Positioning Simulator for Location-based Service

Anusuriya Devaraju
Faculty of Information and Communication Technology,
Kolej Universiti Teknikal Kebangsaan Malaysia,
Melaka, Malaysia.
anusuriya@kutkm.edu.my

Simon Beddus
British Telecommunications (BT) Research and Venturing,
Adastral Park, Martlesham Heath,
Ipswich, UK.
simon.beddus@bt.com

***Abstract* - Location Based Services or LBS, refers to the wireless services provided to the subscriber based on his current location. The position can be known by getting it from mobile phone network, or from another positioning service, such as GPS. Mobile location-based system is complex and requires the integration of many different technology components into one seamless system. This paper describes on the design and implementation of the three location-based system components, which are the location server simulator, location service application and device client application. The location server simulator processes the location request and returns simulated positioning data to the location service application. The location service application is aimed at providing mobile users the map of their current position based on the positioning data from the simulator. It senses what the display capabilities the requesting device has and delivers the map accordingly. The last component is a small software entity that runs on Java phones such as Motorola A835, Nokia 7610, and SonyEricssonP900. This component is called device client application, issues a location request that starts the data flow and displays the map or text messages on mobile phones.**

***Keywords* – Location Based Service, Location Server Simulator, Mobile Location Protocol, J2ME**

I. INTRODUCTION

This paper describes the Location-based Service project, which is carried out during industrial placement with BT Research and Venturing. The project has focused on the design delivery and testing of location-based system components. This paper is organized as follows: The Section I introduces a discussion of the problem leading to the purpose of the study. The Section II gives an overview on the most important concepts and technologies. The Section III classifies requirements of the system and provides a details explanation on how every component in the system is designed and developed. In Section IV, the future and open work items are clarified and conclusions are presented.

A. Background

The prototype Wireless API service gateway, code-named 'Erica', is developed by BT to provide a middleware service access point to the network infrastructure enabling the operators to build and deploy wide range of wireless Application Programming Interfaces (API) services and management capabilities [2]. The platform supports a number of API technologies one of which is User Location Service (ULS). ULS is designed to provide a single location API that is independent of location technology or location provider. Location feeds include Global System for Mobile Communications (GSM), Code Division Multiple Access (CDMA), 802.11 and Global Positioning System (GPS) technologies.

B. Problem Discussions

Erica's ULS interface enables remote location application to be validated in a test environment prior to connecting to the live network operators such as *O2, T-Mobile, Vodafone* and so forth. Testing each application with real time positioning data from the network operator is charged per lookup. Therefore, this project introduces a simulator based on Location Interoperability Forum (LIF) API to remove the necessity for testing against live and charged services. Also, an admin website has been developed to enable the administrator to manage simulated positioning data easily.

Normal web application serving location maps cannot be displayed properly on mobile phones because of the small screens. Apart from this, mobile phones have different limited resources in terms of processor speed, power management, memory size, media and colour support. The processing power of mobile phones to render web content is low; the bandwidth available for transferring data is low; data volumes are costly; connections may be lost; and the latency which is the time it takes for the data to be sent from a server and received by a mobile phone can be high [4]. Although latest mobile phone models are equipped with colour displays and powerful CPUs, but it is impossible to implement location information service with advanced functionality and graphical presentation on mobile phones, as in desktop computers. Due to complexity of the user interfaces, ordinary web applications are difficult to use on a mobile phone, whereas specific applications such as .NET Compact Framework and J2ME implement customised widgets and event handling, opening up unlimited possibilities for mobile client interfaces that are easier and more interesting to use.

II. SUPPORTING TECHNOLOGIES

To understand how developing location-based services differs from traditional wire line applications, it is important to understand the basic principles of underlying technologies.

A. Positioning Technologies

There are four mobile systems that are used globally, GSM, Wideband Code Division Multiple Access (WCDMA), CDMA and Time Division Multiple Access (TDMA). There is another system called Personal Digital Cellular (PDC), which is only used in Japan [5]. The GSM is a digital cellular radio network. While the current GSM system was originally designed with an emphasis on voice sessions, the main objective of General Packet Radio Service (GPRS) is to offer an access to public data networks such as Transport Control Protocol/Internet Protocol (TCP/IP) and X.25 directly when the mobile phone is attached to the GPRS network. In GPRS system, the call charging could solely based on amount of transmitted data.

B. Location API and Protocol

There are three main methods to get position information from network today - LIF API, WAP Location Framework and Parlay API. The location server simulator in this project is based on LIF API. The LIF has produced Mobile Location Protocol (MLP) specification, which is an application-level protocol for positioning mobile terminal. The MLP serves as the interface between a location server and a location-based application, which defines the core set of operations that a location server should be able to perform.

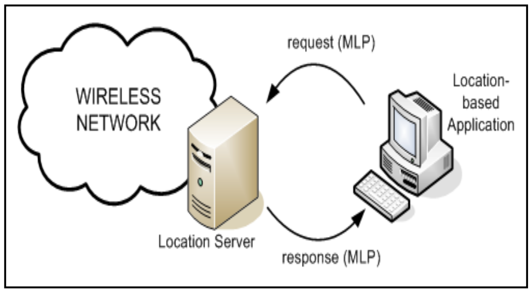


Fig. 1. How the LIF API works

C. Sun's Java 2 Micro Edition (J2ME)

The J2ME technology consists of a set of core Java APIs and virtual machine for tailored runtime environments for resource-constrained devices such as handphones, PDAs and set-top boxes. It provides Java based APIs for local persistence, network connectivity, and user interface that when all combined allows for standalone, mission critical applications not possible with thin clients. Based on Fig. 2, the Mobile Information Device Profile (MIDP), when combined with the Connected Limited Device Configuration (CLDC), provides a basic J2ME runtime environment targeted at small, standalone or connectable consumer and embedded devices.

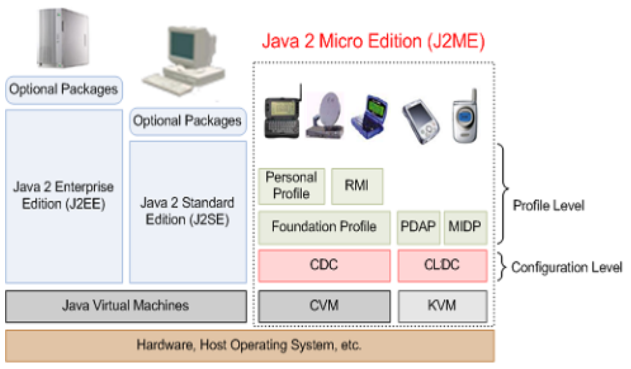


Fig. 2. J2ME runtime environment

D. J2ME versus Competing Technologies

Thin client applications are based on mark-up languages such as Wireless Markup Language (WML), HTML, and Compact HTML (CHTML) for content generation and interactions. These applications are relatively easy to create, however to work they require that the user be connected, online, or in coverage. In a thin client solution, the server is responsible for generating displays mark-up. This requires a round-trip every time the interface changes. In contrast, a J2ME MIDP client is contained within the device, so it can operate even when disconnected. On occasions when the devices do interact with the server, it incurs less network traffic because it downloads only application data; as opposed to application data plus interface mark-up.

Native applications are developed for a particular operating system such as PalmOS or Symbian and are traditionally written in C or C++ language. These applications can work in an offline environment but are completely dependent on the operating system of the device and requires a higher level of expertise because of the complexity of the user interface, communication models, and the wide variations in OS capabilities. However, native applications can take full advantage of inherent featured such as hardware management and interaction because they are tightly coupled to a particular operating system. Compared with the native platforms, the main strength of the Java platform is that it allows us to write portable applications easily [14].

III. Overall System Architecture

Fig. 3 illustrates the interactions within components in the system. Java Server Pages (JSP) and Java Servlet are used to develop the simulator and location service application, while Tomcat server as the application server.

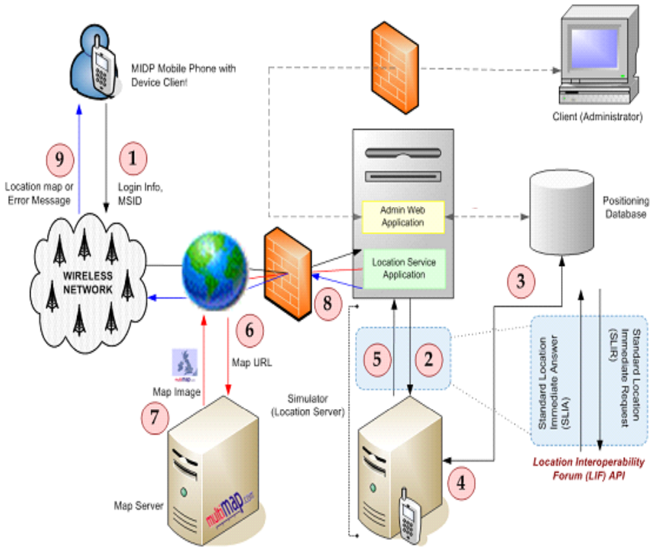

MIDP Mobile Phone with Device Client
Client (Administrator)
9
1
Location map or Error Message
Login Info, MSID
Admin Web Application
Positioning Database
WIRELESS NETWORK
Location Service Application
8
3
6
Map URL
5
2
Standard Location Immediate Answer (SLIA)
Standard Location Immediate Request (SLIR)
Map Image
Simulator (Location Server)
7
4
Location Interoperability Forum (LIF) API
Map Server

Fig. 3. Interaction within components in the system

The following steps describe the interaction between components in the system.

1. The device client application enables mobile subscriber to send an HTTP request to location service application via GPRS wireless network.
2. The service application identifies the subscriber and the service requested by the subscriber. The interface between the simulator and location service application is modeled after LIF's MLP. The location service

application formats the request data as Standard Location Immediate Request (SLIR) message and sends it via HTTP to the location server simulator.
3. The simulator validates application provider's identity and request format.
4. Then it retrieves the relevant positioning data from the positioning database (MySQL) based on mobile number and request time.
5. The simulator constructs a Standard Location Immediate Answer (SLIA) message, which consists of the positioning data and other supporting location elements. Then, it returns the SLIA message to the location service application.
6. The service application parses the SLIA message to get the positioning data. Then, it binds the latitude and longitude in map URL, and opens a connection to the Multimap server. Here, there can be another case where the simulator returns error message, not the positioning data on SLIA. The service will not send a map request to the map server, and instead it will report the error message directly to the device client application.
7. The Multimap server sends the map to the location service application based on the latitude and longitude.
8. The service application resizes and changes the format of the image, and then sends the processed map to the user.
9. The J2ME device application installed on mobile terminal allows the user to view the map and interact with other function like zooming. The next actions the subscriber is likely to ask for his/her current location map to a different scale.

A. The Design and Implementation of Device Application

The J2ME Wireless Toolkit is used to develop the device client application. The toolkit provides a simple path from source code to develop and execute MIDlets. MIDlets are the client applications installed on mobile phone. This toolkit includes a Graphical User Interface tool that automates some of the tedious details of building and packages MIDlets [7]. The device application supports basic operations such as map visualisation, selective zooming and progress indicator to let the mobile users know that the application is working.

Making Connection with HTTP. The device application is based on MIDP 1.0 and in MIDP 1.0 the only protocol guaranteed to be implemented is HTTP. The idea of having MIDP supports the HTTP protocol very clear. For network programming, developer can revert to the HTTP programming model, and the applications will run on any MIDP devices, whether it is a GSM phone with a WAP stack, a phone with I-mode, a Palm VII wireless, or handheld device with Bluetooth [10].

Why not XML or SOAP? The kSOAP[12] and kXML[3] are packages designed to enable SOAP and XML applications to run within a KVM - a low-memory virtual machine running on a micro device. However, these are not implemented on device client application because data exchange via SOAP and XML require implementation on both client and server side. Generating and parsing XML documents is a time consuming task and needs a substantial amount of CPU compared to passing data as part of URL. The complex schemas or Document Type Definitions (DTDs) slow down the processing of the content of the document even more.

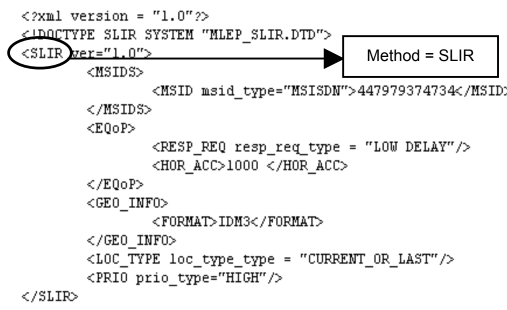
B. The Design and Implementation of Location Service Application

A location-based service application is introduced as part of the test and evaluation process for the simulator.

XML over HTTP Request. According to the LIF API, the protocols that can to be used for communication between the location service application and location server are the SOAP and XML [9]. This project has implemented communication based on XML. The location service application requests the positioning data by sending SLIR message via HTTP to the simulator. The request must include the entity-header Content-length field as part of the request. The message body of the request includes the XML formatted request, which consists of three HTTP post parameters:

i. *Context* - This will contain the authentication details of the Application Service Provider.
ii. *Method* – This will identify the method that is to be invoked. It is equal to the 'Name' of the DOCTYPE declaration in the body and therefore identical to the root element of the request, too.
iii. *Body* – This will contain the XML payload for the specified method.

Header and body have to correspond with XML and have to be url-encoded. Fig. 4 shows the XML Fragment of SLIR's *Body* constructed by the location service application.



```
<?xml version = "1.0"?>
<!DOCTYPE SLIR SYSTEM "MLEP_SLIR.DTD">
<SLIR ver="1.0">
        <MSIDS>
                <MSID msid_type="MSISDN">447979374734</MSID>
        </MSIDS>
        <EQoP>
                <RESP_REQ resp_req_type = "LOW DELAY"/>
                <HOR_ACC>1000 </HOR_ACC>
        </EQoP>
        <GEO_INFO>
                <FORMAT>IDM3</FORMAT>
        </GEO_INFO>
        <LOC_TYPE loc_type_type = "CURRENT_OR_LAST"/>
        <PRIO prio_type="HIGH"/>
</SLIR>
```

Fig. 4. XML fragment of SLIR Body

XML Parsing. The simulator returns response as SLIA message. When the service application receives a SLIA message, it retrieves the positioning data by parsing the XML document. Parsing XML means retrieving data from an XML document based on its meaning and structure. The XML parser used in this project is Apache Xerces for Java [1]. The method that is used to parse the request-response messages is *Simple API for XML* or SAX. The parser reads the SLIA contents line by line and fires events that contain information about the line that was just read. User defined Java program that listens to a particular events will extract the location data from the XML document. XMLBeans[13], an XML-Java binding tool, is also used to create instances of SLIR and SLIA documents because SAX (even though faster) can be more complicated for writing XML from scratch.

Map Request and Processing. After parsing the SLIA message, the service application includes the positioning data in a map URL. The format of map URL is as specified in Multimap In-line Maps Implementation Guidelines. The map file is read from the URL. The Multimap server returns a map in *Graphics Interchange Format* (GIF) format. Here, image format conversion is necessary, as the MIDP specification only requires support for the *Portable Network Graphics* (PNG) format. PNG provides a patent-free replacement for GIF. PNG images are compressed with a loss-less algorithm. The algorithm is identical to the algorithm used for JAR files, so the MIDP implementation can save space by using the same algorithm for both purposes [8]. The location service application converts and resizes the image instead of the device client application because it makes the client smaller and likely to be much faster. Images already in PNG format can benefit from being funneled through a service, because the service can adapt the image to fit the device's characteristics. The service application gets the display size of a mobile phone from the database based on 'User-Agent' parameter supplied by the device on request header. The 'User-Agent' parameter contains the mobile phone model name. This value has been programmed into the MIDlet before installing it on the mobile phone.

Caching Previous Location Lookups. The mobile number, positioning data and time of response of each successful reply will be stored temporarily by the location service application. When the service application receives a request for same mobile number, the application compares the current request time with the 'cached' time. If the time difference is less than the specified minutes, the application will retrieve the cached positioning data and send a new map request to map server, else, a new SLIR request will be sent as usual to the simulator. This time-checking feature enables user to get faster response and saves processing on the server.

C. The Design and Implementation of Location Server Simulator

The simulator developed in this project conforms to standard operations in *Basic Location Services* as specified in LIF's specification. Among the services under the Basic Location Services category, the simulator supports the *Standard Location Immediate Service (SLIS)*. SLIS is the standard service for requesting the location of at least one subscriber. This service consists of SLIR and SLIA messages. SLIS is used if the location service application wants to receive the location response immediately (within a set time) or the request may be served by several asynchronous location responses (until a predefined timeout limit is reached) [9].

Positioning data. There are maximum six sets of latitudes and longitudes per mobile number, which are stored in the positioning database. Each mobile number has 6 unique position ids, which values from 0 to 5. Position id represents the seconds range, for instance the range from 0 to 9 seconds has position id '0', 10 to 19 seconds has position id '1' and so on. Thus, different positioning data will be returned for the same mobile number for each 10 seconds.

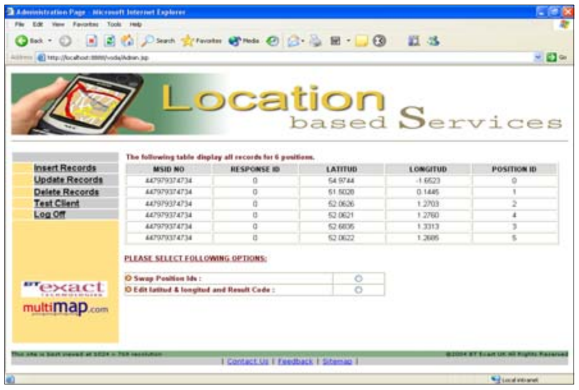
How Does the Simulator Work? When a SLIR message is sent by the location service application to the simulator, the simulator checks the format or method of request message. If the format is correct, it performs XML parsing to extract the request values. Based on these values, the simulator will evaluate the privacy settings of the application provider on the request header. If the privacy settings will not allow localizing, an error will be returned with the SLIA message. If the authentication is successful, the simulator connects to the positioning database and retrieves the relevant positioning data. This is done by getting the current time and extracts the seconds. From the seconds, the simulator assigns the relevant position id for that request. This id and the mobile number will be used to get latitude and longitude from the database. The simulator also calculates other response elements such as radius, local time and GMT. Finally, it constructs a SLIA message, consists of positioning data and other response elements' values. It classifies the MIME type as 'text/xml' and writes the SLIA contents into opened connection from the application service provider.

Error Handling. Apart from retrieving positioning data from the database, the simulator also performs error handling concerning unknown and incorrect requests, localization, authentication and authorization. If an error occurs while processing the request, the simulator will send a SLIA error message, which contains error code, error title and description as specified in MLP documentation.

D. Admin Website

An admin web application is built to enable the administrator to view, insert, edit, delete and swap positioning data quickly and easily. The website is developed by using JSP, JavaScript, and HTML.

IV. PUTTING THE PIECES TOGETHER



The following table display all records for 6 positions.

MSID NO	RESPONSE ID	LATITUD	LONGITUD	POSITION ID
447979374734	0	54.9744	-1.6523	0
447979374734	0	51.5028	0.1445	1
447979374734	0	52.0626	1.2703	2
447979374734	0	52.0621	1.2760	4
447979374734	0	52.6835	1.3313	3
447979374734	0	52.0622	1.2695	5

PLEASE SELECT FOLLOWING OPTIONS:

Swap Position Ids :	○
Edit latitud & longitud and Result Code :	○

Fig. 5. Admin website

Once the device client has been tested on a J2ME emulator, the next step is to package the application and associated components into a Java Archive (JAR) and Java Application Descriptor (JAD) file pair using J2ME Wireless Toolkit. The MIDlet suites can be installed on mobile phone from computer via a serial cable, or transfer them over a wireless network, *Over–The-Air (OTA).*

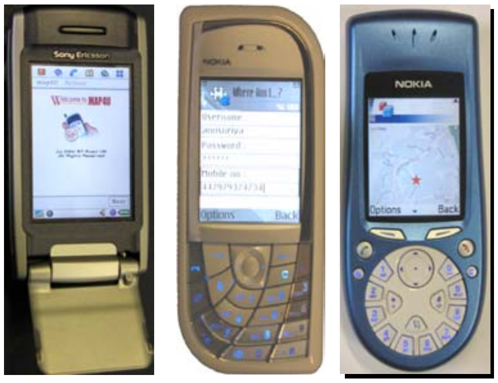


Fig. 6. Screenshots of device client application

V. Future Works And Conclusions

A. Location Server Simulator

Currently, the simulator only supports SLIS from LIF. There are a number of different possible types of LIF services can be implemented such as Emergency Location Immediate Service, Emergency Location Reporting Service and Triggered Location Reporting Service.

For security reasons, the interface between the simulator and location service application should be modelled after secure version of HTTP (HTTPS). Simple Object Access Protocol or SOAP is the preferred interface protocol because of the limitations of XML over HTTPS. Firstly it is independent from the transport protocol like HTTP(S) and it can be used as an underlying transport for distributed object communication such as Common Object Request-Broker Architecture (CORBA) and Distributed-Component Object Model (DCOM). SOAP also permits the use of formal fault handling. None of these points is easily supportable using XML documents over HTTPS. Utilizing this by transporting XML documents using SOAP realizes the best of both technologies and eliminates most of the problems.

The simulator should support more error handling concerning other possible errors such as the request cannot be handled due to congestion in the positioning database, too many position items have been specified in the request, account is suspended when an application tries to connect more than three times with a wrong password, etc.

B. Location Service Application

The location service application can provide other different types of location-based services such as

i. Friend Finder application offers subscribers the ability to see the location of their friends, family members or co-workers quickly and easily using their mobile phones. Mobile user need permission from his/her friend before can check the location.
ii. Business Connect allows subscribers to search for business listings closest to their current location by category, name, or choosing from a list of favorites. Once user has selected a Point of Interest (POI), they can get either directions or a map based on their current location to the POI.
iii. Navigating through a new city, or even a familiar one, can be challenging. Getting accurate directions saves time, petrol, and effort.

C. Device Client

The current MIDP 1.0 specification does not provide enough in the way of security APIs. But with the help of third-party libraries and vendor support, the MIDP 1.0 can be used to implement secure applications. Developers at Sun have created ‘Kilobyte’ SSL or KSSL, a ‘small-footprint, client-side-only implementation of SSL v3.0 for handheld and wireless devices’ [11]. The Phaos Micro Foundation toolkit was built for both the CDC and CLDC environments and is a 100% pure Java solution. Finally, the Bouncy Castle is an open source project for Lightweight Cryptography APIs for J2ME.

D. Conclusions

This project has developed three location-based system components for prototype ERICA Mobile Application Platform. The simulator conforms to standard operations as specified in LIF API. The location service application is built to prove that the simulator works well with other third party provider’s applications in future. The J2ME device application is designed to test the whole system with real mobile device on live wireless network. To conclude, this project has identified appropriate communication mechanism for delivering location data to the new generation of smart phones such as Symbian, J2ME and others. Besides that, it has also defined the issues, barriers and opportunities to the deployment of device based location service applications.

References

[1] Apache Xerces Parser; http://xml.apache.org/xerces2-j/
[2] BT Exact Technologies, “Erica Wireless API service Gateway”, British Telecommunications plc, UK, 2002.
[3] Cadena R, “KXML: A Great Find for XML Parsing in J2ME”, Article on DevX.com, 2003; http://www.devx.com/xml/Article/11773/0
[4] Gibbs M, “Getting Mobile: Using WML and WAP to Display Web Sites on Mobile Devices”, FrontPage 2002 Technical Articles, MSDN Home, 2002.
[5] Hjelm J, “*Creating Location Services For The Wireless Web*”, John Wiley & Sons, Inc, 2002.
[6] Jagoe A, “*Mobile Location Services- The Definitive Guide*”, Prentice Hall PTR, 2002.
[7] Knudsen J, “*Wireless Java: Developing with J2ME*”, 2nd Edition, Apress, 2003.
[8] Kroll M and Haustein S, “*Java 2 Micro Edition (J2ME) Application Development*”, Pearson Education, 2002.
[9] Location Interoperability Forum, “Mobile Location Protocol Specification”, 2002; http://www.openmobilealliance.org/tech/affiliates/lif/lifindex.html
[10] Mahmoud Q, “*Learning Wireless Java*”, 2nd edition, O'Reilly, 2002.
[11] Mahmoud Q, “Secure Java MIDP Programming Using HTTPS with MIDP”, 2002; http://developers.sun.com/techtopics/mobility/midp/articles/https/
[12] McHugh J, “Low Bandwidth SOAP”, Article on XML.com, 2003; http://webservices.xml.com/pub/a/ws/2003/08/19/ksoap.html
[13] XMLBeans; http://xmlbeans.apache.org
[14] C.Enrique Ortiz , “Elements of a Typical J2ME MIDP Business Application”, 2001; http://www.microjava.com/articles/techtalk/midp