International Journal of Computers, Communications & Control
Vol. III (2008), No. 1, pp. 51-59

Mobile Message Passing using a Scatternet Framework

Brendan J. Donegan, Daniel C. Doolan, Sabin Tabirca

Abstract: The Mobile Message Passing Interface is a library which implements MPI functionality on Bluetooth enabled mobile phones. It provides many of the functions available in MPI, including point-to-point and global communication. The main restriction of the library is that it was designed to work over Bluetooth piconets. Piconet based networks provide for a maximum of eight devices connected together simultaneously. This limits the libraries usefulness for parallel computing. A solution to solve this problem is presented that provides the same functionality as the original Mobile MPI library, but implemented over a Bluetooth scatternet. A scatternet may be defined as a number of piconets interconnected by common node(s). An outline of the scatternet design is explained and its major components discussed.
Keywords: Bluetooth, Scatternet, Message Passing, Network Formation

1 Introduction

Mobile technology is one of the fastest growing fields of technology, with over one billion mobile phones shipped during 2006 [8]. The power of mobile devices is also growing quickly, in October 2005 ARM announced the ARM Cortex A8 [1][2], having a clock speed of 1Ghz. This increase in both availability and performance makes mobile devices a prime candidate for parallel computing systems. Having a multitude of mobile devices available one can now take advantage of the situation by performing complex computational tasks across several devices. The original MMPI library was restricted in terms of its world size by the upper bound of the piconet network standard, this limits the maximum number of nodes in an MMPI world to eight. The evolution of the MMPI library to allow for scatternet formation requires significantly more work behind the scenes, both to setup the network infrastructure and to allow for complete inter-device communication.

1.1 Need for Mobile Parallel Computing

The MMPI library was created to allow for parallel computing across mobile devices [6]. Mobile devices have very limited resources and processing capabilities. The Nokia 6630 and 6680 [14] mobile phones have 220Mhz processors and just a few megabytes of memory. Along with such limited capabilities they also have a finite amount of battery power. Running processor intensive applications on such devices drains battery power at a higher rate than standard phone usage. Therefore if one device doesn't have sufficient battery power it may not be capable of solving a complex problem. Such a problem would also take too long to process from the users perspective. Splitting the processing among several devices not only speeds up the processing, but also distributes the battery drain across all devices, allowing more computationally intensive tasks to be performed.

1.2 Review of Scatternet Formation Algorithms

The Bluetooth Specification [3], describes the concept of a scatternet. A scatternet (Figure 1) is defined as two or more piconets joined together through the mechanism of a common node (bridging node). Significant research has been conducted on scatternets, much of it focusing on how they can be optimized. Miklos [10] described some of the aspects of scatternet formation, including the performance bottleneck caused by the bridging node switching between the frequency hopping patterns of its masters.

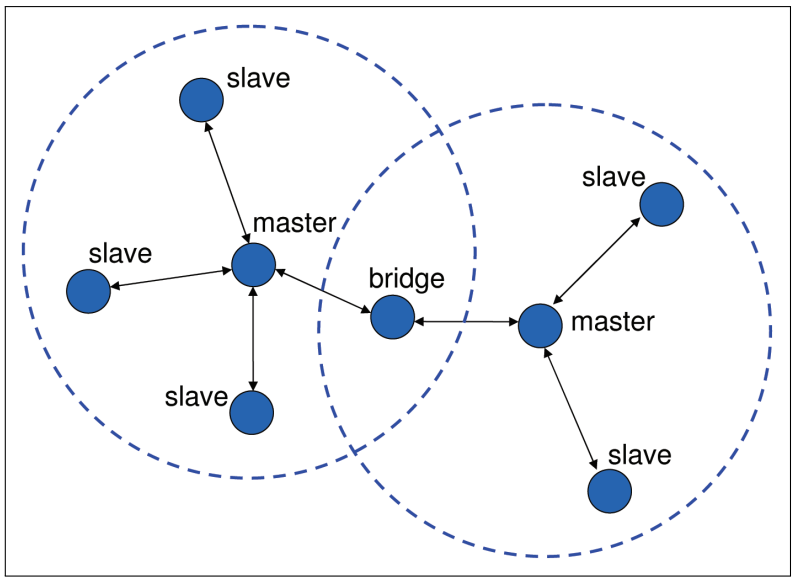

Figure 1: Scatternet Topology

The DynaMP project [13] is a mobile parallel computing architecture that uses Bluetooth Scatternets as the underlying infrastructure. The application area for this body of work is in mobile robotics, specifically that of the Cybot toy. The main reason for the parallel computing architecture is real-time image processing of data acquired from the robots sensor system. To avoid the upgrading of the processing capabilities of the robots, the path of distributing the processing requirements across a group of robots was investigated. The target architecture is the TINI from Dallas Semiconductor that uses java as its native environment. The iPAQ PDA is also used, running a Linux based OS and the Keffe JVM.

Unlike native message passing libraries such as the C and Fortran based MPI that use a static number of nodes, a far more dynamic approach is required within the mobile environment. The head node is therefore responsible for discovering all the active nodes within the network. This is achieved by broadcasting a Process Request packet throughout the network. An example of a simple node topology is given in the paper that shows a linear like architecture, thereby communication between distant nodes would certainly require one or more intermediary nodes for forward on the message.

Bluetrees [15] generates a tree-like scatternet. The network formation algorithm is initiated by a single node that forms the root of the tree. The root node begins by acquiring slave devices that are within it vicinity. These in turn page their own neighbours. In order to limit the number of slave devices connected to another node at the level above some branch reorganization may be required.

Jayanna & Zaruba [9] use an approach where by all nodes maintain a dynamically generated list of its neighbours based on the number of hops required to reach the node in question. The strategy ensures that if neighbour information is included for one and two-hop piconets, then two adjacent piconets can share only one bridging node. Therefore all nodes of one piconet share a single path to all nodes of a neighbouring piconet.

1.3 Mobile Message Passing Interface

The Mobile Message Passing Interface (MMPI) [6] provides many of the functions that can be found in a standard MPI [11] [12] implementation. The main difference between MMPI and standard MPI implementations is that it is designed for mobile devices that communicate via Bluetooth [4] [5]. The library has one drawback that makes it less than an ideal candidate as a platform to support parallel computation on mobile devices. It was designed to operate only on a Bluetooth piconet of up to eight

devices. In a parallel computing system in general one can achieve faster computation by throwing more processors at the problem.

2 Enhanced MMPI Library Structure

The problem of creating scatternets and performing communication within them after they have been created is a complex one. The original MMPI library could not simply be adapted, as the differences between communication in a piconet and communication in a scatternet are too great. A new solution was therefore designed from the ground up, whilst retaining all of the original functionality. The enhanced library comprises of a number of distinct components that provides an abstraction from lower level layers.

2.1 Components of the Library

The enhanced MMPI library is made up of a number of components. The most important parts of the architecture are shown in Figure 2

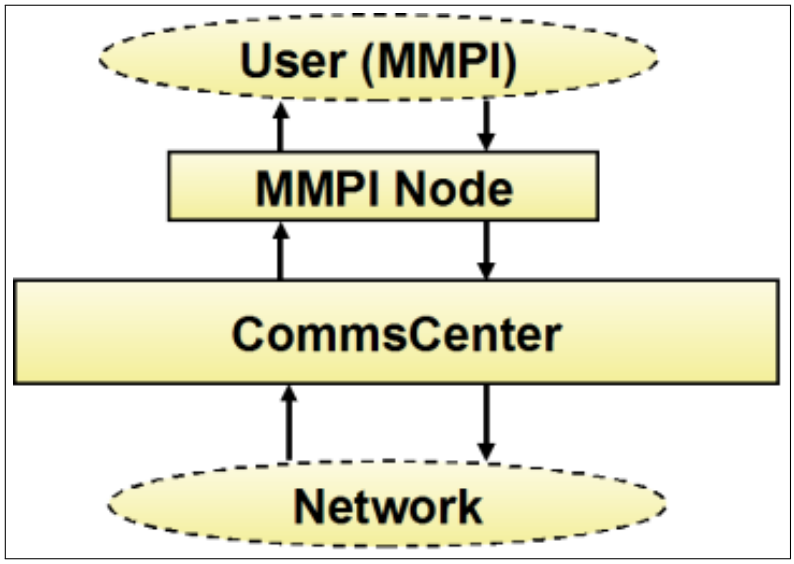

Figure 2: Library Components

The CommsCenter class forms the heart of the library. Its role is to receive raw data from the network and translate it into MMPI messages. The MMPINode class provides the interface between the CommsCenter and the MMPI class and also performs the initial device discovery. Finally, the MMPI class is the interface to the library as a whole, and is the only class whose methods are exposed to the developer using MMPI. Relevant data is fed up through the hierarchy, starting at the CommsCenter and continuing up to the MMPI class. This simplifies matters greatly as information that is relevant at a particular level of the hierarchy is available only at that level.

2.2 Messages

On receipt of a message by the CommsCentre it must inspect the header to identify the type of the message it has received and accordingly perform the appropriate action.

The message types are categorised as follows:

- Bridge - The node should take up the role of bridging node.

- Master - The node should take up the role of master.
- Slave - The node should be a slave exclusively.
- Confirm - The network formation is complete.
- Data - The message contains a data payload.

The first four messages are used only during the formation of the scatternet and the Data message is used only after the scatternet is formed. Messages will only be processed by the node for which they are addressed, otherwise they will be forwarded by the message routing system.

2.3 Scatternet Formation

Formation of the scatternet is initiated at a chosen node (the 'root node') by first performing an inquiry for devices that provide the MMPI service. The root node then determines how many piconets are required to support this number of nodes. In the case of thirteen MMPI capable devices are discovered then two piconets will be created. The root node selects a device to be the bridge node and sends it a list of addresses of the other nodes in the network (Algorithm 1).

Data: List of all MMPI capable devices
initialize list of devices to send to bridge node;
foreach *device in the list* **do**
    if *bridge node* **then**
        create slave connection;
        add bridge node to routing table;
    else if *prior to bridge node* **then**
        create slave connection;
        add slave node to routing table;
        send Slave message;
    else if *after bridge node* **then**
        add device to list of devices to send to bridge node;
        add node to routing table;
add number of devices to Bridge message;
add list of devices to the Bridge message;
send Bridge message;

Algorithm 1: Establishing initial connections at the root node

The bridge node then chooses one of these to be the master of the other piconet and sends it the list minus the device chosen (Algorithm 2). The second master makes connections to each device on the list, completing the scatternet formation. It then sends a confirmation message which propagates through the network via the message routing system.

2.4 Message Routing

In order for a message to reach a recipient with which the sender has no direct connection, the message must be routed through the network. The library uses a simple message routing table (Table 1) where each node keeps a table of all the nodes except itself. Each entry in the routing table contains an index that is used to navigate between nodes.

```
Data: list of devices sent by root node
initialize list of devices to send to additional master;
foreach device in the list do
    if second master then
        create master connection;
        add second master to routing table;
    else
        add node to routing table;
add number of devices to Master message;
add list of devices to the Master message;
send Master message;
```

Algorithm 2: Establishing connections at the bridge node

Node	Node 2	Node 0	Node 5	Node 6
0	0	-	0	0
1	0	0	0	0
2	-	1	0	0
3	0	2	0	0
4	0	3	0	0
5	0	4	-	0
6	0	4	1	-
7	0	4	1	1
8	0	4	1	2
9	0	4	1	3

Table 1: Routing tables for nodes 2, 0, 5 & 6 of a ten node scatternet

In order to send a message from node 2 to node 8 (Figure 3), the following steps are taken. Node 2 looks up node 8 in its routing table (Table 1). Since node 2 has only one link, the index will be 0, and the message will be sent through that link to node 0. Node 0 then looks up its routing table, finding the index 4, and sends the message through that link to node 5. Node 5 looks up the routing table again and sends the message through link 1 to node 6. Node 6 then looks up node 8, finding the index 2 and sends the message to it through that link.

In summary, for a slave node of piconet one (on the left) (Figure 3) to communicate to a slave node of piconet two (on the right) all communications traffic is first routed through the master node for piconet one and forwarded on to the bridge between the two networks. This is then picked up the the master node of piconet two and again forwarded on to the correct end point. For a node such as the master node of piconet one to communicate with the master node of piconet two, data need only be forwarded directly through the bridging node.

3 Using the Library

Using the library to write message passing programs is nearly identical to the original library. The operations that were supported in the original version of the library are still supported.

3.1 Creating an MMPI Node

To create an MMPI node, the developer first calls the Initialize(...) method of the MMPI class. A handle to the MIDlet which is using the MMPI library needs to be passed, as does a Boolean value

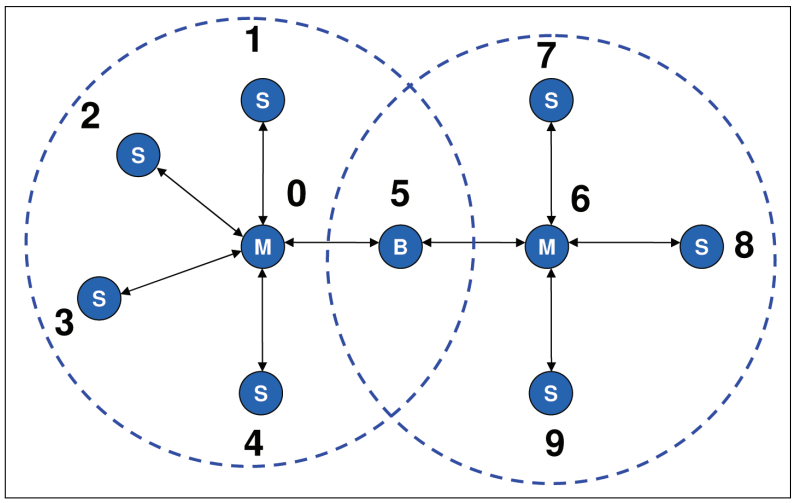

Figure 3: Structure of a ten node scatternet topology

indicating whether the node is the root node. The developer needs to determine how this value will be set themselves, and it should be ensured that all non-root nodes are set up before the root node. This is because calling Initialize(...) activates the Bluetooth device and registers the MMPI service, allowing the node to be discovered. The root nodes reaction to the Initialize(...) call will be to create the network. The non-root nodes reaction will be to wait until this event has occurred and completed. When network formation is complete, the program can continue execution.

3.2 Message Passing Operations

After calling Initialize the communications world has been created thus giving the developer the ability to work with both point to point and global communication methods (Listing 1). Unlike many methods of the original MMPI library (and the MPI libraries), several of the new communications methods return the actual data received as an object. This is quite different to the other libraries that would pass a reference into the method for population with the received data.

```
int Broadcast(Object buffer, int count, int dataType, int root, int tag)
int Scatter(Object sendbuf, int sendcnt, int sendtype, int rcvcnt, int rcvtype, int root)
int Send(Object buffer, int count, int dataType,int dest, int tag)
Object Receive(int count, int dataType, int source, int tag)
Object Gather(Object sendbuf, int sendcnt, int sendtype, int recvcnt, int recvtype, int root ↩
    )
Object Reduce(Object sendbuf, int count, int datatype, int op, int root)
```

Listing 1: MMPI Communication method prototypes

4 Evaluation

To fully test the library one needs a significant number of mobile devices (far in excess of eight). Such resources were unavailable at the time of implementation, but the system was developed and tested using the J2ME emulators of the Sun Wireless Toolkit. The system performed very well on the emulator, and several applications were developed for test purposes, including such classical applications as the Mandelbrot Set.

Using the Mandelbrot Set as an example it takes on average 139,360ms to generate an image of 200^2 pixels at 500 iterations on a single device using the WTK Emulator. This is significantly slower that a real world phone such as the Nokia 6630, capable of generating the same image in 52,344ms. When

multiple instances of the emulator are running and carrying out a complex processing task they do not appear to achieve total parallelism as one would expect. Many of the emulated devices however do give processing times as expected on the order of 13 to 14 seconds, when the application is executed with a set of ten phones.

4.1 Other Applications

Although we have developed the framework mainly to alleviate the node restriction on the existing version of MMPI, it is not limited to this use only. MMPI itself has been used as a communications library for Bluetooth gaming [7] and this framework could be used to increase the number of players that may participate in the game.

5 Conclusion

This paper has outlined a library for creating scatternet based applications, which is capable of parallel computing on adhoc Bluetooth networks of more than eight devices, using the scatternet framework. The structure and operation of the library has been outlined. It was found that there is a performance overhead associated with message routing and parsing. The framework can be used for a myriad of applications, such as multiplayer gaming or chat applications, quite easily. The most important aspect of the framework is that it can be deployed to any mobile device with MIDP 2.0 and Bluetooth functionality, therefore it is capable of running on a significant number of todays mobile devices.

6 Acknowledgment

Development of the MMPI Library was funded under the “Irish Research Council for Science, Engineering and Technology” funded by the “National Development Plan”.

Bibliography

[1] ARM. ARM Cortex-A8, 2005. `http://www.arm.com/products/CPUs/ARM_Cortex-A8.html`.

[2] ARM. ARM Introduces Industry’s Fastest Processor for Low-Power Mobile and Consumer Applications, Oct 2005. `http://www.arm.com/news/10548.html`.

[3] Bluetooth-SIG. Bluetooth Specification Version 1.1.

[4] Bluetooth.com. The Official Bluetooth website. `http://www.bluetooth.com/`.

[5] Bluetooth.org. The Official Bluetooth Membership Site. `http://www.bluetooth.org/`.

[6] D. C. Doolan, S. Tabirca, and L. T. Yang., Mobile Parallel Computing, In *5th International Symposium on Parallel and Distributed Computing (ISPDC06)*, pp 161–167, Timisoara, Romania, July 2006.

[7] K. Duggan, D. C. Doolan, S. Tabirca, and L. T. Yang. Single to Multiplayer Bluetooth Gaming Framework. In *6th International Symposium on Parallel and Distributed Computing (ISPDC07)*, Hagenberg, Austria, July 2007.

[8] ITFacts. 1.019 Billion Mobile Phones Shipped in 2006. `http://www.itfacts.biz/index.php?id=P8049`.

[9] D. Jayanna, G. Zaruba, A Dynamic and Distributed Scatternet Formation Protocol for Real-life Bluetooth Scatternets In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS05)*, 2005.

[10] G. Miklos, A. Racz, Z. Turanyi, A. Valko, and P. Johansson. Performance aspects of Bluetooth scatternet formation. In *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, pages 147–148, 11 Aug. 2000.

[11] MPI. The Message Passing Interface (MPI) Standard. `http://www-unix.mcs.anl.gov/mpi/`.

[12] MPICH. Mpich - Free Implementation of MPI. `http://www-unix.mcs.anl.gov/mpi/mpich/`.

[13] R. Shepherd, J. Story, S. Mansoor, Parallel Computation in Mobile Systems Using Bluetooth Scatternets and Java In *Proceedings of the International Conference on Parallel and Distributed Computing and Networks*, 2004.

[14] Symbian Freak. Nokia 6680 is Loosing the Battle to 6630. `http://www.symbian-freak.com/news/0305/6680.htm`.

[15] G. Zaruba, S. Basagni, I. Chlamtac, Bluetrees - Scatternet Formation to Enable Bluetooth based adhoc Networks In *IEEE International Conference Communications (ICC2001)*, pp 273–277, 2001.

Brendan J. Donegan, Daniel C. Doolan, Sabin Tabirca
University College Cork
Department of Computer Science
College Road, Cork, Ireland
E-mail: d.doolan@cs.ucc.ie, tabirca@cs.ucc.ie
Received: October 18, 2007

Brendan J. Donegan, was a Postgraduate student in the Computer Science Department of University College Cork between 2005-2006, studying Mobile Networking and Computing. He is now working as Graduate Software Engineer for Symbian Software Limited in London. This work is the result of the MSc project titled "Mobile Parallel Computing" he undertook in the Mobile Multimedia group.

Daniel C. Doolan, is currently in the final stages of completing a PhD in the area of Mobile Computing. He holds a BSc in Computer Applications, an MSc in Multimedia Technology, and over half a dozen other degrees at certificate and diploma level in the areas of business and computing. He has authored approximately 40 publications, including 6 book chapters, covering topics such as mobile computing, computer graphics and parallel processing.

Sabin Tabirca is a lecturer in Department of Computer Science of National University of Ireland, Cork. His main research interest is on Mobile and Parallel Computing for Scientific Problems. He has published more than 100 articles in the areas of mobile multimedia, parallel computation, number theory and combinatorial optimization.