled to prevent privacy and security threats.

2.1.4 Ongoing Work

There are also several research groups actively working to making pervasive computing environments a reality.

2.1.4.1 Aura

Aura [35] is a Carnegie Mellon University project that studies distraction free ubiquitous computing. Its goal is to create a halo of computing and information services that persist regardless of locality. The research group's approach is to redesign today's systems to decrease user distraction. Their current research areas include task driven computing, energy-aware adaptation, intelligent networking, resource opportunism, speech recognition, user interfaces, and nomadic data access.

2.1.4.2 Oxygen

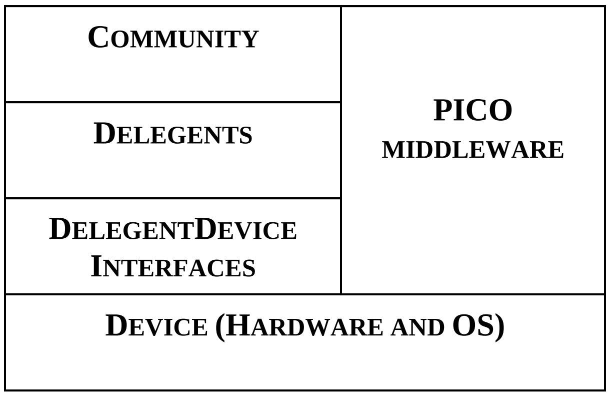
The mission of project Oxygen [36] is pervasive, human-centered computing. Its goal is to bring abundant computation and communication naturally in the lives of people. The research group's approach is to use specific user and system technologies to address human needs, perform automated tasks, and allow humans to interact with the system as if interacting with another human being. Their research areas are computer speech and vision, knowledge access, automation, and collaboration.

2.1.4.3 The Endeavour Expedition

The Endeavour Expedition [37] is a University of California at Berkeley pervasive computing project. Its mission is to chart the fluid information utility, or sea of information. The project's goal is to enhance human understanding through the use of information technology. The research group envisions a sea of information that needs to be mapped or charted, and made fluidly available. Their approach is to make interaction with information, devices, and other people more convenient. Their research areas are dynamic adaptation, self-organization, and personalization.
use sensors and resource status data to maintain a model of the physical world. Their research areas are location sensing, context awareness, user interfaces, and resource management.

2.1.4.4 PICO

The University of Texas at Arlington's pervasive computing research is on going with the Pervasive Information Community Organization (PICO) project [38]. The PICO project's mission is to create a simple, unique, and versatile middleware framework for pervasive computing. The research group's goal is to create mission oriented dynamic communities of software entities that collaborate to automatically perform tasks on behalf of users and devices. Their research areas include pervasive information acquisition and dissemination, context and location aware computing, quality of service, and security. The research group is investigating the use of middleware services to create Internet-enabled pervasive computing applications in heterogeneous networks.



2.1: PICO's layered structure

2.2 Energy Consumption

The shrinking size and increasing density of next generation wireless devices imply reduced battery capacities, meaning that emerging wireless systems must be more energy-efficient than ever before. The fraction of the time a wireless device remains usable to the user is limited and the user may have to plug in his device to recharge the battery regularly, which hampers the use of wireless devices.

Also, studies show that battery consumption of the wireless communication devices like a network interface card can account for over 50% of the total system power on these devices.

The energy capacity of batteries has doubled roughly every 35 years. While this trend has somewhat accelerated in recent years due to the needs of portable electronic devices, the rate of improvement is still fairly slow.

The development in battery technology has not kept up with the development of wireless technologies and hence we cannot expect any major progresses in battery capacity in the near future. Hence, it is very important that the higher levels of the system are actively involved to ensure that the available energy is conserved and utilized in the most efficient way.

2.2.1 Sources of power consumption

User tasks: The applications that the user is running on his/her mobile device consume energy. Here we take in to account only those tasks that are run locally on the device and require no communication through network interfaces. It involves the usage of the CPU, main memory, disk, I/O and other components.

System Tasks: These include tasks that are performed by the operating system as part of its routine.

Display: The display is considered to be the Achilles heel of power management as the entire display has to be backlit. This contributes a factor of 25 %– 35 % to the energy consumption of wireless devices like a PDA, cell phone or laptop.

Network Related: Sources of power consumption in relation to the network protocol stack can be classified into two types: communication related and computation related. Communication related consumption refers to the usage of the transceiver at the source, intermediate (in the case of ad hoc networks), and destination nodes. The transmitter is used for sending control, route request and response, as well as data packets originating at or routed through the transmitting node. The receiver is used to receive data and control packets – some of which are destined for the receiving node and some of which is forwarded. Communication energy is, among others, dictated by the signal-to-noise ratio (SNR) requirements and the radio cell diameter. A typical mobile device exists in three modes: transmit, receive and standby. Most of the power is consumed in the transmit mode and least in the standby mode.

Computation considerations on the other hand are concerned with protocol processing aspects. It mainly involves the usage of the CPU and main memory and, to a minute extent, the disk or other components. Also, data compression techniques, which reduces packet length (and hence energy usage), may result in increased computation. Computation energy is a function of the hardware and software used for the tasks such as compression and forward error correction. There exists a potential tradeoff between computation and communication costs. Techniques that may strive to achieve lower communication costs may results in higher computation needs, and vice-versa. Therefore a balance between the two goals must be struck for energy efficiency to be achieved.

2.2.2 Energy Conservation Techniques

Energy conservation techniques can be broadly classified into hardware techniques, communication protocols related techniques and application level techniques. The following sections discuss these approaches in detail and the related work in each field.

2.2.2.1 Hardware Power Management

Operating system: The operating system plays a key role in energy management [39]. It controls all the devices, so it must decide what and when to shut down. If it shuts down a device and that device is needed again quickly, there may be an annoying delay while it is restarted. On the other hand, if it waits too long to shut down a device, energy is wasted for nothing.
Hard Disk: It takes substantial energy to keep it spinning at high speed, even if there are no accesses. Many computers, especially laptops, spin the disk down after a certain number of minutes of activity. When it is next needed, it is spun up again. Unfortunately, a stopped disk is hibernating rather than sleeping because it takes quite a few seconds to spin it up again, which causes noticeable delays for the user. In addition, restarting the disk consumes considerable extra energy. If a good prediction could be made (e.g., based on past access patterns), the operating system could make good shutdown predictions and save energy. In practice, most systems are conservative and only spin down the disk after a few minutes of inactivity. Another way to save disk energy is to have a substantial disk cache in RAM. Another way to avoid unnecessary disk starts is for the operating system to keep running programs informed about the disk state by sending it messages or signals [39].

CPU: The CPU can also be managed to save energy. A laptop CPU can be put to sleep in software, reducing power usage to almost zero. The only thing it can do in this state is wake up when an interrupt occurs. Therefore, whenever the CPU goes idle, either waiting for I/O or because there is no work to do, it goes to sleep. On many computers, there is a relationship between CPU voltage, clock cycle, and power usage. The CPU voltage can often be reduced in software, which saves energy but also reduces the clock cycle (approximately linearly). Since power consumed is proportional to the square of the voltage, cutting the voltage in half makes the CPU about half as fast but at 1/4 the power [39].

Memory: Two possible options exist for saving energy with the memory. First, the cache can be flushed and then switched off. It can always be reloaded from main memory with no loss of information. The reload can be done dynamically and quickly, so turning off the cache is entering a sleep state. A more drastic option is to write the contents of main memory to the disk, then switch off the main memory itself. This approach is hibernation, since virtually all power can be cut to memory at the expense of a substantial reload time, especially if the disk is off too [39].

Display: Many operating systems attempt to save energy here by shutting down the display when there has been no activity for some number of minutes. Often the user can decide what the shutdown interval is, pushing the trade-off between frequent blanking of the screen and using the battery up quickly back to the user (who probably really does not want it). Turning off the display is a sleep state because it can be regenerated (from the video RAM) almost instantaneously when any key is struck or the pointing device is moved. One possible improvement was proposed by Flinn and Satyanarayanan [12]. They suggested having the display consist of some number of zones that can be independently powered up or down.

2.2.2.2 Energy Efficient communication protocols

Communication is a significant consumer of power and hence, considerable research has been devoted to low-power design of the entire network protocol stack of wireless networks in an effort to enhance energy efficiency [16].

Application programs using the network do not interact directly with the network hardware. Instead, an application interacts with the protocol software. The notion of protocol layering provides a conceptual basis for understanding how a complex set of protocols work together with the hardware to provide a powerful communication system. The application and services layer occupies the top of the stack followed by the operating system/middleware, transport, network, data link, and physical layers. The problems inherent to the wireless channel and issues related to mobility challenge the design of the protocol stack adopted for wireless networks. In addition, networking protocols need to be designed with energy efficiency in mind. A lot of research has been done in all layers of the protocol stack to design energy efficient protocols. We will review a few techniques at each layer that have been proposed.

MAC Layer: The MAC (Media Access Control) layer is a sublayer of the data link layer which is responsible for providing reliability to upper layers for the point-to-point connections established by the physical layer. The MAC sublayer interfaces with the physical layer and is represented by protocols that define how the shared wireless channels are to be allocated among a number of mobiles. Collisions should be eliminated as much as possible within the MAC layer since they result in retransmissions. Retransmissions lead to unnecessary power consumption However, it may not be possible to fully eliminate collisions and retransmissions in a wireless mobile network. In this case, using a small packet size for registration and bandwidth request may reduce energy consumption. The EC-MAC protocol is one example that avoids collisions during reservation and data packet transmission [16].
In a typical broadcast environment, the receiver remains on at all times which results in significant power consumption. The mobile radio receives all packets, and forwards only the packets destined for the receiving mobile. This is the default mechanism used in the IEEE 802.11 wireless protocol which consumes a lot of energy. One solution is to broadcast a schedule that contains data transmission starting times for each mobile [16]. This enables the mobiles to switch to standby mode until the receive start time.

Link Layer: At the link layer, transmissions may be avoided when channel conditions are poor. Also, error control schemes that combine automatic repeat request (ARQ) and forward error correction (FEC) mechanisms may be used to conserve power (i.e., tradeoff retransmissions with ARQ versus longer packets with FEC) [16].

Network Layer: Energy efficient routing protocols may be achieved by establishing routes that ensure that all nodes equally deplete their battery power. This helps balance the amount of traffic carried by each node. A related mechanism is to avoid routing through nodes with lower battery power, but this requires a mechanism for dissemination of node battery power. Also, the periodicity of routing updates can be reduced to conserve energy, but may result in inefficient routes when user mobility is high. Another method to improve energy performance is to take advantage of the broadcast nature of the network for broadcast and multicast traffic, the topology of the network is controlled by varying the transmit power of the nodes, and the topology is generated to satisfy certain network properties [16].

Transport Layer: The transport layer provides a reliable end-to-end data delivery service to applications running at the end points of a network. TCP and similar transport protocols resort to a larger number of retransmissions and frequently invoke congestion control measures. Increased retransmissions unnecessarily consume battery energy and limited bandwidth. Various schemes have been proposed to alleviate the effects of non congestion-related losses on TCP performance over networks with wireless links. The energy consumption of Tahoe, Reno, and New Reno versions of TCP is analyzed and results of the study demonstrate that error correlation significantly affects the energy performance of TCP and that congestion control algorithms of TCP actually allow for greater energy savings by backing off and waiting during error bursts [16]. A modified version of TCP, referred to as TCP Probing is proposed, in which data transmission is suspended and a probe cycle is initiated when a data segment is delayed or lost, rather than immediately invoking congestion control [16].

2.2.2.3 Application Level Power Management

The application layer in a wireless system is responsible for such things as partitioning of tasks between the fixed and mobile hosts, audio and video source encoding/decoding, and context adaptation in a mobile environment. Energy efficiency at the application layer is becoming an important area of research. Some of the research being conducted at the application layer with respect to power conservation include:

Load partitioning: Due to power and bandwidth constraints, applications may be selectively partitioned, for example, between the mobile and base station [16]. Thus, most of the power intensive computations of an application are executed at the base station, and the mobile host plays the role of an intelligent terminal for displaying and acquiring multimedia data [16].

Proxies: Another means of managing energy and bandwidth for applications on mobile clients is to use proxies. Proxies are middleware that automatically adapt the applications to changes in battery power and bandwidth.

Databases: Impact of power efficiency on database systems is considered by some researchers. For example, energy efficiency in database design by minimizing power consumed per transaction through embedded indexing has been addressed in [16].

Video processing: Multimedia processing and transmission require considerable battery power as well as network bandwidth. This is especially true for video processing and transmission. However, reducing the effective bit rate of video transmissions allows lightweight video encoding and decoding techniques to be utilized thereby reducing power consumption.

Degradation mechanisms: can be used where the programs are instructed to use less energy, even if this means providing a poorer user experience to the user. Typically, this information is passed on when the battery charge is below some threshold. It is then up to the programs to decide between degrading performance to lengthen battery life or to maintain performance and risk running out of energy. Flinn and Satyanarayanan [12] provide four examples of how degraded performance can save energy. Information is presented to the user in various forms. When no degradation is present, the best possible information is presented. When degradation is present, the fidelity (accuracy) of the information presented to the user is worse than what it could have been. The first program measured was a video player which plays 30 frames/sec in full resolution and color. One form of degradation is to abandon the color information and display the video in black and white. Another form of degradation is to reduce the frame rate, which leads to flicker and gives the movie a jerky quality. Similar experiments were conducted with a speech recognizer, map viewer that fetched the map over the radio link and transmission of JPEG images to a Web browser. Some degradation mechanism was applied to each application when power was low and experiments showed that by accepting some quality degradation, the user can run longer on a given battery.

2.3 Resource Exploitation or Cyber Foraging

The need to make mobile devices smaller, lighter and have longer battery life may require that their computing capabilities be compromised. But meeting the ever-growing needs of mobile users may require high computing capabilities in lightweight devices. Reconciling these contradicting needs can be difficult.

Cyber foraging provides an effective solution to this problem. The idea of cyber foraging is to dynamically augment the resources of a wireless mobile device by exploiting the resources of a nearby-wired infrastructure [3]. When an intensive computation accessing a large volume of data has to be performed, the mobile device ships the entire workload to the wired computer. This computer performs the required processing and ships the results back to the mobile device. The remote machine can also be a server.

2.3.1 Challenges

Cyber foraging opens a number of challenges [3]:

- How does the wireless device detect the presence of a wired infrastructure? Since the wireless devices are highly mobile, they would have to constantly detect the presence of new infrastructure and absence of older ones. What type of service discovery mechanism would be best to use.
- How does the mobile device become aware of the capabilities of the wired computer?
- Security issues: What is the level of security needed and how is access granted to users of the services.
- What is the level of trust between the wired device and mobile devices?
- How is load balancing done on the wired devices?
- What are the implications of scalability?
- How can this be done so that it as transparent to the user as possible?

The performance increase experienced by the use of resource exploitation is very important. Noticeable improvements in energy consumption or response time are necessary to substantiate the use of resource exploitation. A consideration in this area is the size and complexity of the middleware system.

2.3.2 Related Work

Two related approaches based on [7] are Cyber-foraging (Spectra) and Cyber-foraging (Lightweight). Cyber-foraging uses the concept of surrogates which are unused machines that mobile devices dynamically locate and control to store data or run applications. Spectra [7] is designed for data staging, which allows a device user to cache data at a nearby surrogate while a user is in a locality, and remote execution through remote procedure calls (RPC). Spectra uses a heavyweight middleware component and a centralized server to allow a device access to surrogates. Cyber-foraging (Lightweight) [7] removes some of the heavyweight middleware components to decrease energy consumption. Each surrogate creates virtual servers with quality guarantees upon receiving a client request. The client can then bypass the surrogate and customize and utilize its virtual server. This approach still uses a centralized resource manager and has added virtual server allocation and configuration overhead. Portability and interoperability are other concerns that need to be addressed. The goal is to maximize energy savings on the low power device so system independent components such as Java can be used to provide interoperability and limited functionality when necessary.

Rudenko et al. [8] propose a remote processing framework for automatically migrating tasks from a portable computer over a wireless network to a server and migrating the results back. The design of the Remote Processing Framework (RPF) is based on a client-agent-server paradigm. On the client's side the framework automatically decides whether to run a process remotely or locally, moves necessary portions of the user's tile system to the remote machine, runs the process locally or remotely, and handles the database of processes. On the server side the package performs the operations the user requested and reliably communicates the results back.

Rollins et al. [9] investigate the possibility of managing the user's data across a collection of devices that the user carries. It is assumed that the user carries a number of small devices like a laptop, PDA, cell phone and say, a digital watch. Cyber foraging can also be applied in this environment where the workload is migrated to another wireless device of the user based on the capabilities of the device.

In addition to issues above we have to deal with the issues of choosing which device to migrate the task to, the communication used between the devices and the problem of heterogeneity of these devices.

2.4 Context Awareness

According to DEY et al. [20] context can be defined as: any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

The important aspects of context are: where you are, who you are with, and what resources are nearby. Examples of context are numerous and include location, temperature, network bandwidth, remaining energy, date and time etc. Application designers will need to determine what context-aware behaviors to support in their applications and hence decide what contexts they are interested in.

2.4.1 Context-Aware Systems

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

2.4.2 User Intent

A proactive pervasive computing system must capture the user intent. Systems today are poor at exploiting user intent and have no idea what the user is trying to do. Though attempts are being made, most are more often annoying than useful. In designing systems that capture user intent, one has to decide how the user intent is going to be determined. Is it going to be inferred or does the user have to provide it, how the user intent would be represented internally, how accurate is this information and will this effort put a burden on the user and on the system's resources (CPU, memory and energy).

2.4.3 Context awareness

A pervasive computing system must be cognizant of its user's state and surroundings, and must modify its behavior based on this information. A user's context can consist of his physical location, physiological state (e.g., body temperature and heart rate), emotional state (e.g., angry, distraught, or calm), personal history, daily behavioral patterns, and so on. Schmandt et al. [22] develop a location aware system that detects the user's location using GPS. Schilt et al [23] discuss an auto-receptionist system that detects user's location and forwards calls to the user. Contexts such as weather, body temperature, light etc. require the use of sensor nodes in the environment.

A system can modify its behavior based on its own context information. A system's context could consist of the applications currently running on the system, the current energy level of the battery, the type of device being used and the internal resources like CPU, memory available etc. The context could also include external resources available like network bandwidth, nearby wired infrastructure, services available etc.
Zimmer [21] identifies the important attributes of "context data" as:
Relevance: depends on the age and the distance between where the context data was generated and where it is being used.
Reliability: usually depends on the reliability of the input data used to determine the context
Context history: context is usually dynamic and context history can be viewed as a representation of its dynamic character.
Validity: The value of this attribute can depend on the relevance and reliability of the data and its history.

2.4.4 Adaptation

A pervasive computing environment is highly dynamic and the contexts in the environment change rapidly as users enter and leave the network. Hence most applications use changes in contexts to trigger events. Thus, it is important that the devices are aware of the changing context and adapt to these changes accordingly. This leads to the challenge of how the devices would detect these changes and secondly how they would adapt to these changes after they are detected. This is called the adaptation strategy.

It is interesting to see if this context information can be used in some way to reduce energy consumption of the device. However, learning and storing context information, processing and analyzing this information and using this information to adapt accordingly would require system resources (CPU, memory and energy). Hence, it is important to decide if the cost is worth the benefits.

CHAPTER 3

PROPOSED SCHEME: ConAEC

The scheme proposed in this thesis is a context aware energy conservation scheme that uses cyber foraging. In this section, we first address the concepts of context aware energy conservation, cyber foraging for energy conservation and the related work in these fields. We then describe the proposed scheme, the energy model and the algorithms.

3.1 Context Aware Energy Conservation

Energy is a vital resource for mobile devices. The development in battery technology has not kept up with the development of wireless technologies and hence we cannot expect any major progresses in battery capacity in the near future. Hence, it is very important that the higher levels of the system are actively involved to ensure that the available energy is conserved and utilized in the most efficient way.

One of the major challenges of pervasive computing is to exploit the changing environment with a new class of applications that are aware of the context in which they run. A context aware system can examine the computing environment and react to changes to the environment.

Context information can be used to reduce energy consumption of the device. Contexts such as location, network connectivity, date and time, remaining energy, local and remote resources available can be used by the higher levels of the system in efficient utilization of energy. The system should also be able to adapt to changes in these contexts. However, learning and storing context information, processing and analyzing this information and using this information to adapt accordingly would require system resources (CPU, memory and energy). Hence, it is important to decide if the cost is worth the benefits and keep the cost of this overhead low.

3.1.1 Related Work

There has been limited research in the area of context aware energy conservation where devices are context aware and modify their operation according to changes in contexts or the environment around them. The behavior that is desirable in such wireless devices is adaptation where the system is able to adapt to changes in the environment.

Flinn and Satyanarayanan [12] explore how applications can dynamically modify their behavior to conserve energy. The context used in their work is the remaining energy on the resource-limited device. Based on this context, information is presented to the user in various forms. When energy is plentiful, application behavior is biased toward a good user experience; when it is scarce, the behavior is biased toward energy conservation by degrading the quality of the applications running on the device. The first program measured was a video player which plays 30 frames/sec in full resolution and color. One form of degradation is to abandon the color information and display the video in black and white. Another form of degradation is to reduce the frame rate, which leads to flicker and gives the movie a jerky quality. Similar experiments were conducted with a speech recognizer, map viewer that fetched the map over the radio link, and transmission of JPEG images to a Web browser. Some degradation mechanism was applied to each application when power was low and experiments showed that by accepting some quality degradation, the user can run the device for a longer period on a given battery.

Rollins et al. [9] focus on data management and power limitations and investigate the benefit of using power aware schemes to manage data among a number of devices to prolong data availability to the user. The energy level of each device is monitored and the workload is migrated from devices that are in danger of dying, to other devices that can handle the workload with their remaining energy. Here again, the context used is the remaining energy on each of the devices.

Chong et al. [10] propose a context aware approach to conserve energy in wireless sensor networks. They present a framework for supporting the use of context to trigger power saving functionalities in sensors. The Context Discovery component serves to discover useful contexts from sensor data. In the context-trigger engine, the discovered context is used as a trigger. Context input is discovered and used to analyze data streams and if it matches the triggering information, an output reaction is triggered by the engine.

3.1.2 Context Aware Energy Conservation Techniques

This section discusses the different context aware energy techniques.

3.1.2.1 Hardware Power Management for context aware energy conservation

The operating system plays a key role in energy management, controlling the devices and decides what and when to shut down. The operating system uses some context information such as time of inactivity to make this decision. Other context information such as the user's location, remaining energy, information derived from user's calendar and date and time can be used to put the device in low power mode.

3.1.2.2 Communication related context aware energy conservation

As described in section 2.2, communication is a significant consumer of power. The energy associated with communication can be reduced by monitoring contexts such as available network bandwidth, idle times of the network interface card and the network environment. For example, if system is aware that the network interface is in idle mode and will be idle for some period of time, the system may then turn the NI off and put it into sleep mode to conserve energy; or if the system is aware of higher power devices in the network it can offload its tasks to the remote device.

3.1.2.3 Application-specific context aware energy conservation

As discussed in section 2.2, the applications that the user is running on his/her mobile device consume energy. The application can adapt to changes in contexts such as location, remaining energy or date and time. For e.g. the fidelity (quality) of the application can be changed based on energy supply or in case of e-mail application the display can be changed to a lower power consuming quality and resolution than in case of camera mode [12].

3.2 Cyber foraging and Energy Conservation

Cyber foraging can be adopted as an energy conservation technique. One of the major challenges of cyber foraging is determining the tasks that are worth migrating and the costs involved in the migration.

Firstly, we have to determine the transmission costs involved in the migration of the task. The network interface consumes energy in the sending, receiving and idle state. Secondly, we have to determine the set of tasks that can be migrated, based on the capabilities of the remote device and the type of application. Thirdly, we will have to compute the energy that will be consumed if the task would be executed locally on the device.

3.2.1 Determining the set of tasks that can be migrated

The set of tasks that can be migrated depends on the type of application (e.g. real-time or device-specific tasks) and the capabilities of the remote machine. The device may know the capabilities of the remote device or the device can probe the remote device to determine its capabilities and available computing resources.

3.2.2 Measuring the energy spent in communication

This includes the energy spent in sending data, receiving data and the energy spent in idle mode. The energy consumed in idle mode is directly related to the time required for the task to execute on the remote machine. During this time the NI is idle and waiting for a response from the remote machine. Hence, the total execution time on the remote device is an important factor in determining if the benefits of migration are worth the cost. Also, the amount of data shipped to and from the device is directly related to the size of the task. Therefore, the size of the data transferred should be considered in cyber foraging.

3.2.3 Measuring the energy consumed by the local execution of a task

To make an offload decision we would have to compare the energy spent by the device to execute the task locally with the energy spent by NI in transferring the task to a remote machine. The most accurate way to measure the power consumption due to the execution of a task on a device would be to insert appropriate electronic instrumentation between the battery and the device. However this leads to many practical problems. Rudenko et al [8] propose the use of less direct methods. They suggest the use of the Advanced Power Metric of a laptop that reports the remaining energy as a percentage of the maximum. Flinn and Satyanarayanan [12] use the PowerScope energy profiler that can determine what fraction of the total energy consumed during a certain period of time is due to a specific process. These methods can be used to determine the energy drain of the local device if the task is executed locally, and this can be used to determine the trade-offs in task migration.

3.2.4 Related Work

Rudenko et. al [8] explore the idea of power saving and battery life extension through wireless remote processing of power-costly tasks. The authors have analyzed the trade-off between communication power and local processing cost. The tasks are moved to the remote machine before the task starts execution. The server would process the task and ship the results back to the portable device. In the meantime, the portable device can run other tasks or be idle. The first part of the experiments was run in a noiseless environment. All major sources of noise (like other laptops equipped with wireless cards) were isolated. Power consumption of local and remote execution was compared. For relatively small tasks, the cost of local execution was lesser than that of remote but for larger tasks savings of remote execution were as high as 45%. Further experiments were performed to determine the power saving characteristics of remote execution in a realistic noisy environment where the devices have to contend for the channel. The same set of experiments were conducted to measure the total energy and to measure the transmission energy. In a noisy environment there are more re-transmissions as a result of collisions and the time the client has to wait for a response increases. Hence, the energy savings through remote execution decreases. Results show that the increased cost of transmissions in a noisy medium is about 1.5 – 2% of the battery.

Li et al. [6] propose a partition scheme for computation offloading to save energy on handheld devices. In this work, a cost graph is constructed for a given application program based on profiling information on computation time and data sharing at the level of procedure calls. A partition scheme is then applied to statically divide the program into server and client tasks such that the energy consumed by the program is minimized.

3.3 Context Aware Energy Conservation with Cyber foraging

All schemes proposed thus far do not use context information, do not consider the network environment context while making a decision to offload the task, or use static context information like size of the task to make an offload decision.

The scheme proposed in this thesis is a context aware energy conservation scheme that uses cyber foraging. We have explored the combination of context awareness and cyber foraging and see if this could yield an effective method to deal with the problem of energy conservation.

A mobile environment is highly dynamic and this leads to a situation where there is unpredictable variation in the network resources available, the services available and the devices nearby. The user's and the devices' context also change dynamically with time. We would like to devise a scheme that takes the current context of the environment, the device and the user in making decisions on task migration.

For e.g. if available network bandwidth is low, the energy spent on communication may dominate the benefits of remote task execution. If the available bandwidth increases over time the system should be able to detect this change in context and adapt accordingly.

Our scheme uses the available network bandwidth or the achievable throughput in an end-to-end path as the context to make intelligent decisions on task migration. We examined the trade-off between communication power expenditures and the power cost of local processing using this context.

When a task is to be performed a decision is made to execute it locally or offload it to one of the available remote devices. Estimations of energy that will be consumed if the task is executed locally and remotely is made. For local execution, the activities that contribute to energy consumption primarily include CPU processing and disk accesses. For remote execution, energy is consumed by the network interface in transmission of the task request and data (if required), in reception of the results, and in the idle state while the device waits for the results to be shipped back. If the end-to-end throughput to the remote device is high, then, the energy consumed is less. Hence, we use this context of achievable throughput to each device to make a decision to offload, and the device to offload to.

We maintain a history of the achievable throughput to each device and use an average of the values to estimate what the current throughput to each device would be. Using these values we estimate the energy that will be spent in communication to each device.

A history of execution times for each task on the local and each of the remote devices is also maintained. These values are used to estimate the energy that will be spent to execute the task on the local device and the energy spent by the NI in the idle state if the task is offloaded. A decision is made to execute the task on the local device or on a remote device so as to minimize the energy consumption on the local device.

3.4 Energy Model

This section discusses the energy model used for estimations and decision making.

3.4.1 Decision making

A decision to offload a task is made if the estimated energy for local execution is greater than the estimated energy for remote execution by a value greater than the threshold value. The threshold value, E_{th}, is introduced to account for dynamically changing network environments. When network conditions are close to static, E_{th} approaches zero.

```
if [$E_{pl} > E_{po} + E_{th}$] then
        offload = true
else
        offload = false
```

where:

E_{pl} :	is the predicted energy consumption for local execution
E_{al} :	is the actual energy consumption for local execution
E_{po} :	is the predicted energy consumption for remote execution
E_{ao} :	is the actual energy consumption for remote execution
E_{th} :	is the error margin or threshold

3.4.2 Checking for correctness of the decision

After a decision is made and the task is executed, the actual energy consumed in completing the task is measured. A comparison between the actual and predicted values is made and the value of the threshold is modified appropriately if the decision was wrong.

The following section discusses when a decision is considered wrong and what actions are taken to ensure that the further decisions are made correctly.

A. When task is offloaded:

When the task is offloaded, if the difference between the actual and predicted values for energy is less than the threshold value, $\boldsymbol{E_{th}}$, then the decision is considered to be right. This is because the decision to offload is made considering a possible error margin equal to the threshold. Therefore, as long as the difference is less than this threshold value, the decision is right. If the actual energy consumed is greater than the predicted value, the decision is considered to be wrong. The value of the threshold, $\boldsymbol{E_{th}}$ is increased to avoid further wrong decisions.

```
if [ offload = true ] then
    if [ $E_{ao} - E_{po}$ < = $E_{th}$ ] then
        decision = right
        if [ $E_{ao} < E_{po}$ ] then
            decrease $E_{th}$
        endif

    else
        decision = wrong
        increase $E_{th}$
    endif
endif
```

B. When task is executed locally:

When the task is executed locally, if the actual energy consumed is less than or equal to the predicted offload energy plus the threshold, then the decision is considered to be right, otherwise, the decision is considered as wrong. The assumption is that the predicted offload energy is almost accurate and energy savings would have been higher if the task had been offloaded.

```
if [offload = false] then
    if [$E_{al} <= E_{po} + E_{th}$] then
        decision = right
    else
        decision = wrong
    endif
endif
```

3.4.3 Predicting Energy Consumption

This section describes the method adopted to estimate the energy consumed if a given task is executed locally and energy consumed if the task is offloaded. The basic principle used to estimate the energy is given by the equation:

$$\textbf{Energy} = \textbf{Power} * \textbf{Time}$$

A. When the task is executed locally:

The energy that will be consumed to execute the task locally is estimated as the product of the power dissipated by the device for performing calculations while doing no communication, and the time to execute the task locally on the device. The time to execute the task is estimated based on the past values of execution time for the same task. This method is described in next section.

$$E_{pl} = P_{proc} * t_{proc} \text{ , where,}$$

E_{pl} : is the predicted energy consumption for local execution

P_{proc}: power in watts the node consumes when performing computation and no communication

t_{proc}: time to execute the task locally

B. When the task is offloaded:

The energy that will be consumed to offload the task and receive the results back is estimated as the sum of the energy consumed to transmit the data needed to execute the task to the remote device, the energy consumed by the network interface in the idle state waiting for the results from the device and the energy consumed in the reception of the results. Each component is calculated as the product of the power and the time for each activity to complete [4].

$$E_{po} = (P_{tx}\, t_{tx}) + (P_{idle}\, t_{idle}) + (P_{rx}\, t_{rx}) \text{, where,}$$

E_{po} : is the predicted energy consumption for remote execution

P_{idle}: is the power in watts the node consumes when idle waiting for packets or ACKs

$\boldsymbol{P_{tx}}$: is the power in watts spent in processing and transmitting TCP segments and ACKs

$\boldsymbol{P_{rx}}$: is the power in watts spent in processing and receiving TCP segments and ACKs

$\boldsymbol{t_{idle}}$: idle time of the node waiting for packets or ACKs

$\boldsymbol{t_{tx}}$: time for processing and transmitting TCP segments and ACKs

$\boldsymbol{t_{rx}}$: time for processing and receiving TCP segments and ACKs

The method to estimate the time required for transmission, in idle state and reception is as follows:

The values of $\boldsymbol{t_{tx}}$ (time for processing and transmitting TCP segments and ACKs) and $\boldsymbol{t_{rx}}$ (time for processing and receiving TCP segments and ACKs) are estimated using the principle:

Time for transmission (reception) = Size of the data / Rate of the link

$$t_{tx} = \frac{Bsend}{r}$$

$$t_{rx} = \frac{Brecv}{r}$$

The value of $\boldsymbol{t_{idle}}$ (idle time of the node waiting for packets or ACKs) is estimated by first estimating the total time for sending and receiving the data using the value of the average throughput of the end-to-end path and the time required to execute the task on the remote device.

$$t_{total} = \frac{Bsend + Brecv}{\tau}$$

$$t_{idle} = t_{rproc} + (t_{total} - t_{tx} - t_{rx})$$

where:

$\boldsymbol{B_{send}}$: size of data to be transmitted

$\boldsymbol{B_{recv}}$: size of data to be received

$\boldsymbol{r}$: transmission rate of the link

τ: estimated achievable throughput

t_{total}: total time to send and receive the completed task

t_{rproc}: time taken by the remote machine to execute the task

3.4.4 *Estimating t_{rproc}, t_{proc} and τ*

We maintain a history of the last N measured values of these parameters (*param*) and estimate $param_{(N+1)}$ as:

$$param_{(N+1)} = \alpha * param_N + (1-\alpha) \frac{\sum_{i=1}^{N-1} param}{N-1}$$

3.5 Algorithms

A decision making algorithm is derived based on the energy model described above. This section gives an overview of this algorithm and the algorithm that updates the parameters when actual measurements of the throughput, execution time and energy consumed are made.

3.5.1 *Procedure to make an offload decision*

This algorithm makes an offload decision by first estimating the energy required to perform the task locally. It then estimates the energy required to offload the task to each of the available devices that provide this service. It then makes a decision based on these estimated values to minimize the energy consumption.

```
Notations:
task: the task to be executed
size: size of the task to be executed
Bsend: size of data to be transmitted to remote device
Brecv: size of data to be received from the remote device

Procedure offloadDecision
begin
      Epl := proc estimateLocalEnergy (task, size)
      set targetDevice as "local"

      get list of devices hDevices that provide this service

      set Epomin as infinite
      foreach (device in hDevices)
            Epo := proc estimateOffloadEnergy (task, size,
            Bsend,    Brecv)
            if (Epo < Epomin)
                  Epomin = Epo
                  targetDevice = device
            end if
      end foreach

      if (Epl > Epomin + Eth)
            decision = "offload"

      else
            decision = "local"

end
```

3.5.2 Procedure to estimate the energy for local execution

This algorithm estimates the energy required to execute the task locally based on the energy model described in the above section.

```
Notations:
Tproc: time to execute the task locally
size : size of the task to be executed
Epl: predicted energy for local execution
Pproc: power consumed when performing computation

Procedure estimateLocalEnergy
begin
      Tproc = proc estimateTime (size)
      Epl = Pproc * Tproc
      return Epl
end
```

3.5.3 *Procedure to estimate the energy for remote execution*

This algorithm estimates the energy required to execute the task remotely based on the energy model described in the above section.

```
Notations:
size: size of the task to be executed
LinTr: transmission rate of the wireless link
Epo: predicted energy for remote execution
Trproc: time to execute the task on the remote device
Tidle: idle time waiting for results
Ttx: time for processing and transmitting the data and ACKs
Trx: time for processing and receiving ACKs
bwSend: estimated throughput to send the data to remote device
bwRecv: estimated throughput to receive the data from remote
device
Pproc: power consumed when performing computation


Procedure estimateOffloadEnergy
begin
      Ttx = (Bsend + ACKData) / LinkTr
      Trx = (Brecv + ACKData) / LinkTr

      bwSend = proc estimateBWSend ()
      bwRecv = proc estimateBWRecv ()

      Ttotal = ( Bsend  / bwSend ) + ( Brecv / bwRecv )
      Trproc = estimateTime (size)
      Tidle = Trproc + (Ttotal - Ttx - Trx)

      Epo = ((Pidle * Tidle) + (Ptx * Ttx) + (Prx * Trx))
      return Epo
end
```

3.6 Mobility and Service Discovery

A pervasive environment is highly dynamic and devices may enter and leave the network at any time. Mobility introduces problems such as the maintenance of connections as devices move between areas of differing network connectivity, and the handling of network disconnections.

The framework must constantly keep track the set of devices that are available for remote processing. If a device leaves the network each of the other devices should be able to adapt to this change in the environment and update the information accordingly.
However, the process of service discovery is beyond the scope of this thesis work.

CHAPTER 4

SYSTEM MODEL

The framework uses a software engineering concept known as middleware. The term middleware describes a software layer that bridges the gap between operating systems, network components, and applications. The Internet2 Middleware Initiative [40] describes middleware as a modular or layered approach to providing communication, identification, authorization, authentication, security, and other services. It simplifies system development by isolating and hiding complex system components and their associated interfaces. The goals of middleware also include masking heterogeneity and providing a useful distributed programming model. Middleware services are defined by their interfaces and data formats.

4.1 Pervasive Information Community Organization (PICO)

The PICO framework consists of an interconnection of hardware devices called devices and autonomous, intelligent software agents called *delegents* (or intelligent delegates). PICO's objective is to provide "what we want, when we want, where we want, and how we want" types of services autonomously and continually [1].

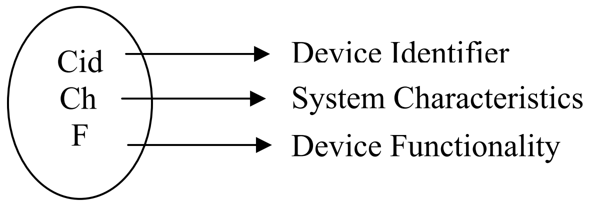
In a pervasive computing environment the hardware device capabilities and resources in the system vary. This problem of heterogeneity in the environment is addressed by the software agents which provide interoperability. The software agents are also proactive which allow them to automatically perform tasks on behalf of human and device entities.

4.1.1 Architecture of the PICO framework

Devices and delegents are PICO's basic building blocks. A delegent representing a user, application, or device can reside on another device. A community's delegents can reside on one or more devices. Community operations involving several delegents and the device resources make up the PICO architecture [1].

4.1.1.1 Devices

A device possesses one or more functionalities— such as see, hear, adapt, compute, communicate, learn, or process information.

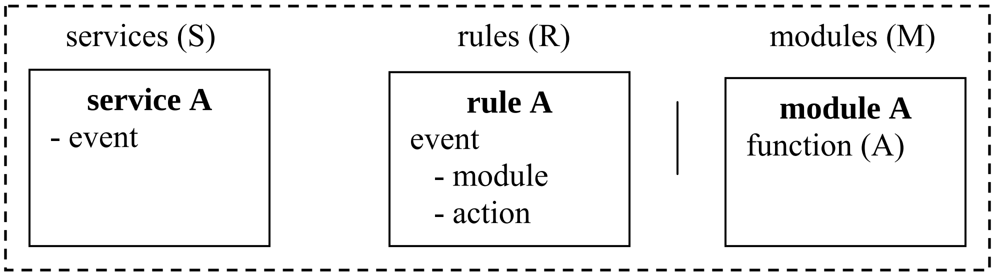


4.1: A Device Tuple

A device is described by three tuples, *C* = < *C*id, *C*h*, F*>, where *C*id is the device identifier, *C*h is the set of system characteristics, and *F* is the set of functionalities. For example, we describe an image processing device by *C*(*IP*) = <###, *C*h, *F*>, where ### is the identifier, *C*h = <operating system, processor type, memory, I/O type, battery, wireless transceiver>, and *F* = <image processing, communication> [1].

4.1.1.2 Delegents

A delegent works diligently on behalf of a device or user. Delegents are intelligent software agents that consist of a tuple set *D* = <*D*id, *F*d>, where *D*id is the delegent's identity and *F*d is its functional description. Functionally, a delegent can be represented by a three tuple, *F*d = <*M, R, S*>, where *M* is the set of program modules, *R* is the set of rules for delegent behavior, and S is the delegent's goal or mission [1].



4.2: A Delegent Tuple

The tuples (*delegent* = <*M, R, S*>) are used to map service requests into executing software entities. Changing the rule sets changes the behavior of each software agent, while updating the software modules allows the addition or removal of software and hardware capabilities.

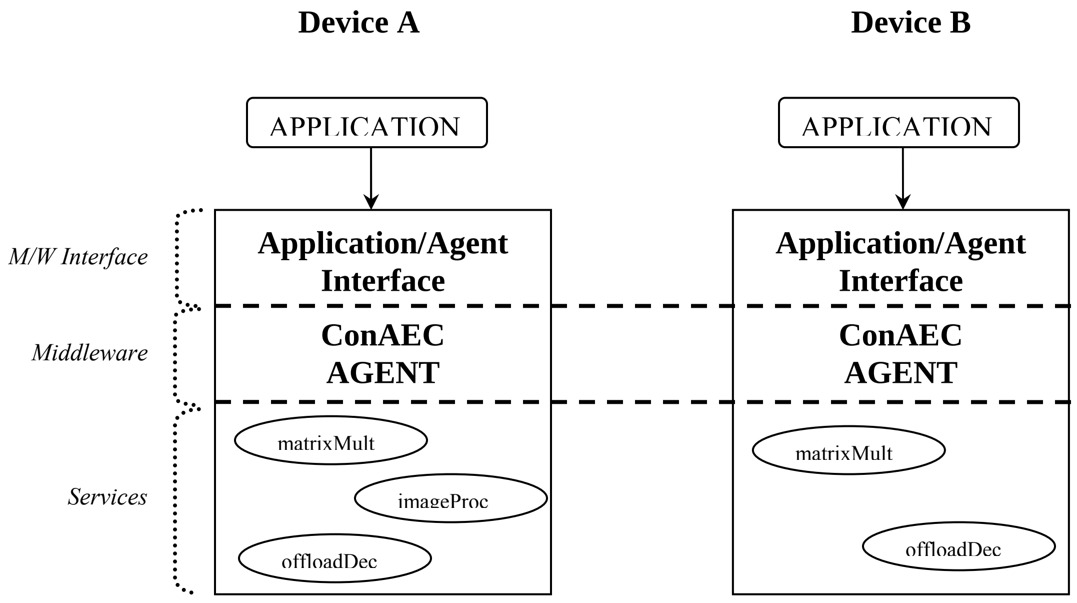
A delegent receives inputs from many sources such as external events, internal events, and intentions. A delegent can be in a dormant, active, or mobile state. A dormant delegent is activated when certain events take place in its environment. A mobile delegent migrates from one device to another. After completing the designated tasks in the remote device, a delegent returns to the active state.

4.2 ConAEC

A middleware-based service architecture called ConAEC (Context Aware Energy Conservation) was developed as part of this thesis work to examine the practicality and limitations of real world implementations. This architecture is an adaptation of the PICO middleware-based framework as described above. Discovering other devices in the network and their capabilities is important. This requires a service discovery function that provides a list of devices in the network, their IP address, the port at which they can be contacted and the services they have to offer. This protocol should also take care if devices that may dynamically enter or leave the network at any time. However, the implementation of such a protocol is beyond the scope of this thesis. One possible approach is to use a distributed server that maintains this information. Each device may register its address, port and services at this server which can be retrieved by other devices. Another possible solution is to use a point-to-point protocol in which devices share all the information they possess of other devices in communication range.

4.2.1 ConAEC Agents

ConAEC agents are the middleware entities that perform tasks on behalf of human users, software entities and hardware devices. An agent resides on each device and may interact with agents on other devices to collaborate and accomplish a service request. Each agent maintains information about the delegents residing on the local device, the services they provide and the system characteristics, like CPU, memory and remaining battery power of the device. It also keeps track of other devices in the network with information about the services offered by each of the devices, their system characteristics and the address and port at which service requests can be made. The agent runs a manager known as a delegent manager at a port number advertised to all remote agents in the environment. Service requests are made to this port. Hence, a device in the ConAEC framework can be defined as $C = < C\text{id}, C\text{adport}, C\text{h}, F>$, where Cid is the device identifier, Cadport is the address of the device and the port number at which its delegent manager can be contacted, Ch is the set of system characteristics, and F is the set of functionalities or services that it provides. In this framework, service names are unique aliases used to access middleware services. One or more service names map to a single middleware service. Requesting a service triggers an event that is handled by the corresponding delegent's rule set. Rule sets map events into software module actions which invoke methods or call functions. Modules contain sets of actions with associated input parameters that are used to invoke an executing software entity.



4.3: ConAEC Framework

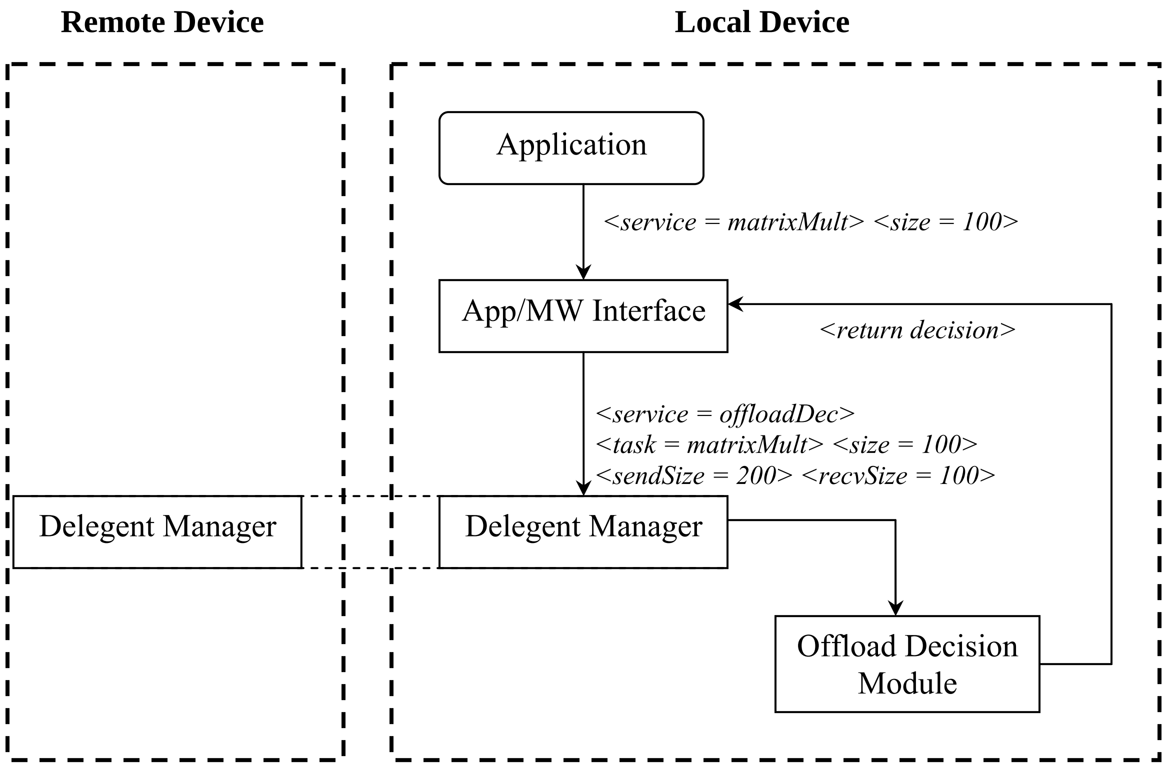
The Application/Agent Interface bridges the gap between the middleware and the application, constructs service request messages in the format defined by the middleware protocol and conveys results back to application in the appropriate format. The ConAEC agents accept service requests through the delegent manager running at a known port number, and invoke the corresponding delegents providing the service. These invoke software modules which execute functions or call methods.

4.2.2 Control flow

This section describes the flow of control within the framework when a request for a service is generated by an application. The application invokes the application/middleware interface with the appropriate parameters.

Based on the current context information of achievable throughput to each device, a decision must be made to execute the task locally or on a remote device. Hence the application/middleware interface first requests an offload decision service from the agent. It includes in the service request message the task to be executed, the size of the task, the size of the data that will have to be transmitted and received and other parameters needed to make the decision.

When the delegent manager receives the service request, it awakens the delegent that provides the offload decision service. The '*offloadDec*' service invokes the offload decision module which makes a decision based on the input parameters using the algorithm described in the previous chapter and returns its decision to the application/middleware interface.

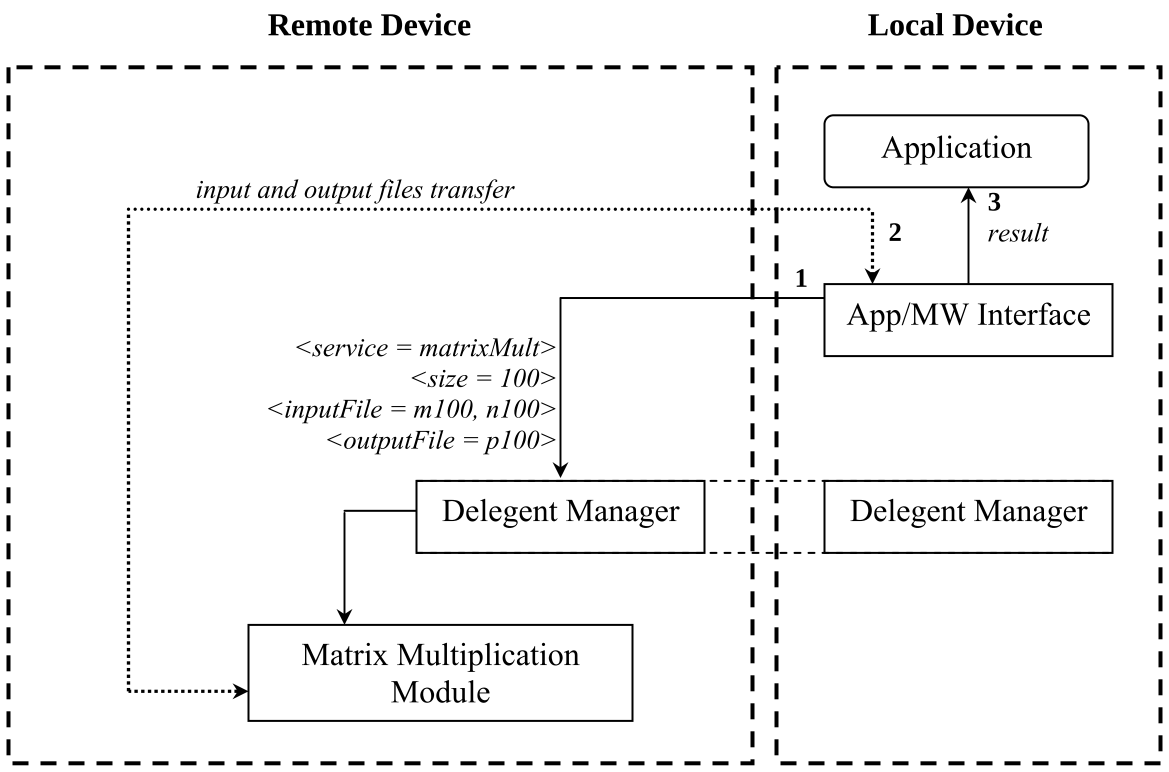


4.4: Making a service request for an offload decision

Case I: When decision is to offload

When the Offload Decision Module returns a decision to offload the task, it also returns the address and port of the device to offload to. The App/MW Interface then contacts the Delegent Manager on the remote device and requests for the '*matrixMult*' service. The delegent manager awakens the '*matrixMult*' delegent which invokes the Matrix

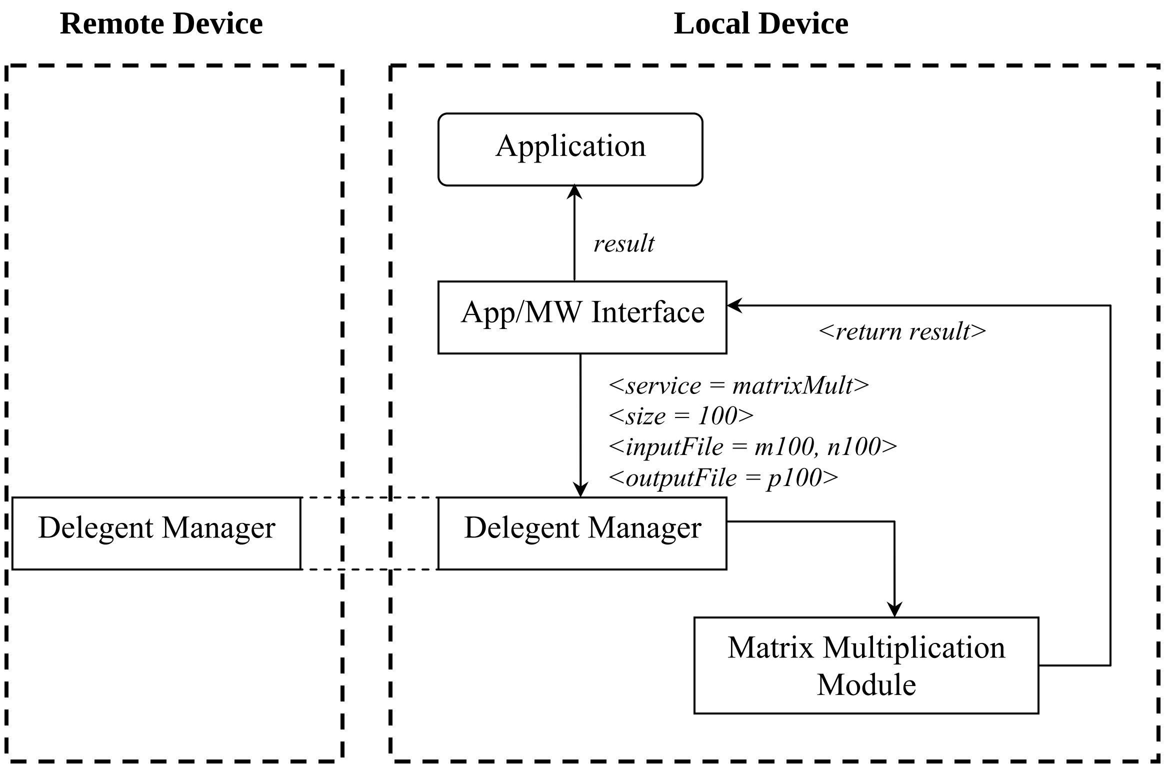
Multiplication module. This module interacts with the App/MW and requests for the files necessary to accomplish the task. The final results are shipped back to the device and are received by App/MW interface.



4.5: Offloading the task to a remote device

Case II: When decision is to execute locally

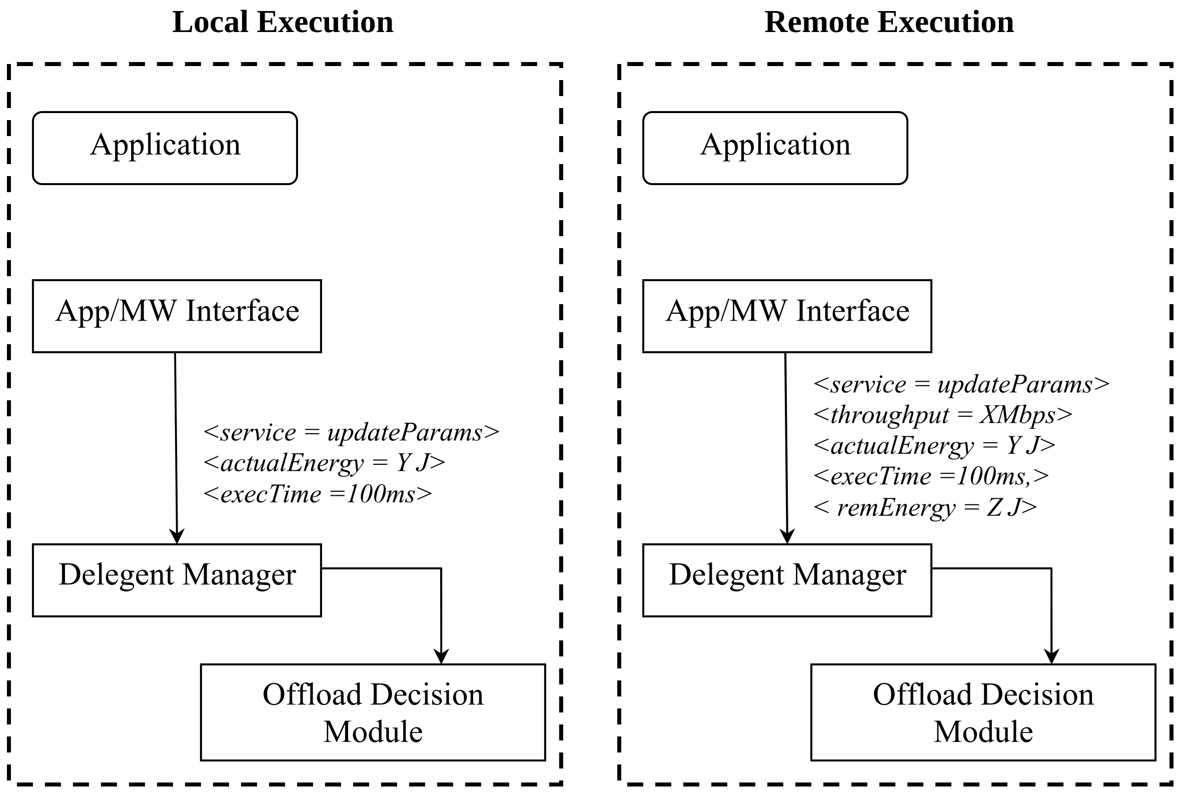
When the Offload Decision Module decides to execute the task locally, the App/MW Interface contacts the Delegent Manager and requests for the '*matrixMult*' service. The delegent manager awakens the '*matrixMult*' delegent which invokes the Matrix Multiplication module. This module performs the matrix multiplication and returns the results back to the App/MW interface.



4.6: Executing the task on the local device

4.2.3 Updating Parameters

When the service request has been completed, the parameters, such as, energy consumed, average throughput, actual execution time that are measured are updated. This is accomplished through an *'updateParams'* service. The figure shows the flow of control when the execution task is remote and local.



Updating parameters for local execution ***Updating parameters for remote execution***

4.7: Updating parameters

4.3 Environment

The environment consists of a wide variety of devices such as wired devices, wireless devices, handheld devices, sensors and so on. There exist resourceful and resource limited devices. Resourcefulness is determined by CPU speed, communications link speed, storage space, energy limitations, available peripherals, and available services.

Using this model resourcefulness is meaningful with respect to what resources are needed and when the resources are requested. For example, a PDA considers a Pentium IV desktop system to be a resourceful device with respect to CPU speed and memory. However, a desktop system considers a wireless camera resourceful for an application requiring a snapshot of a room. The end result is a community of hardware devices and software agents dynamically and statically collaborating to accomplish common goals. The architecture of ConAEC attempts to mask the heterogeneity and provide a useful distributed programming model.

4.4 Software

The ConAEC framework is implemented to be system independent and portable to small mobile devices. There are many system dependent obstacles that are overcome by using operating system and interoperability technology.

The framework is implemented using the Java framework to maximize portability and system independence. The major obstacle is that many mobile devices use a scaled down version of the Java virtual machine. This obstacle is overcome by implementing any missing components such as an XML parser, string operations and generating binary class files compatible with older Java virtual machines.

CHAPTER 5

IMPLEMENTATION AND RESULTS

This chapter describes the implementation details and shows the performance results from test bed experiments.

5.1 Devices and Environment

The environment consists of devices varying in capacity in terms of CPU processing speed and remaining energy.

The resource limited devices are two Sharp Zaurus SL-5500 PDAs. Each contains an Intel StrongARM processor running at 206MHz. They run Jeode Personal Profile for Java with 32MB of RAM. Their operating system is the Qtopia Desktop Environment (QDE) with Linux kernel 2.4.18 (Cacko). Their communications links are D-Link Air DCF-50W compact flash cards.

The first resourceful device is a Dell Inspiron 5150 laptop with a 3.06 GHz, Pentium 4 Processor and 256 MB RAM. It has a built-in Dell TruMobile 802.11g wireless network card. Its operating system is the Fedora Core 4 and it has the JDK 5.0 version of the java virtual machine.

The second resourceful device is a Toshiba Satellite Centrino Processor and 256 MB RAM. An external LinkSys 802.11b wireless network card is used for communication. Its operating system is the Fedora Core 4 and it has the JDK 5.0 version of the java virtual machine.

The devices in the environment form an ad hoc network and communicate with each other using the Wi-Fi (802.11b) protocol.

5.2 Power Specifications

Energy consumption is estimated using the time necessary to perform all subtasks of a task along with the associated power level for each subtask. The power specifications for the Sharp Zaurus SL-5500 is as follows: data transmission ($\boldsymbol{P_{tx}}$) is 1654mW; data reception ($\boldsymbol{P_{rx}}$) is 1324mW; the power dissipation for performing calculations while doing no communication ($\boldsymbol{P_{proc}}$) is 961mW; power level for processor being idle with the communication link in standby mode ($\boldsymbol{P_{idle}}$) is 661mW.

5.3 Measurement of actual energy consumption

The ConAEC framework needs to measure the actual energy consumed for the completion of a task. The most accurate way to measure the power consumption due to the execution of a task on a device would be to insert appropriate electronic instrumentation between the battery and the device. However this leads to many practical problems. Rudenko et al [8] propose the use of less direct methods. They suggest the use of the Advanced Power Metric (APM) of a device to determine this energy. The APM reports the remaining energy as a percentage of the maximum capacity of the device. The total energy in Joules, $\boldsymbol{E_{total}}$, available from a battery is obtained from the Ampere hour rating, $\boldsymbol{Q}$ of a battery, and the nominal voltage, $\boldsymbol{V}$ as,

$$E_{total} = 3600 * Q * V$$

The actual energy consumed due to the execution of a task can be calculated by obtaining the difference in the APM readings before the start and after completion of the task. This gives the percentage, $\boldsymbol{P}$, of the battery power dissipated to execute the task. The actual energy in Joules consumed for a task execution, $\boldsymbol{E_{actual}}$, is then calculated by the following equation:

$$E_{actual} = P / 100 * E_{total}$$

The display, which is one of the main sources of energy consumption on a device, is turned off during the experiments to get a more accurate estimation of the energy consumed for the execution of a task.

5.4 Assumptions and Limitations

One of the assumptions while measuring the energy consumed using the APM is that all the energy is spent only in execution of task. It assumes that other system daemons that may be running in the background do not consume significant energy.

5.5 Results

This section describes the experiments performed and discusses the results obtained.

Application I: Matrix Multiplication

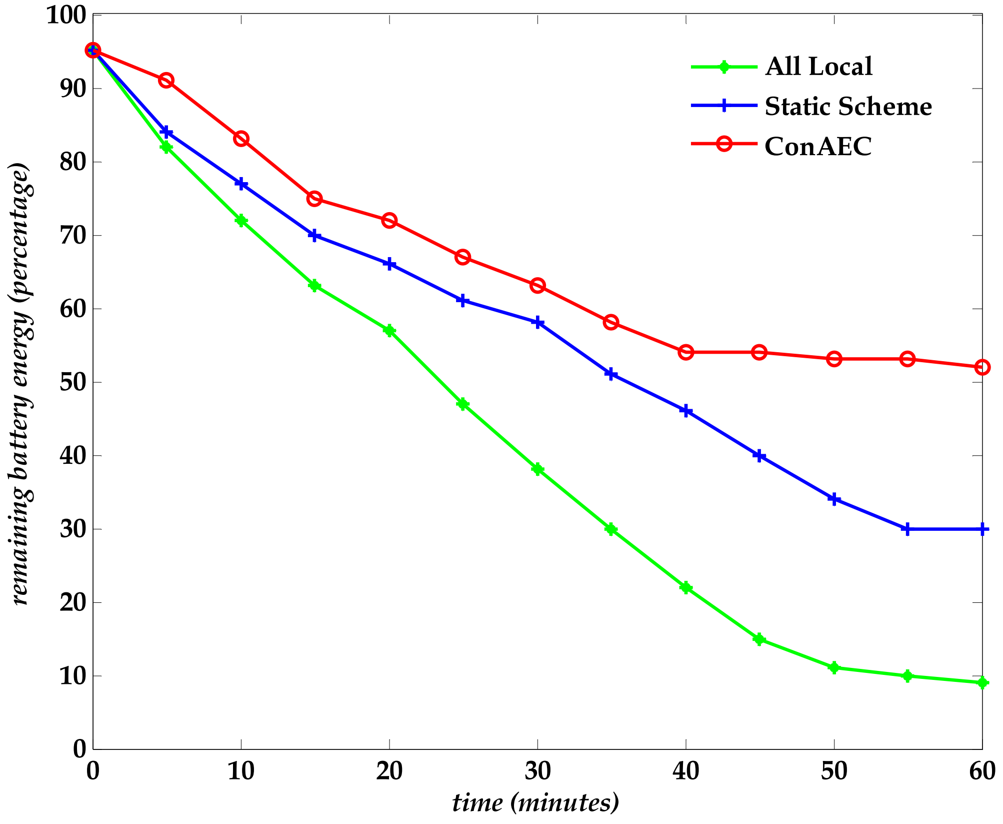
Experiment I

In this scenario, the user wishes to execute a set of tasks on the Sharp Zaurus SL-5500 PDA. This experiment measures the energy consumed by the PDA to perform a given set of tasks using the ConAEC framework. The result is compared with the scheme in which all tasks are executed locally without the ConAEC framework and a scheme in which a static context i.e. the size of the task, is used to make an offload decision. The remaining battery energy on the PDA is periodically recorded until the set of tasks is completed.

Task Set: Perform matrix multiplication of square matrices starting from size 20, in increments of 20, up to 500 and in decrements of 40 down to 20.

Environment: The environment consisted of the low power Sharp PDA and the two laptops, all providing the *matrix multiplication* service.

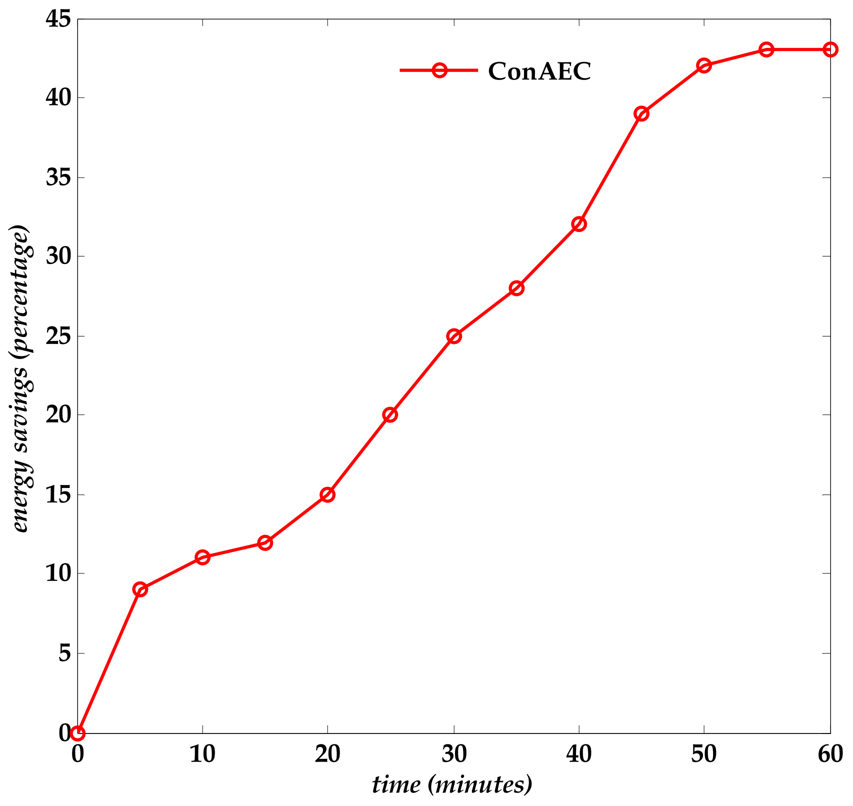
The plot shows the percentage remaining battery energy over time as the set of tasks is executed, when all the tasks are executed locally, for a static scheme where the task is offloaded when the size of the task is greater than 160, and the ConAEC scheme.



5.1: Remaining battery energy with time

We see that, when the set of tasks have been completed, the remaining battery energy in the all-local scheme is less than 10%, in the static scheme is 30% and in ConAEC scheme is more than 50%. This gives battery power savings of about 43% over the all-local scheme and about 22% over the static scheme.

This plot shows the energy savings of the ConAEC scheme over the all-local scheme as time progresses.



5.2: Energy savings for ConAEC

As mentioned before, savings of up to 43% are achieved using the ConAEC framework.

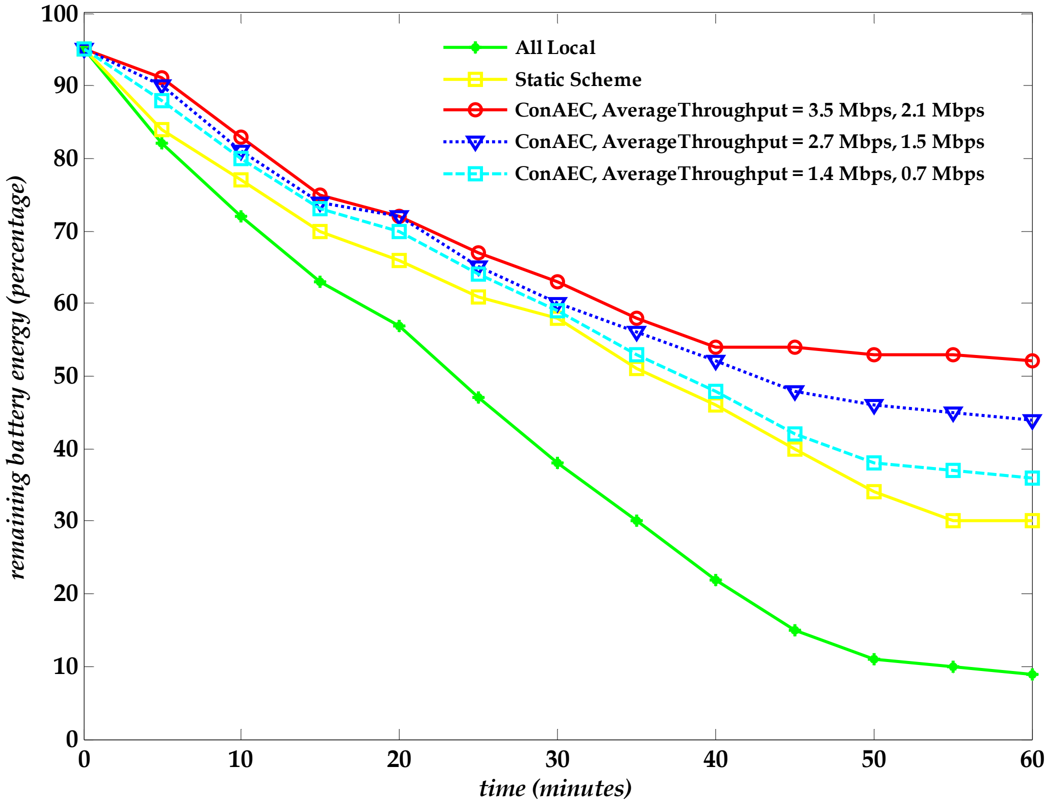
Experiment II

This experiment is conducted to measure the energy savings that can be achieved using the ConAEC scheme at different values of average throughput. A given set of tasks is executed on the PDA using the ConAEC framework. The battery power consumed to execute the same set of tasks at the different throughput values is measured and plotted. The remaining battery energy on the PDA is periodically recorded until the set of tasks is completed.

Task Set: Perform matrix multiplication of square matrices starting from size 20, in increments of 20, up to 500 and in decrements of 40 down to 20.

Environment: The environment consisted of the low power Sharp PDA and the two laptops, all providing the *matrix multiplication* service. The experiment is first conducted when there is no other traffic in the network and the average throughput to the two laptops is 3.5 Mbps and 2.1 Mbps, respectively. The experiment is repeated by introducing other traffic in the network and reducing the average throughput to 2.7 Mbps and 1.5 Mbps. Finally, the achievable throughput to each device is further reduced to 1.4 Mbps and 0.7 Mbps.

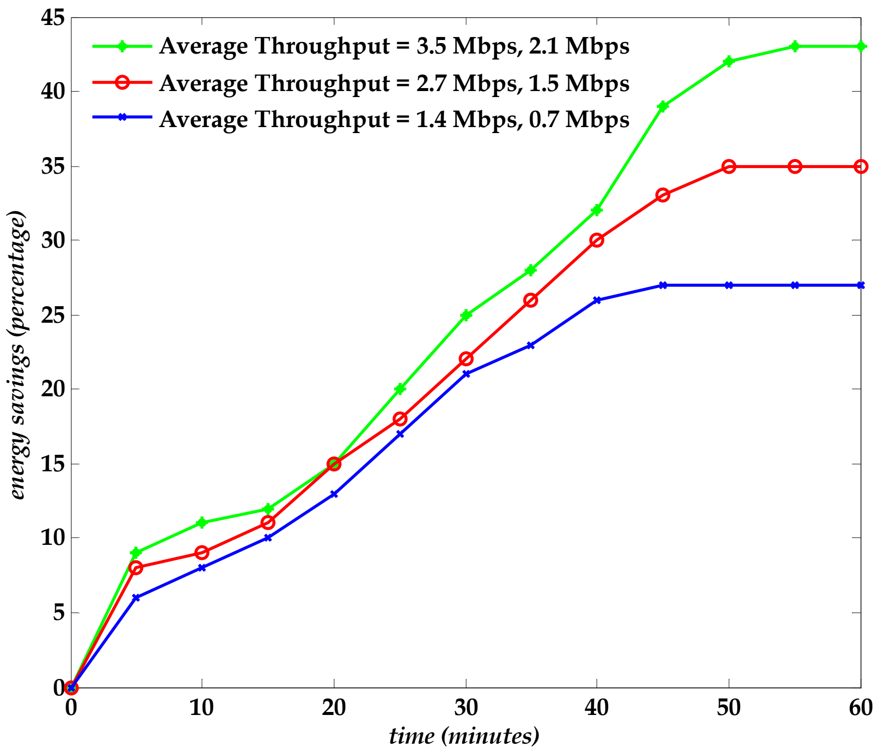
The plot shows the percentage remaining battery energy over time as the set of tasks is executed, when all the tasks are executed locally and the ConAEC scheme at different values of the average throughput.



5.3: Energy consumption at different throughput values

We see that at higher values of throughput, the remaining battery power when all tasks have been executed is higher. This is because at higher throughput, more tasks can be offloaded and the process of sending and receiving the data for task execution is faster.

This plot shows the energy savings at different throughput values for the ConAEC scheme over the all-local scheme as time progresses.



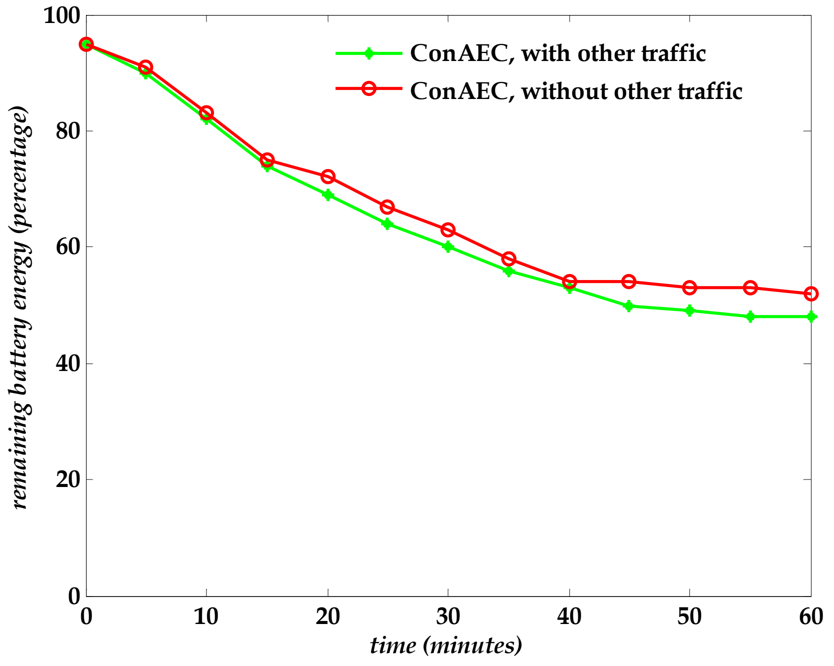
5.4: Energy savings for ConAEC at different throughput values

Experiment III

This experiment is conducted to verify how ConAEC scheme adapts to changes in the environment. A given set of tasks is executed on the PDA using the ConAEC framework. The throughput value is modified as the experiment progresses and the aim is to verify that ConAEC detects this change in context and adapts accordingly. The result is compared with the scheme in which a static context i.e. the size of the task is used to make an offload decision. The remaining battery energy on the PDA is periodically recorded until the set of tasks is completed.

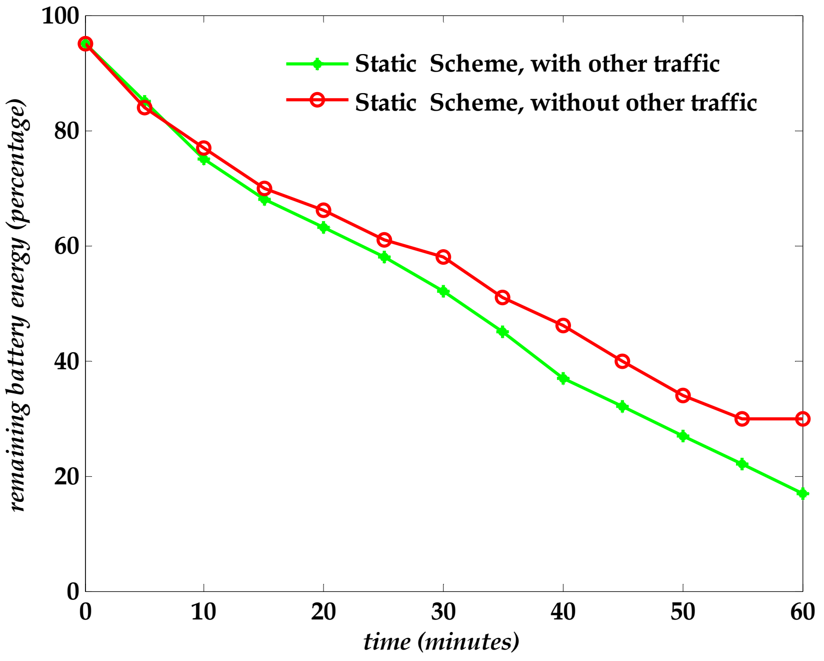
Task Set: Perform matrix multiplication of square matrices starting from size 20, in increments of 20, up to 500 and in decrements of 40 down to 20.

Environment: The environment consisted of the low power Sharp PDA and the two laptops, all providing the *matrix multiplication* service. The experiment is first conducted in a noiseless environment in which the average throughput to the two laptops is 3.5 Mbps and 2.1 Mbps, respectively. The experiment is repeated by reducing the average throughput to 2.0 Mbps and 1.7 Mbps between the 10th minute and the 35th minute, by introducing other traffic in the network.



5.5: Adaptation in ConAEC

The plot shows the percentage remaining battery energy over time as the set of tasks is executed, for the static scheme when there is no change in throughput and when the throughput is modified.



5.6: Adaptation in the static scheme

We see that the ConAEC scheme adapts better to changes in the environment as the difference in remaining battery energy between the two cases is less than 5%. ConAEC detects the change in the environment and makes a decision to offload task taking into account the reduced throughput. In the static scheme, the difference between the two cases is about 10% as it does not adapt to the dynamically changing environment and offloads tasks, based only on the size of the task

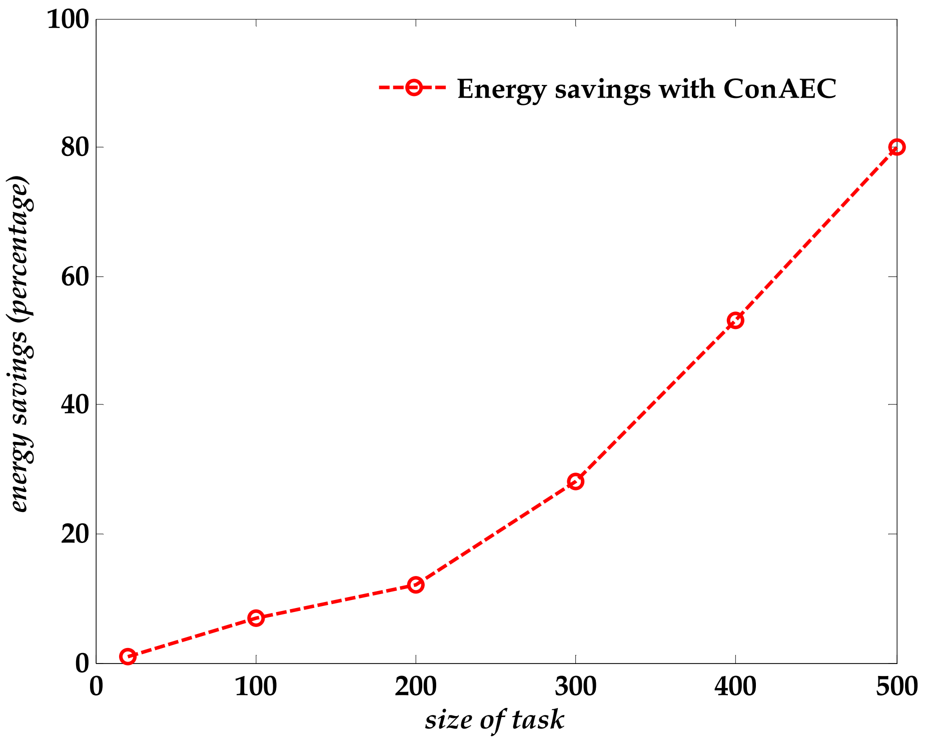
Experiment IV

This experiment is conducted to measure the energy savings that can be achieved using the ConAEC scheme for tasks of different sizes. A given set of tasks, each of different sizes, is executed on the PDA using the ConAEC framework. The result is compared with the scheme in which all tasks are executed locally without the ConAEC framework. The remaining battery energy on the PDA is periodically recorded until the set of tasks is completed.

Task Set: Perform matrix multiplication of square matrices of sizes 20, 100, 200, 300, 400 and 500.

Environment: The environment consisted of the low power Sharp PDA and the two laptops, all providing the *matrix multiplication* service.

This plot gives the energy savings for different task sizes using the ConAEC scheme over the all-local scheme.



5.7: Energy savings for different task sizes

We see that as the size of the task increases, the energy savings that can be achieved using this scheme is higher.

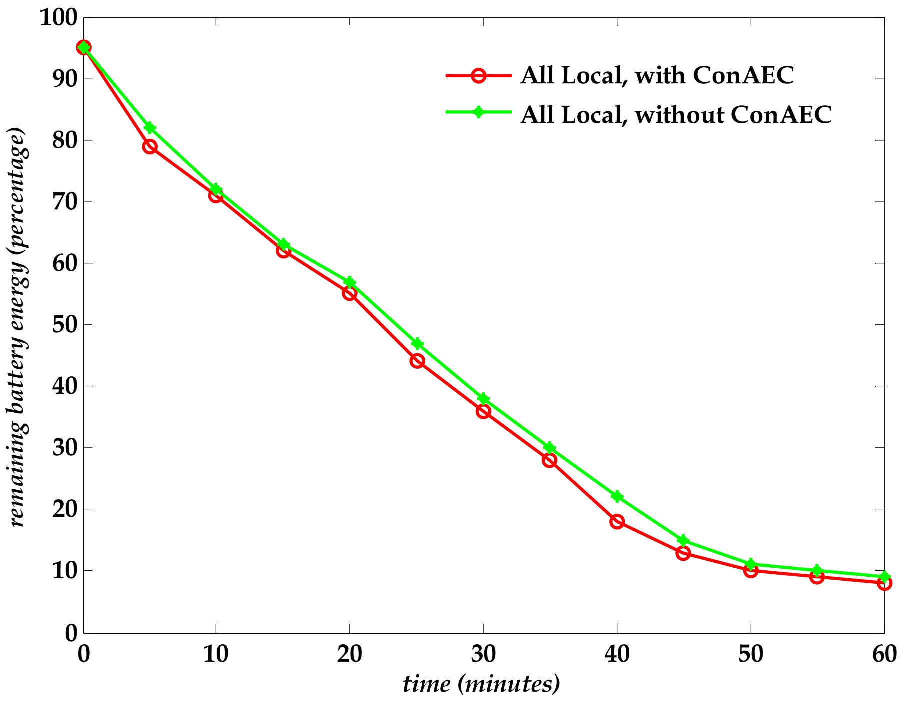
Experiment V

This experiment is conducted to measure the overhead due to the ConAEC framework. This overhead includes the cost to keep the delegent manager running at all times, the processing cost involved in the computation for storing and predicting energy consumption values, probing other devices in the network to maintain updated values of throughput to each of them and making the decision on where a task should be performed to minimize energy consumption.

A given set of tasks is executed locally on the Sharp Zaurus PDA without the ConAEC framework. The same set of tasks is executed on the PDA with the ConAEC framework. All the processing involved in making a decision is completed but the decision is, always, to execute the task locally.

Task Set: Perform matrix multiplication of square matrices starting from size 20, in increments of 20, up to 500 and in decrements of 40 down to 20.

Environment: The environment consisted of the low power Sharp PDA and the two laptops, all providing the *matrix multiplication* service.



5.8: Overhead due to ConAEC

We see that the overhead due to the ConAEC framework is negligible compared to the savings we derive by using a context aware energy conservation scheme.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This thesis work presents the design and evaluation of a novel middleware-based framework for context aware energy conservation through cyber foraging.

The framework is able to make intelligent decisions on task migration to maximize energy savings on the low power device. It also keeps this process transparent to the user.

The proposed framework is compared with a scheme in which a decision to migrate a task is made based solely on the size of the task. It is also compared with a scheme in which no cyber foraging is used and all tasks are executed locally.

Implementation results show that the ConAEC framework achieves significant energy savings through task migration by adapting to network conditions. Experiments also sho that the overhead introduced by ConAEC is negligible when compared to the energy savings we can achieve through this scheme.

The ConAEC framework can also be integrated with service discovery schemes to better support mobility. Also, support for parallel, distributed task execution can be incorporated where the task is migrated to more than device in the environment and executed in parallel for faster, distributed execution. Other contexts such as user's location, priority of the task, time of the day, daily behavioral patterns can be included to increase energy savings.

REFERENCES

1. Kumar, M.; Shirazi, B.A.; Das, S.K.; Sung, B.Y.; Levine, D.; Singhal, M.
PICO: a middleware framework for pervasive computing
Pervasive Computing, IEEE Volume 2, Issue 3, July-Sept. 2003

2. Weiser, M.
The Computer for the 21st Century
ACM SIGMOBILE Mobile Computing and Communications Review, Volume 3, Issue 3, July 1999

3. Satyanarayanan, M.
Pervasive Computing: Vision and Challenges.
IEEE Personal Communications, August 2001

4. Singh, H.; Saxena, S.; Singh, S.
Energy consumption of TCP in ad hoc networks
Wireless Networks, Volume 10 Issue 5, September 2004

5. Prasad, R.; Dovrolis, C.; Murray, M.; Claffy, K.
Bandwidth estimation: metrics, measurement techniques, and tools
Network, IEEE Volume 17, Issue 6, Nov.-Dec. 2003

6. Li, Z.; Wang, C.; Xu R.
Power-and Energy-Aware Computing: Computation offloading to save energy on handheld devices: a partition scheme
Proceedings of the 2001 international conference on Compilers, architecture, and synthesis for embedded systems, November 2001

7. Flinn, J.; SoYoung Park; Satyanarayanan, M.
Balancing performance, energy, and quality in pervasive computing
Distributed Computing Systems, 2002. Proceedings. 22nd International
Conference on. 2002

8. Rudenko, A.; Reiher, P.; Popek, G.J.; Kuenning, G, H.
The remote processing framework for portable computer power saving
Proceedings of the 1999 ACM symposium on Applied computing, February 1999

9. Rollins, S.; Almeroth, K.; MilojivHiE, D.; Nagaraja, K.
Power-aware data management for small devices.
Proceedings of the 5th ACM international workshop on Wireless mobile
multimedia, September 2002

10. Chong, S, K.; Krishnaswamy, S.; Loke, S.K.
A context-aware approach to conserving energy in wireless sensor networks
Third IEEE International Conference on Pervasive Computing and
Communications Workshops, 2005

11. Rudenko, A.; Reiher, P.; Popek, G.J.; Kuenning, G, H.
Saving portable computer battery power through remote process execution
ACM SIGMOBILE Mobile Computing and Communications Review, Volume 2
Issue 1, January 1998

12. Flinn, J.; Satyanarayanan, M.
Energy-aware adaptation for mobile applications
ACM SIGOPS Operating Systems Review , Proceedings of the seventeenth ACM
symposium on Operating systems principles SOSP '99, Volume 33 Issue 5,
December 1999.

13. Milojičić, D.S.; Douglis, F.; Paindaveine, Y.; Wheeler, R.; Zhou, S.
Process migration
ACM Computing Surveys (CSUR), Volume 32 Issue 3, September 2000

14. Stemm, M.; Katz, R.H.
Measuring and Reducing Energy Consumption of network interfaces in hand-held devices
IEICE Trans. Communications VOL E80-B, No 8 August 1997

15. Liu, R.; Chen, F.; Yang, H.; Chu, W.C.; Yu-Bin Lai;
Agent-based Web services evolution for pervasive computing
11th Asia-Pacific Software Engineering Conference, 2004.

16. Jones, C.E.; Sivalingam, K.M.; Agrawal, P.; Chen, J.C.
A Survey of Energy Efficient Network Protocols for Wireless Networks
Wireless Networks, Volume 7 Issue 4, September 2001

17. Kalapriya, K.; Nandy, S.K.; Srinivasan, D.; Maheshwari, R.U.; Satish, V.
A framework for resource discovery in pervasive computing for mobile aware task execution
Proceedings of the 1st conference on Computing frontiers, April 2004

18. Negri, L.; Barretta, D.; Fornaciari, W.
Application-level power management in pervasive computing systems: a case study
Proceedings of the 1st conference on Computing frontiers April 2004

19. Ephremides, A.

Energy concerns in wireless networks

Wireless Communications, IEEE Volume 9, Issue 4, August 2002

20. Dey, A,K.

Understanding and Using Context

Personal and Ubiquitous Computing, Volume 5 Issue 1, January 2001

21. Zimmer, T.

Towards a better understanding of context attributes

Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference March 2004

22. Schmandt, C.; Marmasse, N.

User-centered location awareness

Computer Volume 37, Issue 10, Oct. 2004

23. Schilit, B.; Adams, N.; Want, R.

Context-aware computing applications

Mobile Computing Systems and Applications, 1994. Proceedings., Dec. 1994

24. Wang, Y.

Context awareness and adaptation in mobile learning

Wireless and Mobile Technologies in Education, 2004. Proceedings. The 2nd IEEE International Workshop. 2004

25. Salber, D.; Dey, A.K.; Abowd, G.D.

The context toolkit: aiding the development of context-enabled applications

Proceedings of the SIGCHI conference on Human factors in computing systems: May 1999

26. Abowd, G.D.; Dey, A.K.; Orr, R.; Brotherton, J.
Context-awareness in wearable and ubiquitous computing
Wearable Computers, 1997. Digest of Papers., First International Symposium.
Oct. 1997

27. Zaslavsky, A.
Mobile agents: can they assist with context awareness?
Mobile Data Management, 2004. Proceedings. 2004 IEEE International
Conference on. 2004.

28. Schilit, B.N.; Hilbert, D.M.; Trevor, J.
Context-aware communication
Wireless Communications, IEEE Volume 9, Issue 5, October 2002

29. Mostefaoui, G.K.; Pasquier-Rocha, J.; Brezillon, P.
Context-aware computing: a guide for the pervasive computing community
Pervasive Services, 2004. ICPS 2004. Proceedings. The IEEE/ACS International
Conference on. July 2004

30. Yau, S.S.; Karim, F.; Yu Wang; Bin Wang; Gupta, S.K.S.
Reconfigurable context-sensitive middleware for pervasive computing
Pervasive Computing, IEEE Volume 1, Issue 3, July-Sept. 2002

31. Narayanan, D.; Flinn, J.; Satyanarayanan, M.
Using history to improve mobile application adaptation
Mobile Computing Systems and Applications, 2000 Third IEEE Workshop on.
2000

32. Noble, B.D.; Satyanarayanan, M.; Narayanan, D.; Tilton, J.E.; Flinn, J.; Walker, K.R.
Agile application-aware adaptation for mobility
ACM SIGOPS Operating Systems Review , Proceedings of the sixteenth ACM symposium on Operating systems principles SOSP '97, Volume 31 Issue 5 October 1997

33. Berhe, G.; Brunie, L.; Pierson, J.
Modeling service-based multimedia content adaptation in pervasive computing
Proceedings of the 1st conference on Computing frontiers, April 2004

34. Winograd, T.
From Computing Machinery to Interface Design
Beyond Calculation: The next fifty years of Computing, Springer-Verlog, 1997

35. Lu, Y.; Benini, L; Micheli, G.D.
Operating-system directed power reduction
International Symposium on Low Power Electronics and Design
Proceedings of the international symposium on Low power electronics and design, 2000

36. Project Aura, Carnegie Mellon University: http://www.cs.cmu.edu/~aura/

37. Project Oxygen, Massachusetts Institute of Technology
http://www.oxygen.lcs.mit.edu/

38. Endeavour Expedition, University of California, Berkeley
http://endeavour.cs.berkeley.edu/

39. PICO, University of Texas at Arlington
http://www.cse.uta.edu/pico@cse/

40. Internet2 Middleware Initiative
http://middleware.internet2.edu/

BIOGRAPHICAL STATEMENT

Prathiba Joseph received her Bachelor’s degree in Engineering at Bangalore Institute of Technology, India in 2003. She worked as a software intern at Philips Software Ltd., India during her undergraduate study and as a software engineer at Infosys Technologies Ltd., India. She received her Masters of Science in Computer Science and Engineering at the University of Texas at Arlington in August 2006. Her research interests include pervasive and wireless networks and energy conservation in mobile devices.