

Ibis for Mobility: Solving Challenges of Mobile Computing Using Grid Techniques

Nicholas Palmer
Vrije Universiteit
De Boelelaan 1081A
Amsterdam, The Netherlands
palmer@cs.vu.nl

Thilo Kielmann
Vrije Universiteit
De Boelelaan 1081A
Amsterdam, The Netherlands
kielmann@cs.vu.nl

Roelof Kemp
Vrije Universiteit
De Boelelaan 1081A
Amsterdam, The Netherlands
rkemp@cs.vu.nl

Henri Bal
Vrije Universiteit
De Boelelaan 1081A
Amsterdam, The Netherlands
bal@cs.vu.nl

ABSTRACT

Mobility is an increasingly important part of today's computing landscape. There is currently an incredible growth in the deployment of "SmartPhone" devices, mobile computers with a variety of networking and sensor technologies. In addition, the growth of wireless networks such as WiFi have untethered users from the wall bringing mobility to traditional laptop computers. The challenges that the mobility of these devices create for networked computing are analogous to many of the problems faced in the area of Grid Computing.

In this paper we outline parallel challenges in these two areas and argue that solutions to the problems in the Grid Computing space are applicable to the problems faced by these new platforms. We demonstrate how the Ibis platform, developed to address challenges in the area of Grid Computing, is ideally suited for building distributed applications for mobile devices and detail our work to bring Ibis to the Android Smartphone platform. We demonstrate that the use of this system gives mobile devices the computing power of the Grid, integrating the two areas and solving issues with limited compute power on mobile devices. We also explain how Ibis provides a unique API for building distributed applications on mobile devices enabling truly distributed computing on this new platform.

Categories and Subject Descriptors

H.1.m [Information Systems]: Models and Principals—*Miscellaneous*; C.2.4 [Computer Systems Organization]: Computer-Communication Networks—*Distributed Systems*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

HotMobile 2009, February 23-24, 2009, Santa Cruz, CA, USA.
Copyright 2009 ACM 978-1-60558-283-2/09/02-1 \$5.00.

General Terms

Mobile Computing, Grid Computing, Distributed Computing, Fault Tolerance

Keywords

Ibis, Smartphone, Distributed Computing, Mobility, Mobile Computing, Grid Computing, High Performance Computing, Fault Tolerance, Malleability, Elastic Computing

1. INTRODUCTION

Mobile devices are rapidly growing in importance in today's computing landscape in large part due to the massive deployment of so called "Smartphones". With sales of the Apple iPhone topping 1 million in the first weekend alone and approaching 10 million¹ today, Smartphones pose a compelling mobile computing platform. In particular, the Smartphone presents a unique platform for doing truly distributed computing, with applications that do not simply rely on the existence of centralized infrastructure but rather interact directly with each other in more complex and powerful ways.

Clearly the growth of the Smartphone market is huge and it is expected to continue to grow, therefore, such devices are already being hailed as the "next wave in computing[13]." Smartphones are predicted to become nearly ubiquitous and are thus a major step towards the vision of ubiquitous computing[12] so often dreamed of. The combination of pervasive wireless networks and computational devices able to take advantage of them has created an era of mobile computing, the likes of which have never been seen before.

While mobile computing is growing quickly there are still a large number of challenges that these devices face which remain open problems to be solved. Of chief importance for our work is the complexity of writing truly distributed applications for these devices². While many powerful players,

¹http://www.toptechnews.com/story.xhtml?story_id=013000R3PE55

²Distributed application state is not, in general, directly observable, which brings to mind Heisenberg's Uncertainty

from Google, to Amazon, to Microsoft, and others argue that the future is in “The Cloud”, with data and applications provided via a web browser to mobile devices from a proprietary Grid, such a platform cannot easily leverage the advanced sensors being bundled into these new devices, such as a GPS, microphone, camera or accelerometer. Nor are such applications really *distributed* applications, rather they are more similar to terminal servers and mainframes from a bygone era than to novel *distributed* applications which take advantage of the distribution of the nodes in powerful ways. The rise of peer to peer applications like BitTorrent³ demonstrate the value of moving beyond the browser to true distributed applications. The cloud solutions advanced by Google and others have numerous drawbacks not least of which is that they lock users into using a proprietary application stack hosted far from their device, with the natural problems of connectivity and node failure rendering applications unavailable, as well as force users to give up control of their precious data, a trend Richard Stallman has decried as a trap⁴

The platform we present in this paper allows users and developers to escape from proprietary Cloud platforms by leveraging the power of the Grid directly from their mobile device. This allows users to use their own computing infrastructure and non-proprietary applications combined with their own data storage. It thus provides users with cloud computing which they control, and also opens the door to massive computational power from the mobile device, addressing the lack of computational power inherent to the platform. While it certainly is the case that computational power on these devices is increasing, the limited battery power available for doing computation combined with the stagnating growth of energy densities in batteries limits the possible to include even more powerful computational elements as well as limits the complexity of calculations that can be undertaken from the mobile device. The Ibis system we present in this paper solves the limited computation problem by allowing mobile devices to be integrated into Grid environments where computational power is abundant. By offloading computation to a Grid these mobile devices can now perform much more complex operations such as realtime multimedia content analysis which we discuss in Section 4.3.

There are also problems on mobile devices associated with the diversity of communication technologies being deployed today. From WiFi to 3G, Bluetooth to WiMax, the variety of networks fall under various administrative domains, further complicating the use of them by a single device. Users encounter various firewall settings as well as Network Address Translation (NAT) as they move from one network to another and this hampers their ability to connect to other devices to do truly distributed computing. The Ibis platform we present solves these problems with a simple to program API.

We[7] and others[6] believe that Grid computing and Mobile computing face similar challenges, including limited processing power at a single node and limited bandwidth connecting nodes, as well as the need for truly distributed appli-

cations. In the following sections we will explore in more detail these issues and note the similarity of these two problem domains. In section 2 we explore the challenges of mobility in more detail. We then, in section 3, explore the similarity of these challenges to those faced in grid computing in more detail. In section 4 we present solutions to these problems in the area of Grid Computing provided by the Ibis[10, 11] platform and outline our work to bring Ibis to the Open Handset Alliances new platform for Smartphones called Android. We conclude in section 5 and outline the direction of our future work with the Ibis platform to solve the challenges posed by mobility of computing resources and bring the power of the Grid to mobile devices.

2. CHALLENGES OF MOBILITY

There are several challenges imposed on mobile computational devices such as Smartphones. Chief among this is the limited computational power available to run applications. The limiting factor is not so much the computational power but rather the battery power available to run the computations. While improvements in energy densities could fix this situation long in the future other problems will always remain. Building distributed systems is notoriously complex and the problems posed by mobile distributed systems are the same as for any other distributed system. We will first examine the problem of limited computational power and other resource limits caused by the limited battery power and then turn our attention to mobile distributed networking.

2.1 Limited Resources

The growth of energy density in batteries is not matching the exponential growth in available computational power, memory, storage or networking bandwidth in today’s computers[5]. While research is being done to harness energy from other sources[8] it is unlikely that such technologies will reach the market any time soon. This severely limits what can be done with such devices. The limited battery power has a direct impact on the available computational power and networking bandwidth available on these new devices.

As such, energy constraints are a dominating issue for the algorithm and system design trade-offs in small devices like Smartphones[2]. Fuel cells may offer higher energy densities by a factor of 5 they are not the exponential increase required to match the growth in computational power and network speed. A cubic millimeter of current battery technology has enough energy to perform about 1 billion 32-bit computations, or send and receive 10 million bits of data.[2]. One important thing to realize from this is that computation is cheaper than data transmission by orders of magnitude. None the less the lack of battery power has a direct effect on the computational algorithms used in these devices, thus efficient algorithms and programming techniques are of the utmost importance. One effective technique for addressing the limited computational power is to offload the computation to a machine which has an unlimited amount of power and we discuss how the Ibis system provides access to the vast computational resources of a Grid in section 4. To take advantage of these resources, Ibis uses the network which raises the issue of networking complexity to which we now turn our attention.

2.2 Networking Complexity

Principle, a strong statement of the complexity of building truly *distributed* applications.

³<http://www.bittorrent.com/>

⁴<http://www.guardian.co.uk/technology/2008/sep/29/cloud.computing.richard.stallman>

In 1994, Peter Deutsch outlined 7 fallacies of distributed computing to which James Gosling added an eighth in 1997[9]. These “8 fallacies of distributed computing” are faced by all networked applications and are a good representation of the complexities which application designers face. As we move toward a future of networked mobile devices, these challenges become more and more important. The first fallacy “The network is reliable” is particularly relevant in the wireless networking world where radio interference is the norm not the exception. Dropped calls are far from a thing of the past, and as devices move they are highly likely to experience network failure. Applications must be written with these disconnections in mind. Ibis provides a computational model which naturally handles the malleability required to support this as we describe in section 4.2.

The second fallacy is that “Latency is zero”. This is a particularly prevalent fallacy in the Web 2.0 world with the rise of Asynchronous Javascript and XML (AJAX) type web applications. Latency is critical to responsive applications and this is hard to get right[9] when designing AJAX based web applications. This is also a common problem with remote procedure call (RPC) interfaces and successors such as remote method invocation (RMI) type interfaces where the programming model attempts to hide the communication from the programmer. All too often application designers are tempted by these systems into making many small networking requests instead of bundling them into one larger request. The problem with the bundling solution to the latency issue is the third fallacy: “Bandwidth is infinite.” Thus a careful balance has to be struck. Latency and bandwidth are not the only concerns when it comes to networking though. There is also the cost of data transport.

Perhaps the most important fallacy for *mobile* distributed computing is the seventh fallacy: “Transport cost is zero.” The first interpretation is obvious in the Smartphone area of mobile computing where users may have to pay for every byte that they send over 3G networks. However, even if users have an unlimited data plan commonly provided with the purchase of a smart phone users still have to pay considerable energy for every byte sent. This forces application designers to consider carefully every byte they send and its cost to the user not just in terms of money but also in terms of energy. There is another interpretation however which is not as obvious. This interpretation speaks to the fact that it costs computational time to marshal data from the application layer and turn it into bits at the transport layer. This is where many of the Web Services[1] oriented architectures fail. These solutions tend to build extremely large XML documents for both request and responses. These documents are not only expensive to send due to the verbose nature of XML but are also expensive to marshal in both directions. While one could argue that the use of XML compression techniques can be used to solve the large data size this only further increases the marshaling costs for the use of such formats and is extremely wasteful of CPU cycles. There seems to be little to gain from expending CPU cycles expanding a compact binary form into a verbose document and then spending more CPU cycles to convert that back into a compact form. The popularity of XML is largely due to its ability to deal with the eighth fallacy which we look at next.

The eighth fallacy: “The network is homogeneous”, added to Deutch’s original 7 by Gosling in 1997, speaks not only to

the devices which are connected to the network but also to the network itself. It is obvious when looking at the current Smartphone market that the available devices are diverse, running different processors and operating systems. Gosling was perhaps most interested in addressing that problem through the abstraction of the Java Virtual Machine. However, it must not be overlooked that there is a wide variety of networking technologies available today as well, since, for mobile devices, this leads to issues related to the fifth fallacy: “Topology doesn’t change.” As devices move they are highly likely to move to a new networking technology or administrative domain and be given a new address in which case the topology of the network has changed. In section 5 we outline our future work at integrating the multiple networking technologies and addressing systems into one unified “Smart Address” system based on Ibis.

The availability of multiple networking technologies like WiFi and 3G networks is unlikely to disappear[5] with a single monopolistic winner of the wireless spectrum. Instead, these technologies will exist in a heterogeneous environment where devices will be able to switch easily and seamlessly between what would otherwise be competing technologies. The federation of WiFi networks available today are under a variety of administrative domains, from the owner of a coffee shop to a large telecommunications operator or a municipal government. These operators put a variety of restrictions in place via firewalls that complicates connectivity for application, particularly those which do not rely on outbound port 80, the port of the world wide web, for connectivity. This is the sixth fallacy at work: “There is one administrator.” In reality there are multiple domains with multiple administrators imposing a variety of restrictions on their portion of the network. The same is true in Grid computing and we will now look at the similarity of the challenges in that domain and outline how Ibis addresses these challenges.

3. CHALLENGES OF GRID COMPUTING

Grid Computing is targeted at addressing very large problems which require massive computational power to solve by partitioning the problem into smaller parts and distributing those parts across multiple computers. The key to achieving these results is the combination of Computational Power and the Network, each of which we will address in turn.

3.1 Computational Power

While there is no limitation of available electrical power in the Grid environment, there is a limited amount of computational power available relative to the size of the problems being solved. This means that efficient computations are still extremely important since the focus is on throughput and not simply on getting the job done. There is also a limited amount of computational power in any particular node participating in a computation. Small changes in efficiency can have dramatic effects on the final computation being performed particularly because of issues of load imbalance, where some nodes are idle waiting for the results from other nodes. This is wasteful of both time and energy.

Furthermore, energy consumption is becoming an increasingly important part of the operational costs of large scale computing infrastructure. In 2006 the nearly 6,000 data centers in the United States consumed roughly 61 billion kilowatt-hours (kWh) of energy at a total cost of \$4.5 billion dollars. The cost of the required electricity for power

and cooling in today's data centers is growing faster than spending on new hardware and already a new server may cost as much in operating costs as it does to purchase it in the first place[4]. So while the energy available in data centers is virtually unlimited the need for efficiency is important in order to keep the total costs lower. Thus the source of the limitations are not the same, but the end result is: a focus on the efficiency of operations.

Because Grid Computing focuses on throughput and efficiency many of the lessons learned in this area are of value to the mobile environment. While the motivations for wanting these efficiencies may be different the end results are the same. Computation is only one of the areas where efficiency is important. Because high performance computing relies on the network to connect the various computational elements together networking efficiency is equally important, thus, it is this area which we address next.

3.2 Networking Complexity

Naturally, all of the fallacies of distributed computing apply just as much in Grid Computing as they do in Mobile Computing by virtue of the fact that they are both distributed systems. Grid Computing is continuously faced with the first fallacy of the network being reliable. Connectivity between wide area clusters can fail, as well as individual machines within a given cluster becoming faulty. Any Grid system of appreciable size has to deal with faults simply because of the large number of machines involved and thus the high probability of failure during a given computational run, particularly as run times become longer.

Latency is also important, particularly in applications which have points in the computation where machines require the results of computations which have been completed on other machines. There are many grid applications which get acceptable speedups on a cluster but which fail miserably as soon as there are high latency links in the system. Thus Grid programming is intimately familiar with the problems posed by latency. Latency also has to be addressed at the transport layer just as in mobile computing and grid systems have developed zero copy transport systems in order to minimize transport latency as we discuss in Section 4.1.1.

Furthermore, many Grid systems offer multiple interconnect systems such as Myrinet or Infiniband for low latency local connectivity combined with Internet access for connecting clusters together. We are currently exploring the use of Multi-Ibis, an Ibis implementation capable of using both systems of connectivity simultaneously, in order to take advantage of these multiple connectivity systems. This is analogous to the multiple wireless networks available in mobile devices. What is more, these clusters often fall under different administrative domains with various firewall policies which complicate the connectivity picture due to the various administrative domains which they run in. There is certainly no "One Administrator" and this often leads to varying middleware and firewall systems which applications must contend with and Grid middleware has been developed to address these issues.

Another major issue is the challenge of fault tolerance. With scale comes an increase in probability that a computing resource will fail. These failures become almost certain as the length of time a system is running increases and thus dealing with faults is absolutely required. These faults can be viewed as identical to the faults encountered when mobile

nodes move from one network to another or disconnect all together. Ibis offers a uniquely malleable system capable of dealing with issues of fault tolerance and thus issues of scale as described in section 4.2.

The final challenge we wish to address for Grid applications is that they have to run on heterogeneous systems using a variety of hardware, operating systems and library versions. This is a major problem for compiled applications and is the main reason that our Ibis platform is based upon Java with its Virtual Machine abstraction. With the forthcoming Android platform, being as close to a Java Operating System as we have yet seen, we believe that the value of this approach is about to be realized in the mobile space even more than the limited interface offered by Java Micro Edition has already shown. As such we will next look at our Ibis platform and detail the work we have done to bring Ibis to the Android platform.

4. IBIS: GRIDS AS PROMISED

Ibis[10, 11] is a High Performance Computing Platform with the goal of providing "Grids as promised." While it isn't targeted at mobile computing, as we outlined above the challenges which Ibis addresses for Grid Computing are similar to those on mobile devices. The power of Ibis comes in the form of a distributed communication library which solves many of the challenges of building distributed applications. The API provides primitives for one to one, one to many and many to many connectivity required to create distributed applications. It also includes Smart Sockets which solves the connectivity problems encountered with firewalls common to mixed administrative domains. It also provides an efficient serialization interface for data exchange. Most importantly it gives application programmers an effective model for building systems with the malleability necessary for dealing with issues of faults in order to achieve scalability. We will discuss each of these in turn.

4.1 Distributed Communication

There are a variety of communication primitives that are required to easily build distributed applications. First and foremost is naturally unicast, or point-to-point communication. This is the model of the world wide web and it is clearly very effective. However, it is also limited in that it is not possible for a single message to easily be sent to multiple recipients. The rise of truly peer to peer streaming applications built in Macromedia Flash or even as full blown applications is indicative of the problems with the web model. When peer to peer models are not used the content providers are required to build up complex distribution networks for handling the broadcast operations required in order to allow the leaf nodes receiving the content to be sent via the webs unicast primitives. Thus, for distributed applications it is important to provide broadcast primitives, including advanced primitives required for many to many operations. The consistent interface to these primitives comes in the form of a named port and message interface where applications can construct a named port and a message to be sent to that port.

By providing a port and message oriented abstraction Ibis discourages programmers from making the mistake of the 7th fallacy of distributed computing that "transport cost is zero". Unlike Remote Method Invocation (RMI) or Remote Procedure Call (RPC) interfaces which encourage program-

mers to forget that they are making a call over the network. Instead Ibis forces programmers to be aware of the fact that they are about to pass a message over the wire. None the less Ibis marshals messages extremely efficiently in order to minimize the transport cost. This interface also minimizes the latency of the transport layer using a zero-copy implementation whenever possible. This also helps application designers to be aware of the number of bits they are sending over in order to discourage them from making the mistake of the 3rd fallacy, “Bandwidth is infinite”. Users can use a purely data oriented message but are also free to serialize objects. This serialization is done using an extremely efficient implementation which we address next.

4.1.1 Efficient Serialization

Ibis comes with an extremely efficient serialization interface in the form of the Ibis IO package. This interface uses less data than traditional Sun serialization while also providing better performance[10]. This result is achieved by shifting the cost of type inspection, required to determine what data an object is composed of, from runtime to compile time. This is achieved by using an extra compiler, Ibisc, which examines the bytecode for compiled class files and adds methods and constructors required to serialize the object. This interface is identical to the standard Sun serialization interface so it is familiar to programmers. We have recently ported this interface to Java Micro Edition for use on the wide array of mobile devices and have shown that not only is it able to bring serialization to a platform that did not have it, but the implementation beats out hand coded implementations when dealing with array oriented data due to the zero copy implementation[7]. This interface alone demonstrates the value of bringing high performance techniques from Ibis to mobile computing platforms. However, Ibis still has much to offer the mobile computing world. Of particular interest is the Smart Sockets interface which solves connectivity problems and which we discuss next.

4.1.2 Smart Sockets

Smart Sockets is an abstraction of a standard socket interface which provides a number of features for addressing the problems of Firewalls and Network Address Translation (NAT) commonly found when dealing with the multitude of administrative domains comprising today’s Internet. It provides a socket interface which uses JSTUN⁵, a Java implementation of the STUN protocol described in IETF RFC 3489⁶ for dealing with issues related to NAT, and the SBBI UPNP library⁷ in order to deal with issues associated with firewalls. It also uses a system of hubs as a last resort for connectivity when UPNP cannot be used to open an inbound port to the computer in question. The device connects out to a hub and then inbound connections are proxied through the hub. This system provides transparent socket connectivity which hides the complexity of resolving most connectivity issues.

As proof of its value in the mobile arena, while porting Ibis to Android, we were unable to make connections in to the Android phone running in the emulator because of the emulators built in firewall. By using a smart socket hub we were able to seamlessly connect multiple emulators together and

⁵<http://jstun.javawi.de/>

⁶<http://www.ietf.org/rfc/rfc3489.txt>

⁷<http://www.sbbi.net/site/upnp/>

route inbound and outbound connections without a hitch. While web based applications do not usually require special tricks since connections are only made outbound, there are many protocols which simply cannot work properly without inbound connectivity, particularly those for more advanced distributed applications.

The combination of Smart Sockets for resolving connectivity issues, Ibis Serialization for efficient IO and the Ibis Portability Layer API for doing messaging and providing distributed communication but there is another important piece of the Ibis system that is particularly important for Mobile Systems: malleability. It is this facet of Ibis which we explain next.

4.2 Malleability

The term “Malleability” is borrowed from physical sciences where it describes a material that can deform easily under compressive force without breaking. In the context of distributed systems we use it to describe a system which adapts easily without breaking under the compressive force of node disconnections and re-connections possibly due to node failure. To this end Ibis has a novel resource tracking model for dealing with these issues in the form of a “Join, Elect, Leave” (JEL) API. A deceptively simple concept, JEL gives applications using Ibis the adaptability required to not only deal with faults, but also to scale well across thousands of nodes. Amazon uses a similar term, “Elastic Computing”, to describe a system that is able to scale up and down easily through the addition and subtraction of computational resources to the running system. We do not use this term because we feel it is targeted at scalability issues only, while we feel that malleability implies elasticity under stress and thus focuses more on fault tolerance which is a prerequisite for scalability. Thus elasticity comes naturally from application malleability which the JEL API provides.

The Join operation in the JEL model gives applications notification that a new node has connected to the distributed system allowing applications to scale up in an elastic way as additional resources are added. The elect operation allows an individual to be selected, using a named election, from out of the current set of connected computational nodes allowing the election of leaders or other coordinating roles in the computation. The leave operation notifies the application that a particular computational node disconnected from the system allowing the application to scale down in an elastic way even when the leave is triggered by a fault, which in the case of a mobile node could be because of a loss of connectivity due to mobility. More specifically, leaves can either be triggered voluntarily by an application or can be generated automatically when the node appears to be dead. If an elected node leaves, members will notice the leave and can call the same named election again to elect a new node to act in the elected role. This provides the malleability to handle faults as well as the elasticity required to scale while allowing specific coordination nodes to be selected (and reselected) throughout the lifetime of the application. This very simple set of operations, provided by Ibis, gives applications the malleability required to adapt easily to changing conditions under the compressive force of node disconnections and re-connections and thus also provides for scalable resource tracking. This malleability is precisely what is needed to deal with mobility where applications connect and disconnect as they move between networks or loose connectivity.

With this in mind we have ported Ibis to the Android Smartphone platform. We next outline our efforts in this area.

4.3 Ibis on Android

Because Android is a Java based platform surprisingly little work was required to bring Ibis to this new platform. This clearly demonstrates the value of using Java to deal with the issues of Heterogeneity. We also immediately noted the value of the Smart Sockets interface in providing connectivity within the emulated Android environment. Because of the firewall inherent to the QEMU emulator it is impossible to get an inbound connection, however, with Smart Sockets we were immediately able to connect two emulators together and run sample applications making connections between both emulators as noted in section 4.1.2

We have also already run experiments on the Android emulator in which we enhanced the mobile phone with the computational power of the Grid. In the first of these experiment we use the phone as a portal to the Grid, on which we run a distributed application⁸ The distributed application is stored on the phone and using a mobile version of our deployment tool, called Ibis Deploy, we start it on our Grid. The status of the nodes as well as the topology of the parts of the distributed application are monitored on the phone. We are also currently building a distributed applications, where a part of the application runs on the phone using its sensors (i.e. the camera), and the compute intensive part runs on the Grid (image content analysis), thereby taking advantage of the strong points of both environments.

5. CONCLUSIONS

In this paper we have outlined some of the challenges mobile distributed computing faces and noted the similarities faced by applications running in Grid environments. From limits on per node computational power, to the challenges of distributed networking we have demonstrated how the technology from Grid computing can be applied in the mobile space. We further demonstrated the value of Java for heterogeneous systems by bringing Ibis from a Grid platform to the forthcoming Android mobile platform. We have also outlined how the Ibis system brings the power of Grid computing to mobile devices solving the problem of limited computational power on mobile nodes. In addition, we have outlined how Ibis provides the malleability required to properly handle faults, a prerequisite for scaling applications in an elastic way. Finally, we have demonstrated how Ibis can be used to build new distributed applications such as for image content analysis.

Our future work will be continuing work on the “Multi Ibis” system capable of communicating simultaneously using different wireless technologies. This is required for mobility since devices move from one networking technology to another, often changing address in the process. To this end we are working on a system which bundles multiple addresses into a single “Smart Address” for the device. While technologies like Mobile IP[3] in the IPv6 specification are intended to address some of these problems, deployment of IPv6 has been slow. It also only solves the problem for IP based networks. Bluetooth is not based on IP and so Mobile IP can

⁸We have run a ‘Rest In Pieces’ (R.I.P.) application, which does a parallel distributed sleep. The sleep operation is a place holder for useful computation which we have done in further experiments.

not deal with nodes switching from a Bluetooth network to an IP network and back again. To this end we also intend to implement a version of Ibis running on top of Bluetooth in order to bring a unified API for bluetooth and IP networks to a single device, as well as researching the integration of Delay Tolerant Networking into the Ibis framework.

Finally, we are also working on porting our JavaGAT API, an abstraction over various Grid middlewares, to mobile devices in order to allow the widest possible deployment of applications written using Ibis. In this way we will bring the value of Ibis and Grid applications to mobile devices and bring the computational power of the Grid to the mobile world.

6. ACKNOWLEDGMENTS

The authors would like to thank Rob Van Nieuwpoort, Jason Maassen and Niels Drost for their work creating the Ibis platform, as well as Frank Seinstra for his work on the image content analysis application.

7. REFERENCES

- [1] CHRISTENSEN, E., CURBERA, F., MEREDITH, G., AND WEERAWARANA, S. Web Services Description Language (WSDL) 1.1. Tech. rep., W3C, 2001.
- [2] ESTRIN, D., CULLER, D., PISTER, K., AND SUKHATME, G. Connecting the Physical World with Pervasive Networks. *IEEE PERSVASIVE COMPUTING* (2002), 59–69.
- [3] JOHNSON, D., PERKINS, C., ET AL. Mobility Support in IPv6. *Work in Progress 11* (1998).
- [4] LAWTON, G. Powering Down the Computing Infrastructure. *COMPUTER-IEEE COMPUTER SOCIETY- 40, 2* (2007), 16.
- [5] LEHR, W., AND MCKNIGHT, L. Wireless Internet access: 3G vs. WiFi? *Telecommunications Policy 27*, 5-6 (2003), 351–370.
- [6] MCKNIGHT, L., HOWISON, J., AND BRADNER, S. Guest Editors’ Introduction: Wireless Grids—Distributed Resource Sharing by Mobile, Nomadic, and Fixed Devices. *Internet Computing, IEEE 8, 4* (2004), 24–31.
- [7] PALMER, N., KIELMANN, T., AND BAL, H. Serialization for ubiquitous systems: An evaluation of high performance techniques on java micro edition. In *UBICOMM* (Valencia, Spain, 2008).
- [8] PARADISO, J., AND STARNER, T. Energy Scavenging for Mobile and Wireless Electronics. *IEEE PERSVASIVE COMPUTING* (2005), 18–27.
- [9] ROTEM-GAL-OZ, A. Fallacies of distributed computing explained. *RGOArchitects*, Available: <http://www.rgoarchitects.com/Files/fallacies.pdf> (2006).
- [10] VAN NIEUWPOORT, R., MAASSEN, J., HOFMAN, R., KIELMANN, T., AND BAL, H. Ibis: an efficient Java-based grid programming environment. *Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande* (2002), 18–27.
- [11] VAN NIEUWPOORT, R., MAASSEN, J., WRZESINSKA, G., HOFMAN, R., JACOBS, C., KIELMANN, T., BAL, H., VAN NIEUWPOORT, R., MAASSEN, J., WRZESINSKA, G., ET AL. Ibis: a flexible and efficient Java-based Grid programming environment. *Concurrency and Computation Practice and Experience 17*, 7-8 (2005), 1079–1107.
- [12] WEISER, M. The Computer for the Twenty-First Century. *Scientific American 265*, 3 (1991), 94–104.
- [13] ZHENG, P., AND NI, L. *Smart Phone and Next Generation Mobile Computing*. Morgan Kaufmann, 2006.