THUMBTRIBES: LOW BANDWIDTH, LOCATION-AWARE COMMUNICATION

John P. T. Moore
Zedstar Dot Org Ltd
PO BOX 119, Pinner, HA5 2UR, United Kingdom

Keywords: Mobile computing, location-aware communication, IRC, s-expressions, packedobjects.

Abstract: Mobile application developers need to design their applications with the constraints of the network in mind. In comparison to inexpensive home broadband networks, mobile networks often exhibit high levels of latency and have limited bandwidth. Another significant difference between these networks is the cost of transporting data. Mobile network operators often apply usage tariffs, especially if the customer is on a prepaid service. In such situations, mobile Internet use can prove costly. To help reduce costs, application developers should try to minimise the amount of data their applications communicate. In this paper we present, thumbtribes, an open source project which attempts to add location-awareness to Internet Relay Chat (IRC) in a network efficient manner.

1 INTRODUCTION

Developing applications for mobile devices is not straightforward in comparison to desktop application development (Moore, 2006a). A limited screen size requires clever use of the viewable area. In addition, battery life continues to be an issue for power hungry 3G phones. Today's smart phones are designed to multi-task applications, however, a correlation exists between CPU cycles and battery life. Memory is also in limited supply. Languages such as C are well suited to designing applications within these constraints. Subsequently, C has become the language of choice for embedded developers (Barr and Massa, 2006). However, developers with desktop experience may be more familiar with languages which automate low-level tasks such as memory allocation. Developing for a mobile platform requires knowledge of cross-compilation and its associated tools. Testing and debugging in this environment can also be a challenge.

Even after addressing these issues other stumbling blocks remain. A factor which is often neglected when porting a desktop application to a mobile device is the nature of the network that will be used. Mobile manufacturers continue to release handsets which only support 2.5G technologies such as General Packet Radio Service (GPRS). The limited bandwidth available on these networks prevents the use of certain types of application. The latency on such networks can be remarkably high compared to inexpensive home broadband connections. Round trip times which run into seconds are not uncommon. Significant levels of packet loss can be experienced causing and/or adding to the latency. Despite these limitations, the overriding factor which could hinder mobile Internet use is the cost of transporting data. Mobile phone operators typically charge according to the quantity of data sent across their networks, especially when prepaid services are used. Unless mobile Internet use is made more affordable it is unlikely there will be an explosion in application use. Mobile application developers can help reduce costs by designing their applications in a network efficient manner. The goal of the thumbtribes project is to incorporate location-awareness into Internet Relay Chat (IRC) without incurring a significant cost to a user on a prepaid mobile service. In the thumbtribes system, location-awareness refers to the ability for a user to find out the geographic distance of another user based on some predefined preferences. To help achieve the the main goal of network efficiency, thumbtribes uses packedobjects encoding (Moore, 2006b) .

P. T. Moore J. (2007).
THUMBTRIBES: LOW BANDWIDTH, LOCATION-AWARE COMMUNICATION.
In *Proceedings of the Second International Conference on Wireless Information Networks and Systems*, pages 181-186
DOI: 10.5220/0002151001810186

2 PACKEDOBJECTS

packedobjects is an open source project (Moore, 2007) built with C and an implementation of Scheme known as Chicken (Winkelmann, 2007). Scheme is a multi-paradigm programming language, which due to its small core, is an ideal language for embedding into other languages such as C. A powerful language feature which becomes available with this technique is the ability to use symbolic expressions (s-expressions) (McCarthy, 1960). An s-expression is a syntactic form which allows complex data structures to be represented more concisely than XML. For example, figure 1 represents a personnel record using an s-expression. The same personnel record represented

```
(personnel
 (name
  (givenName "John")
  (initial "P")
  (familyName "Smith"))
 (title "Director")
 (number 51)
 (dateOfHire "19710917")
 (nameOfSpouse
  (givenName "Mary")
  (initial "T")
  (familyName "Smith"))
 (children
  ((name
    (givenName "Ralph")
    (initial "T")
    (familyName "Smith"))
   (dateOfBirth "19571111"))
  ((name
    (givenName "Susan")
    (initial "B")
    (familyName "Jones"))
   (dateOfBirth "19590717"))))
```

Figure 1: S-expression personnel record.

in XML may be up to 44% more verbose, in terms of the number of bytes used. S-expressions are well suited to representing network protocols and have been used in the application domain of cryptography (Rivest and Lampson, 1996) and within email protocols such as IMAP (Crispin, 1988). Even though s-expressions are concise, packedobjects goes one step further and transforms the textual representation of an s-expression into binary before transmitting it over a network.

3 THE NEED FOR EFFICIENCY

Binary protocols have a specific role to play in mobile networks which suffer from problems such as lack of bandwidth, latency and cost of transporting data. To illustrate this we can compare the amount of bytes that would be generated if an s-expression was sent in its textual form with its compressed form. A significant part of the thumbtribes protocol involves sending small amounts of numeric data across the network. The client continually informs the server of its geographic position and the server responds with the distances of other users. This exchange forms the bulk of the traffic in a thumbtribes session. Figure 2 provides an example s-expression request message that might be sent to the server. Due to the size and nature of the message, compression techniques such as gzip (GNU zip) (Deutsch, 1996) are void. Applying such compression actually increases the message size. However, when packedobjects encoding is applied to the s-expression it results in a message of 8 bytes in comparison to 61 bytes in its original form. Before discounting compression techniques such as gzip we need to examine situations which offer more chance of success, such as larger messages. Unless there is significant improvement, the increase of CPU time of such techniques cannot be justified. To test this we can examine the response from the location request which may consist of a sequence of up to and including 50 user names and distances. Figure 3 provides an example of an s-expression response message. Figure 4 compares the bytes generated for gzip compression, packedobjects encoding and raw s-expressions for response messages containing distances for 0 to 50 users. User names and distances have been generated randomly for convenience. In reality the user names, although unique, would exhibit less random-

```
(pdu
  (ping-request
    (latitude 51518928)
    (longitude -151668)))
```

Figure 2: Request message.

```
(pdu
 (ping-response
  ((nick "sdgmokm2") (distance 16485828))
  ((nick "r9ubm7v") (distance 14708129))
  ((nick "z5gp950rb") (distance 12899883))
  ((nick "yci") (distance 10090589))
  ((nick "sr") (distance 18918800))))
```

Figure 3: Response message.

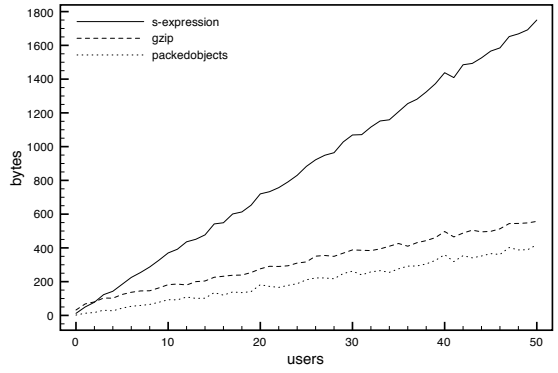

Figure 4: Complete range of location response messages.

ness and therefore may offer different levels of compression. The results show that both gzip compression and packedobjects encoding are able to significantly reduce the amount of bytes generated compared to when using text-based s-expressions. However, on average, packedobjects encoding is able to offer a 36% reduction in message size over gzip compression. Although the results depict the maximum range of possible message sizes, in reality, typical message sizes would be more accurately represented by figure 5 where message sizes range from 0 to 10 users. We

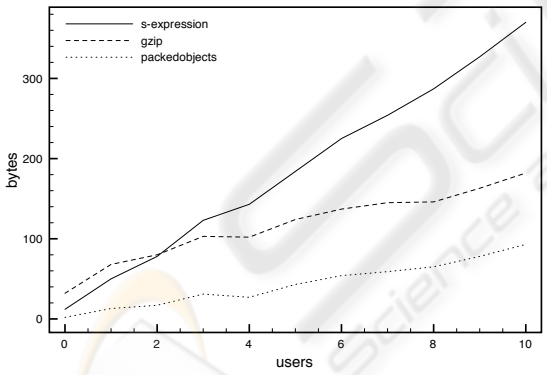

Figure 5: Typical range of location response messages.

can now clearly see that it takes a response message of at least 3 users for gzip compression to produce less bytes than in its uncompressed form. packedobjects now produces on average a 62% reduction in message size over gzip. The significance of these savings depends on several factors, such as the frequency of a request message, the period of time examined, the size of a response message, and ultimately the cost of sending data over the network. To highlight this we can simulate a request/response cycle and then apply a cost metric to the data to show how cumulative costs may vary between both methods of compression. Table 1 shows the cumulative costs, in terms of

Table 1: Cumulative costs in UKP.

hours	s-expr	gzip	packedobjects
1	0.68	0.54	0.14
2	1.29	1.06	0.26
3	1.93	1.58	0.39
4	2.58	2.11	0.53
5	3.2	2.63	0.65
6	3.85	3.15	0.78
7	4.5	3.68	0.91

UK pounds (UKP), when a location request is made every 10 seconds for a period of 7 hours. A response is generated containing a random number of users and their distances. As with figure 5, the number of users is limited to 10 and names and distances were generated at random. A cost metric of 0.0075UKP per kilobyte of data has been applied. This reflects the current tariff on a prepaid service from a leading mobile operator in the UK. Typical use of thumbtribes would involve leaving a client running in the background while other applications run. The results show, after 7 hours, gzip compression would have incurred an extra 2.77UKP over the cost of using packedobjects encoding.

4 THE THUMBTRIBES ARCHITECTURE

The thumbtribes architecture can be divided into three sections: clients, servers and backend, as depicted in figure 6. Three client/server protocols are employed: HTTP (web), IRC and thumbtribes. A web interface is provided to allow users to alter preferences in the thumbtribes system in real time. Altering preferences to restrict the amount of data the thumbtribes server encodes is an important method of reducing usage costs on a metered network. For example, the user may wish to limit responses to users who are within a certain proximity. Using a web client to make these changes eliminates the need to enter data via a mobile handset.

IRC is used to support all user-to-user communication as well as authenticate and create users and nicknames. IRC is a lightweight and simple text based protocol (Oikarinen and Reed, 1993) designed mainly for group communication. The system has been around since 1998 and is well supported in terms of users, servers and choices of client software. The

IRC server supports various server-sanctioned bots [1] for registering and controlling access to user names, nicknames and various other features. This forms the basis of authenticating the user via the web client and the thumbtribes client. Using IRC services for user authentication provides a simple pre-built way to handle accounts across the entire thumbtribes system. The data associated with these server-sanctioned bots is provided by the backend. The backend consists of a Relational Database Management System (RDBMS). All communication between the backend and the three servers takes place via SQL commands over a TCP/IP connection. The separation of the backend from the servers means that all 4 components could reside on physically different machines.

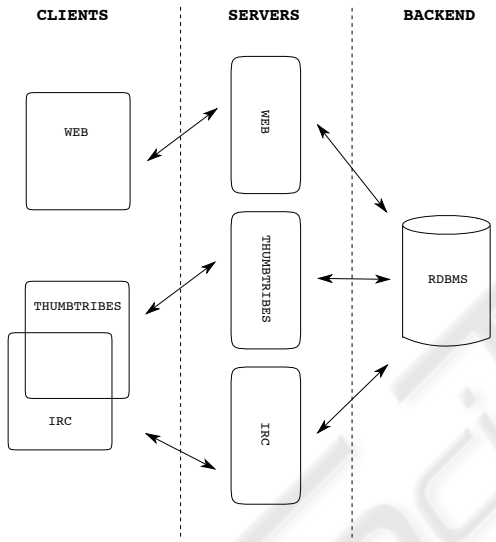

Figure 6: Thumbtribes architecture.

The thumbtribes client is responsible for all location related communication and has been depicted as a rectangle within a rectangle to highlight how the software can be embedded into an existing IRC client. The thumbtribes server calculates information such as the distance of users and filters responses according to preferences stored in the backend. The main purpose of the thumbtribes client is to continually update, or ”ping”, the server with coordinates. The way this location information is supplied is not part of the thumbtribes architecture. One method, for example, which is independent of any network operators is to obtain coordinates from a GPS (Global Positioning System) device. As GPS technology improves and variants such as AGPS (Assisted GPS) (LaMance et al., 2002) become more widespread, this solution becomes more usable.

[1]A bot is an independent program which connects to IRC as a client but appears as a normal IRC user.

5 THE THUMBTRIBES PROTOCOL

thumbtribes uses a simple client/server protocol over a TCP connection. There is an exchange of Protocol Data Units (PDUs) as summarised by the timing graph in figure 7. The sequential request/response nature of the protocol means that the same area of allocated memory can be used for both encoding and decoding of PDUs. This saving should be welcome on an embedded device such as a mobile phone. Figure 8 provides the complete thumbtribes protocol specification. The protocol is described using an s-expression

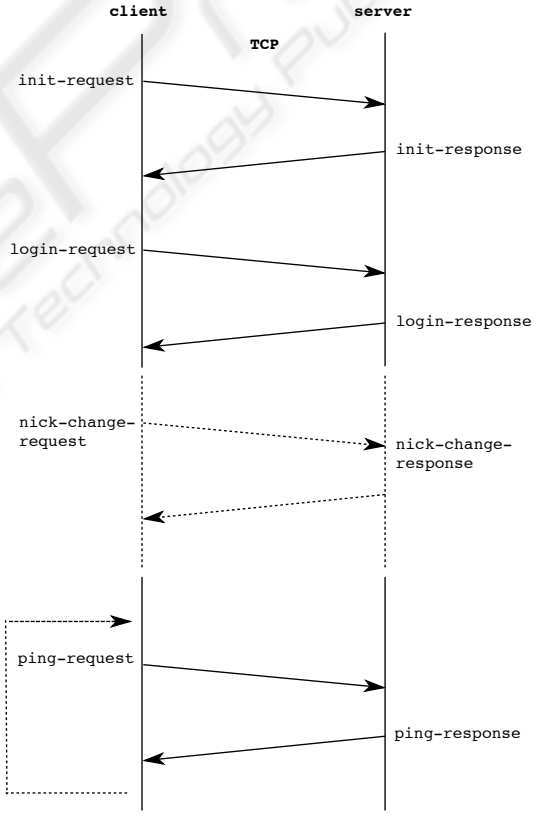

Figure 7: Protocol timing graph.

and makes use of features of the Scheme programming language such as comments and ”quasiquoting”. All comments are represented by lines beginning with a semicolon. A quasiquote is used to define

```
(define protocol
  ;; user defined type
  (let ((nick '(nick string (size 1 9))))
    ;; start of PDU
    `(pdu choice
        ;; inform server of version and machine type
        (init-request sequence
                  (machine string (size 1 20))
                  (version integer (range 1 1000)))
        ;; returns true if ok
        (init-response boolean)

        ;; identify user to server
        (login-request sequence
                   (username string (size 1 10))
                   (password string (size 1 35))
                   ,nick)
        ;; server responds with ping time or error code
        ;;  0  = password incorrect
        ;; -1  = no such username
        ;; -2  = no such nick
        ;; -3  = already logged in
        (login-response integer (range -10 ,MAX-PING-TIME))

        ;; inform server of nick change
        (nick-change-request sequence ,nick)
        ;; returns true if ok
        (nick-change-response boolean)

        ;; send coordinates to server
        (ping-request sequence
                  (latitude integer (range -90000000 90000000))
                  (longitude integer (range -180000000 180000000)))
        ;; server responds with distance of users
        (ping-response sequence-of ; max 50
                   ,nick
                   (distance integer (range 0 20000000))))))
```

Figure 8: Thumbtribes protocol specification.

the type ”nick” which can then be used multiple times throughout the protocol specification. The ability to represent the protocol as code means that no further transformation is required to manipulate the data. If we contrast this to XML, the protocol would need to be transformed from its description into an abstract data type, such as a tree, before it could be manipulated.

The thumbtribes protocol is a choice of 8 PDUs exchanged in 4 sequences: init, login, nick-change and ping. The init sequence informs the server of the client machine type and more importantly the version of software being used. If the client is using a compatible version, the server will respond with a boolean flag set to true. On receiving this response the client initiates a login sequence to the server. The server then authenticates the user and will respond in one of two ways. On successful authentication a value will be retrieved which will represent the amount of time the client will pause during a ping-request/ping-response cycle. Failure to authenticate will return a value less than 1 depending on the error. Comments in the protocol specification indicate the various errors that might occur. Overloading a data type in this way is not recommended in most protocol specifications, however, in this situation it helps simplify the response message. After the login stage completes, the two remaining sequences may take place. The nick-change sequence is represented as a dashed line in figure 7 to indicate it is optional. In the thumbtribes system a user may have multiple nicknames. If the user changes nickname during a session the server needs to be informed. This requires sending the new nickname to the server and the server will respond with a boolean flag to indicate success or failure depending on whether the nickname is registered to the user. The remaining sequence to describe is the most fundamental, the ping. As indicated in figure 7 this sequence may repeat. The client sends a ping-request to the server containing latitude and longitude coordinates. Each coordinate is expressed in integer form with its decimal equivalent represented as $coordinate \times 10^{-6}$. Therefore, coordinates can be expressed to an accuracy of 6 decimal places. On receiving the coordinates the server is able to calculate the distances of other users in the thumbtribes system. The ping-response contains a sequence of nicknames and corresponding distances expressed in meters. The ping sequence will then repeat until the user decides to terminate the thumbtribes session. Termination is initiated when the user quits the client and is detected by closure of the TCP connection.

6 FUTURE WORK

One of the benefits of working with s-expressions in Scheme is they are a native data structure. In comparison, XML requires a library or toolkit to transfer the textual representation into an abstract data type supported by the high level language being used. Several such toolkits exist depending on the high level language. An interesting study would be to measure the amount of processor time and memory required to process XML compared to s-expressions. Both metrics are important for embedded devices such as mobile phones. However, trying to obtain ”apples vs apples” metrics for such studies is not straightforward. In section 3, we indicated the savings that could be made using packedobjects encoding instead of gzip compression of s-expressions. Many protocols, including popular instant messaging protocols such as Jabber (Saint-Andre, 2004), are still being sent over mobile networks as uncompressed XML. A cost comparison could be made against a packedobjects encoded equivalent protocol.

In terms of application enhancements, thumbtribes could benefit from numerous additions. One such addition could be a location-oriented event system. The user would be able to enter details of an event and supply coordinates. Other users would then be able to pick up these events when in a defined proximity. Tagged locations could be another useful thumbtribes addition. Here users could walk around an area and tag locations with a short description. This information could also then be available to other local users. Many similar location-aware features could be implemented, all of which benefit from the low-bandwidth protocol employed in the thumbtribes system. The current development version of thumbtribes is being built as a plugin to an open source IRC client. Rather than embed thumbtribes functionality, a special purpose client could be created. The ideal client would mask the fact IRC is being used as the underlying communication protocol. This would allow a more simplistic and user-friendly interface for mobile phones to be built.

7 CONCLUSION

Users have become accustomed to a certain level of quality of service from the broadband connections they use at home. In contrast to a typical broadband connection, a mobile network is likely to have less bandwidth as well as exhibit higher packet loss in conjunction with increased round trip times. The mobile network operator might also apply a usage tariff. Often when applications are ported from a desktop environment to mobile environment they will lack consideration for these issues. In addition, it is often more straightforward for mobile application developers to design network protocols based on popular standards such as XML. Unfortunately such standards can be verbose, even after applying well known compression techniques. The thumbtribes project demonstrated this by comparing gzip compression with packedobjects encoding. The bulk of the traffic generated during a thumbtribes session occurs during its location request/response cycle. Here the client periodically sends a small message to the server containing geographic coordinates. Tests showed that while packedobjects encoding was able to offer significant levels of compression, gzip compression actually increased the size of the original message. When larger message sizes were examined, packedobjects was able to reduce typical message sizes on average by 62% compared to when gzip was used. The significance of these savings depends on usage. However, with today's multi-tasking phones, it is not an uncommon requirement to be able to leave an application running in the background. In such circumstances, it is easy to show why reducing costs on a metered network is important.

REFERENCES

Barr, M. and Massa, A. (2006). *Programming Embedded Systems*. O'Reilly.

Crispin, M. (1988). Interactive Mail Access Protocol: Version 2, Request for Comments: 1064. http://www.ietf.org/rfc/rfc1064.txt.

Deutsch, P. (1996). GZIP file format specification version 4.3, Request for Comments: 1952. http://www.ietf.org/rfc/rfc1952.txt.

LaMance, J., DeSalas, J., and Jarvinen, J. (2002). Assisted GPS: A Low-Infrastructure Approach. http://www.gpsworld.com/gpsworld/article/articleDetail.jsp?id=12287.

McCarthy, J. (1960). Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. *Communications of the ACM*, 3(4):184–195.

Moore, J. (2006a). Developing on handhelds, University of Birmingham. http://www.cs.bham.ac.uk/internal/courses/comm-prog/slides/moore.pdf.

Moore, J. (2006b). packedobjects. In *WINSYS 2006 International Conference on Wireless Information Networks and Systems*.

Moore, J. (2007). packedobjects. http://packedobjects.com.

Oikarinen, J. and Reed, D. (1993). Internet Relay Chat Protocol, Request for Comments: 1459. http://www.ietf.org/rfc/rfc1459.txt.

Rivest, R. and Lampson, B. (1996). SDSI - A Simple Distributed Security Infrastructure. http://theory.lcs.mit.edu/ rivest/sdsi10.html.

Saint-Andre, P. (2004). Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence, Request for Comments: 3921. http://www.ietf.org/rfc/rfc3921.txt.

Winkelmann, F. (2007). Chicken Scheme. http://www.call-with-current-continuation.org/.