TR03-15

December 5, 2001

SENSE: A Toolkit for Stick-e Frameworks

Siva Kollipara, Rohit Sah,
Srinivasan Badrinarayanan, Rabee Alshemali

Supervisor: Dr. Nigel Davies

SENSE: A Toolkit for Stick – e Frameworks

Siva Kollipara, Rohit Sah, Srinivasan Badrinarayanan, Rabee Alshemali
Computer Science Department
The University of Arizona
{ siva, rsah, srinivas, alshemal } @ cs.arizona.edu
December 5, 2001

Abstract

The increase in the number of handheld devices and active research in the field of Context Aware Computing has led to a growing interest in the development of platforms and toolkits for context aware applications. In this paper we present the experiences in designing and implementing SENSE: An e-note toolkit for context aware applications. SENSE provides a platform to develop applications like campus guides that are both passive and active context aware.

Key words: Context Aware computing, mobile computing, ubiquitous computing, Stick-E notes, Event Based Framework, XML, JAVA, Disconnected operations

1. Introduction

Toolkits provide a platform to build many powerful applications. With the increase in the number of handheld devices and active research in the field of Context Aware Computing there has been a growing interest in the development of platforms and toolkits for context aware applications. The Stick-E framework as proposed by P.J. Brown [1] provides an effectual abstraction to develop context aware applications.

In this paper we present the design and implementation of SENSE: A Context Aware Toolkit for use in a wireless environment. Using SENSE a user with a handheld device like a PDA can place e – notes based on location, time and other contexts that get triggered on the match of the context. The location of the user is detected using a GPS and it forms the basis for triggering of location context based e-notes. The experimental test bed uses GPS traces collected with the University of Arizona as the coverage area. However the application can be widely deployed at any place.

The approach used in the development of SENSE is based upon the Stick-E framework as proposed by P.J. Brown and incorporates most of the features of the framework. Furthermore, we have gone a step ahead and incorporated new techniques to make the application more efficient. These include having client side caching to reduce communication overhead, support for disconnected operations and ability to prefetch data when the user leaves a coverage area.

The remainder of the paper is organized as follows. In section 2, we look at the related work done in this area. Section 3 contains the core of the paper, the design issues and important features of SENSE. In Section 4, software components are listed and related to the components of a Stick-E framework Implementation details are given in Section 5 followed by Testbeds and Experiments in Section 6. Section 7 lists the various areas of future work and finally in Section 8 we present our conclusion.

2. Related Work

Most of the work done in this field so far has focused on the design of the system rather than the implementation.

In [1] P.J. Brown laid the foundation for stick – e notes by describing a Stick-E note architecture. In this he describes the concept of Stick-E notes and their contribution to context aware computing. It also gives a lot of ideas about how to structure the various software components necessary in the implementation of a toolkit for Stick-E notes. Our work has been greatly influenced by this architecture.

Pascoe et. al. have implemented a stick-e note system as described in [12]. Their system is limited to a client side application. The system was developed for field workers in an area where wireless connectivity was unavailable. Our implementation distributes software components between an E-note Server and clients. Our implementation is also meant for wide scale deployment in areas of wireless coverage with facility for disconnected operation.

Support for Disconnected operations has been a hot topic in mobile computing. The Coda and Andrew File Systems support disconnected mode of operations for mobile nodes [5,12]. In this they describe a feasible and efficient mechanism to implement disconnected operations enabled by client side caching. We have used similar mechanisms to support disconnected operations.

3. Design Features

Location Model

There are two types of Location Models [4]. One is the *Symbolic Model* in which the context is associated with a certain area. The other is the *Geometric Model* that refers to a single point in an area. We have used a hybrid model that supports both types of models in which an area is represented by the diagonal . When we want to map to a single point we use the same method as for mapping an area but with the only difference that the top-left and bottom-right coordinates are the same i.e. the diagonal is a single point. So the co-ordinates map to a single point.

Client Caching

The key feature to our client side application running on a handheld device is that it uses caching thereby reducing the communication overhead and reduces the possibility of frequent disruptions in connectivity being a problem.

The schemes used are:

Whole file caching: The entire campus is divided into cells and whenever a user enters a cell all the notes for the client associated with the cell are cached and then are locally triggered based on the context.

- ***Disconnected Operations***: This scheme of caching has the advantage that it is easy to support disconnected operations. Even when the connectivity with the server is down, the client can use the locally cached copies and all updates can be postponed till the time connectivity is reestablished.
- ***Prefetching:*** Related to disconnected operations is the concept of prefetching. This could be particularly useful when a user is moving into an area with no cell coverage, in which case he could prefetch the cache notes for that place.
- ***Reintegration:*** When the user moves from a region of disconnectivity to an area with cell coverage all changes made to the e-notes in the disconnected mode are propagated to the server for updation.
- *A* ***Write Through Cache*** is maintained, so all the updates are propagated to the server immediately, except when the server is down / loss of connectivity.
- ***Write Back Strategy:*** There are timestamps associated with every e-note that has to be updated at the server before the user leaves a particular cell. The Write through mechanism mentioned above doesn't ensure complete updation, as it is possible that the timestamps could have changed without any

modifications to the e-note structure. So in order to complete the updation the e-note data is written back before the client leaves a cell. This is like a Write back Strategy. The advantage of this Write back scheme is that it also ensures that all the changes made to the client e-notes for that cell are updated a second time too. This is helpful if the e-notes were not written to the server during the Write through stage due to loss of connectivity and other reasons.

Note Categories

Notes can be of 3 types.

Private Notes: These are the most common type of notes and are associated with only the owner of the note. Owner refers to the user who created the note. As the name suggests the notes are triggered only for the owners. Examples of these could be reminders that the user could set like "Withdraw Cash from ATM".

Public Notes: These can be associated with everyone registered with the server. Public notes could be advertisement messages, warning messages or other messages that could be for the general public. The user can also categorize the public notes based on the sub type. The sub types just groups the public notes into smaller subsets based on the relevance or importance. We could have sub types "Announcements" that can have advertisement messages and news flashes and "Alert" messages that can have warning messages.

Group Notes: These are the last type of messages and are to support the possibility of the user belonging to groups .Then he should be able to associate notes with the groups. E.g.: User 'A' comes early to a meeting and then he could set a note to all members in a group associated with the meeting location saying "Will be back in 10 minutes don't start the meeting".

Actions

Actions refer to events that are raised when an e-note is triggered. A couple of actions are supported in the current implementation. The first and the most common is a ''Popup form'' that is popped up when an e-note is triggered. The other action is a 'Beep' alerting the user that an e-note has been triggered.

Power Optimization

Power optimization is very critical in a handheld device where battery life is limited. However, when developing an application there should be a judicious balance between the workload on the client and the server. Moreover, if the entire load is on the server, though the server could be on a wired network, there could be the overhead of sending packets across a weak wireless link. In SENSE, the workload is equally distributed between the client and the server. The caching at the client reduces the client – server interaction. However, definitely the client can make more optimizations to save power. The current version does not do any such power optimizations as the first step was to build a bare bones structure that recognized all the contexts and triggered the right notes at the right time. Though our user interface supports these options, they are not implemented.

The future releases of SENSE will have more power saving options.

Multiple Device Delivery

The field of mobile computing is flourishing at a rapid pace and the number and variety of handheld devices coming out in the market is also increasing. Bearing this fact in mind, the implementation and design has been centered on a modular structure that can be ported to any device. The client side development has been done using embedded Visual Basic (eVB) which supports a variety of mobile devices like handheld PC's, pocket PC's and palm PC's. It is believed that devices of the future will also run Windows CE and be eVB compatible. The added advantage of using eVB is that it can be converted to a Visual Basic application, making it possible to run the client on a workstation or laptop.

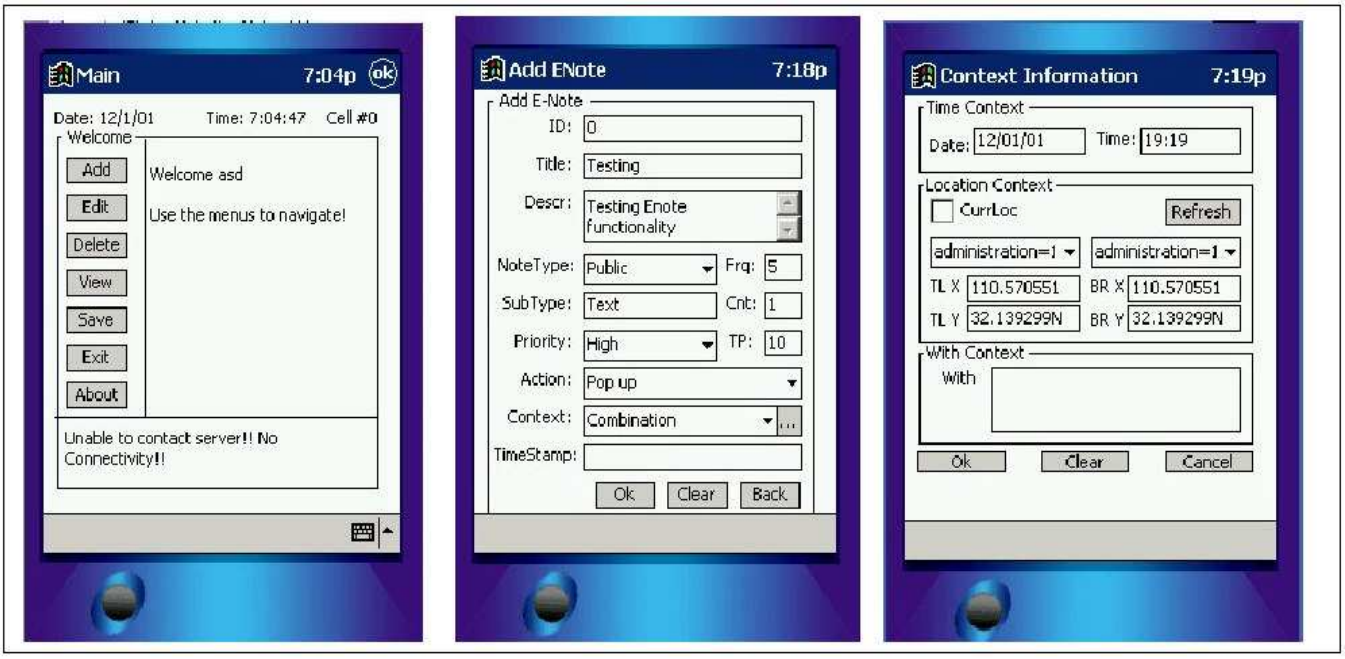

Figure 1: Snap Shots of the User Screens. (a) The main menu (b) screen used to add notes. (c) screen specifying the context information for the enote.

The use of XML, the web language of the future, also furthers the cause of multiple device support. It is platform independent and will definitely not become obsolete for many more years to come.

The server is written in Java that is also platform independent.

For GPS, the standard NMEA protocol has been used. This is a globally acclaimed standard and is supported by most of the GPS receivers.

3. Software Components

The seminal paper by P.J. Brown [1] on a stick e document framework has greatly influenced our system design has been influenced. In [1] Brown proposes the following software components for such a system:

- SEPREPARE
- SEMANAGE
- SETRIGGER
- SESHOW

In this section we explain how these components are implemented in our system.

SEPREPARE

Users / Authors of e-notes can create them using a form based embedded visual basic User Interface module on an iPaq PocketPC. The user can specify the following contexts:

Location: These can be Symbolic locations (an area) or Geometric Locations (a particular point).

Time: These are e-notes set to expire at a specific time.

Adjacency: There can be notes that are triggered when near, in the same cell of the coverage area, another user.

In addition to specifying the context, the user can also set the number of times the e-note must be triggered.

Every e-note that gets created is assigned an 'e-noteid'. The 'e-noteid' is unique and formed from the user name followed by a timestamp.

The e-notes can also be viewed, modified and deleted. The modify and delete operations use the 'e-noteid' to perform the operation on the right e-note.

A screenshot of the user interface is shown in Figure 1.

SEMANAGE
Managing E-notes: When users in any part of the coverage area creates, modifies or deletes notes a message with the information is to a central e-note server that stores all the e-notes. The e-note server stores these e-notes in an XML file. The communication between the user and the server is through a secure TCP connection. The specifics of the communication protocol are given in detail in section 5 on implementation.

SETRIGGER
The e-notes received by the client application on the iPaq get triggered by an embedded visual basic background application that is continually checking the e-note cache to check if any e-note's context is satisfied. Location sensing is done by the application via GPS NMEA input data from a GARMIN eTrek GPS receiver that is connected to the iPaq. The current time is read from the iPaq system clock.

SESHOW
When the e-notes are triggered the action that is associated with that e-note is executed. Currently a 'Popup form' is displayed on the user's screen with the title, description and other details of the e-note. E-notes might also alert the user by a beep on being triggered, if the user while creating the e-note enabled this feature.

5. Implementation

E-note Structure
The e-notes are maintained at the client and the server as XML objects. The server on boot up loads the e-note list from an XML file and stores these details in an XML Document Object Model (DOM) object.
The representation at the client is also similar. A template of the e-note structure is shown in figure 2.

```
- <ENOTES>
  - <ENOTE>
      <CID>sam</CID>
      <ID>sam-12/1/01 3:22:31 AM</ID>
      <TITLE>Call</TITLE>
      <DESCR>Call home</DESCR>
      <TYPE>0</TYPE>
      <SUBTYPE>rff</SUBTYPE>
      <PRIORITY>0</PRIORITY>
      <ACTION>0</ACTION>
      <FREQ>4</FREQ>
      <TIMEC DATE="" TIME="" />
      <LOCATIONC BL="110.574162W" BR="32.140517N" TL="110.574162W" TR="32.140517N" />
      <WITHC FLAGS="1" IDTYPE="5" UID="" />
      <FLAGS>1</FLAGS>
      <TIMEPERIOD>12</TIMEPERIOD>
      <TIMESTAMP>37226.1406365741</TIMESTAMP>
      <COUNT>1</COUNT>
    </ENOTE>
  </ENOTES>
```

Figure 2: XML representation of the enote structure used

Communication Protocol
The client and server communicate with each other using the Transfer Control Protocol (TCP).

The messages that the client server exchange are:

Cache Request Message
The user in the following cases may trigger a CACHE_REQUEST message:

- The client application detects that it has crossed a cell boundary.
- The client wants to prefetch data for a certain cell. In the current implementation the prefetching is explicit. A snap shot of the user prefetch screen is shown in figure 3

It is the responsibility of the user to cache the data before going into a region of no service. We plan, however, to develop heuristics to determine with a high degree of certainty that a client will soon be crossing over to another cell or to an area with no coverage. This algorithm can be based on the observed pattern of motion of the user (both the speed and orientation as provided by the NMEA protocol). This idea is similar to the concept of hoarding in the Coda file system [5]. Such a heuristic has also been

proposed in [4]. This will enable the client application to implicitly prefetch the data.

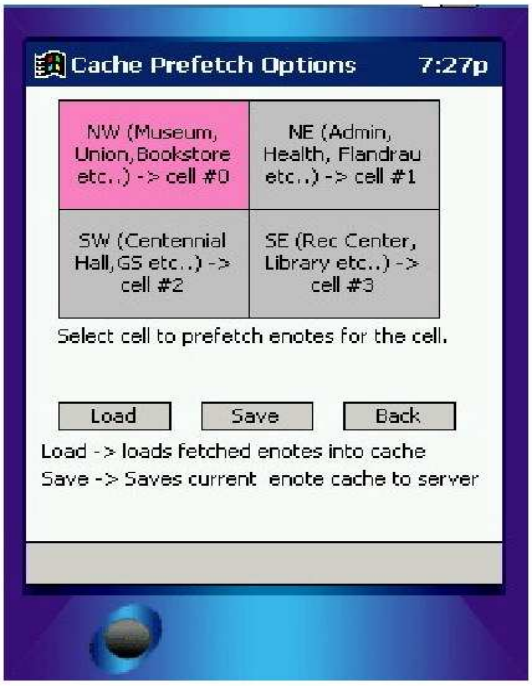

Figure 3: The pink region is where the user is currently located. By clicking on any of the (other) cells he can prefetch the data for that cell.

Figure 4 shows two user movement patterns from which we can with a high degree of certainty predict that user1 will soon be entering cell 1 and user2 will soon be leaving the coverage area.

E-notes are served to clients, on receiving a CACHE_REQUEST message from a user in a specific cell of the coverage area.

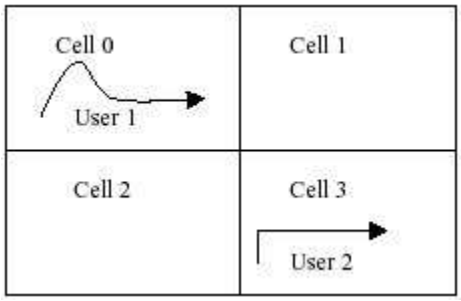

Figure 4: User movement patterns for which cell boundary crossover can be predicted with a high probability.

Cache Response Message

On receiving the CACHE_REQUEST, the e- note server sends a CACHE_RESPONSE to the user containing e-notes (this may be user created and public notes) for that particular cell.

Add Note Message

All new notes that are added by the user are sent to the server as an ADD_NOTE message. The message packet contains the e-note data as an XML file that is parsed by the server to update its e-note list.

Modify Note Message

These are similar to add note messages except that the modified notes are sent as a file and the server replaces the existing notes that match the new modified notes. Matching is based on the note id.

Remove Note Messages

To remove an e-note a REM_NOTE message is sent that just contains the user name and 'e-noteid'.

User Personalization

All user profiles are also stored on the central e-note server. The user may choose to not receive public e-notes by setting his profile to reflect this.

6. Testbed and Experiments

The testbed for our experiments include a

- A Sony VAIO laptop computer with Pentium II, 128 MB RAM and 433 MHz. This was used as a base station (server).
- An IPAQ 3670 PDA, with Windows CE 3.0, 64 MB RAM, 16 MB ROM, Intel Strong ARM, SA1110 was used as the client.
- All the development was done on a 733 MHz, 128MB RAM workstation and then ported to the handheld device and the laptop.

- The laptop and the IPAQ were in the 802.11 network in the adhoc mode. For this, 11 Mbps Lucent Technologies ORINICO Wavelan Cards were used.
- A GARMIN ETREK Summit GPS Receiver.

All the experiments were done within the campus of the University of Arizona.
The coverage area included the 4 boundary points of the campus as shown in figure 5.

Configuring the 802.11 Network

The first step in establishing the communication between the IPAQ and the laptop was configuring the 802.11 network in the adhoc mode. Contrary to our belief that this would be a rather difficult milestone to cross, it was rather trivial configuring the network. To configure the network first both the machines were setup in the same subnet by giving them a same subnet mask. The IP addresses used for the laptop and IPAQ were 172.16.4.10 and 172.16.4.11 respectively. The subnet mask was 255.255.255.0. Once this was done the other details of the network were removed. Then the profiling information was changed so that the devices belong to the same network group. The configuration we used was:
Profile name: cs630c (used in identification), Network Type: P2P group, Network Name: enote (required for the devices to talk). A static channel (7) was used for communication.

GPS Tools

GPS Talk

All the experiments were done using GPS simulation traces that contained the position coordinates of various places in the campus. The reason for this is that it was impossible to test the software indoors along with the GPS, as GPS doesn't work indoors. To make the process of data collection easy a tool, GPS TALK, to collect the information was developed and used. The tool was upgraded many times and the

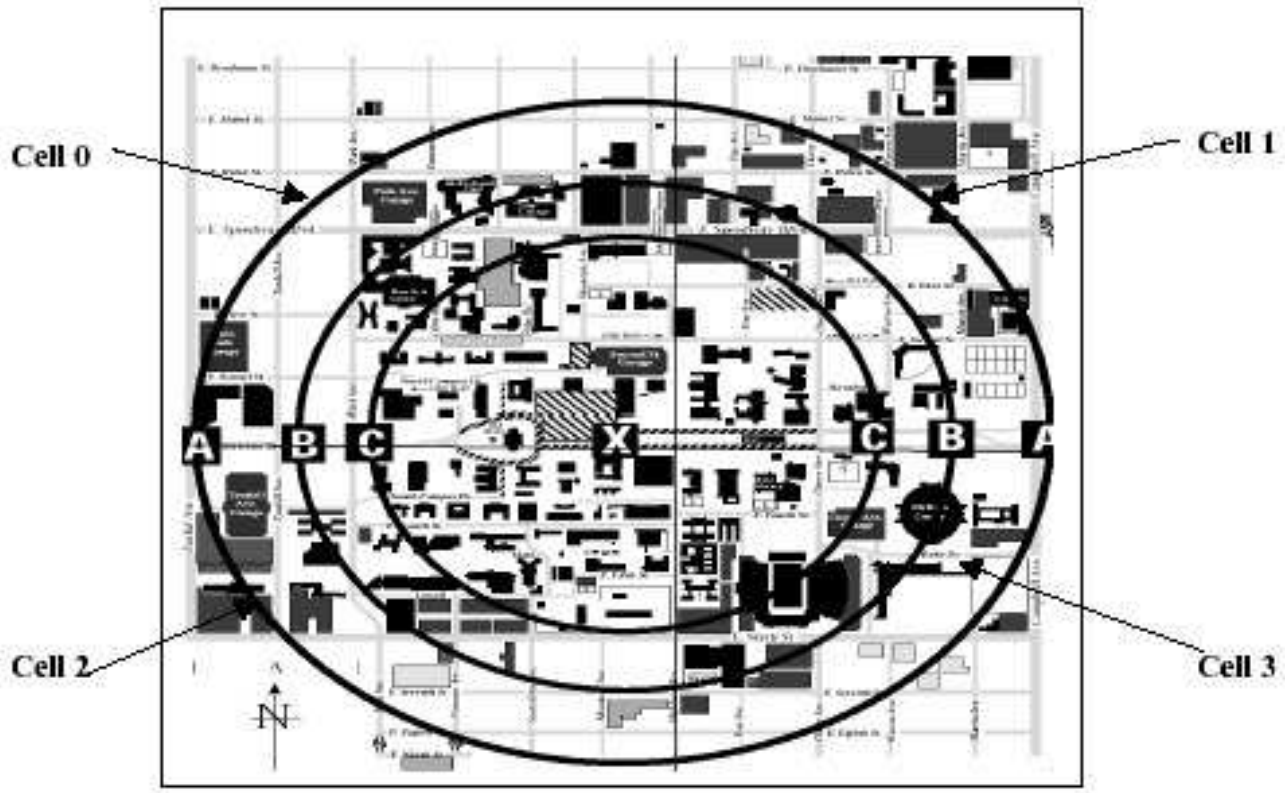

Figure 5: Map of University of Arizona that was the coverage area for the experiments. The campus was divided into four cells that had an equal distribution of important places

latest version available is GPS Talk 3.0. We used this tool to capture location points. This is shipped along with SENSE.

However due to the inherent limitation of a GPS receiver to not work indoor trace files were used for testing. The GPS Emulator mimicked the working of the GPS and provided the values of the co – ordinates from the trace file rather than collecting the information using real time co – ordinates.

```
$GPRMC,003214,A,3213.7992,N,11057.2809,W,0.0,108.5,301101,11.4,E,A*30
$GPRMC,003216,A,3213.7992,N,11057.2808,W,0.0,108.5,301101,11.4,E,A*33
$GPRMC,003218,A,3213.7993,N,11057.2807,W,0.0,108.5,301101,11.4,E,A*33
$GPRMC,003220,A,3213.7993,N,11057.2806,W,0.0,108.5,301101,11.4,E,A*39
$GPRMC,003222,A,3213.7994,N,11057.2805,W,0.0,108.5,301101,11.4,E,A*3F
$GPRMC,003224,A,3213.7994,N,11057.2804,W,0.0,108.5,301101,11.4,E,A*38
$GPRMC,003226,A,3213.7995,N,11057.2802,W,0.0,108.5,301101,11.4,E,A*3D
$GPRMC,003228,A,3213.7995,N,11057.2801,W,0.0,108.5,301101,11.4,E,A*30
$GPRMC,003230,A,3213.7995,N,11057.2800,W,0.0,108.5,301101,11.4,E,A*38
```

Figure 6: Sample Trace File

GPS Emulation

A GPS Emulator was also implemented that provided the GPS values to/from a trace file rather than the actual co – ordinates. This was very useful for testing.

Experiments

Collecting Co – ordinates

Using the campus region as the coverage area a walk of the entire campus was done collecting the co ordinates of the important places within the campus. Also the boundaries of the coverage area were found. GPS Talk 3.0 helped in this data collection.

Based on the co ordinates collected the campus was divided into 4 cells. This was done such that each cell had equally important places like the bookstore, food court, and recreation center. Figure 5 shows the campus map and how it was divided into cells.

Testing

The application developed keeps reading the GPS coordinates from the GPS receiver, that is interfaced with the PDA, every second. Figure 6 shows a sample of the trace file. As the file sample shows it has details of the longitude, latitude of points that would be encountered by the user when he is actually walking around campus. So every second the next location from the file is read and notes are triggered based on this location and the current time.

Results

- Enote contexts were successfully triggered.
  - Addition / deletion / updation of enotes was successful in the presence and absence of connectivity.
  - Enotes were properly triggered at the correct context.
- Operation of SENSE in connected mode was successfully tested.
  - Write-through to the server for e-note based operations was successful.
  - Write-back of cache to the server on moving out of a cell was successful.
- Operation of SENSE in disconnected mode was successfully tested.
  - Prefetching of Enotes before entering a region of no connectivity was successful.
  - Re-integration of cache contents at the server (once connectivity is re-established) was successful.

7. Future Work

SENSE is under active development. In earlier sections of the paper we discussed about the design of SENSE and what has been done so far. In this section we list the various directions in which we intend to broaden our work.

- In the current implementation the Client has only a cached copy of the data. It could be further extended so that the Client also saves the files to his hard disk so that it could help in connectivity and persistency as in Coda.
- One file is maintained at the Server for all the clients. This increases the latency at the server. We plan to have a faster indexing mechanism that will enable quicker lookups. One such mechanism could be to have one XML file per client.
- An excellent opportunity also exists to provide very good QoS to users. E.g. notes delivered could be based on priority. Though the user interface supports options for specifying the number of notes to be cached or the cell size, these options are not supported and will be done in the future.
- Right now only a couple of basic actions are triggered on an event. This can be further extended such that active context aware applications can be triggered.
- As had been stated earlier in the paper, active steps will be taken to optimize to save power.
- The focus of our research was to use the Stick-E framework to develop context aware applications and not much emphasis was given to the server side development. So for commercialization we can extend this.
- Using Microsoft Embedded Tools – Platform Builder, a platform specifically for SENSE can be developed.
- Using SENSE we plan to develop 'documents' of e-notes. For E.g. Tour guide etc.
- In the current implementation when the Save option is selected the Cached data at the client is sent to the server that saves it the information in its Database. This ensures that modified and new notes are updated. Delete operations, however, are not replayed back. In order to do this we can have a replay log with checkpoints, and perform all operations since the last checkpoint. One can also use compaction algorithms to remove redundant log events and reduce the number of transactions.

8. Conclusion

We have presented a toolkit that supports stick – e frameworks. The Stick – e architecture, proposed by P.J. Brown, has inspired the design and implementation of SENSE that have been presented in this paper

The toolkit supports a variety of contexts like location and time and has all the components of the stick – e framework working. It also improves on the suggestions of P.J. Brown by supporting disconnected operations, prefetching and caching at the clients.

This is a preliminary version of the toolkit and work is under progress to improve the performance and also to incorporate many more features that are currently not supported. More information on the work described in this paper is available on the web at http://www.cs.arizona.edu/people/srinivas/enotes

Acknowledgements

We wish to thank Dr. Nigel Davies for his valuable feedback and comments. We also thank the faculty in the Computer Science Department at The University of Arizona and friends for their help and support.

References

[1] P.J. Brown. The Stick – e Document: A Framework for Creating Context – Aware Applications. Proceedings of the EP '96.

[2] P.J. Brown. The electronic Post – it note – a metaphor for mobile computing

applications, University of Kent at Canterbury (1995).

[3] P.J. Brown, P.D. Bovey and X. Chen. Context – Aware applications: From the laboratory to the Marketplace. IEEE Personal Communications 4 (5), 1997, pp.58 – 64

[4] Leonhardt, U. "Supporting Location-Awareness in Open Distributed Systems". PhD Thesis, Department of Computing, Imperial College of Science, University of London. (1998)

[5] Kistler, J.J., Satyanarayanan, M. “Disconnected Operation in the Coda File System”. ACM Transactions on Computer Systems. Feb. 1992, Vol. 10, No. 1, pp. 3-25.

[6] Nigel Davies, Keith Cheverst, Keith Mitchell and Adrian Friday. 'Caches in the Air': Disseminating Tourist Information in the Guide System. Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications1998

[7] A. K. Dey, D. Salber, and G. D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 16. 2001

[8] Andy Harter, Andy Hopper, Pete Steggles, Any Ward, and Paul Webster. The anatomy of a context-aware application. In Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 1999), pages 59--68, Seattle, WA, August 1999. ACM Press.

[9] Jason I. Hong. Context Fabric: Infrastructure Support for Context-Aware Systems. Qualifying Exam Proposal. University of California at Berkley.

[10] Jason Pascoe. Adding generic contextual capabilities to wearable computers. In The Second International Symposium on Wearable Computers, pages 92--99. IEEE Computer Society, Oct 1998

[11] Jason Pascoe. The Stick-e note architecture: Extending the interface beyond the user. In Proceedings of the 1997 International Conference on Intelligent User Interfaces, pages 261-264, Orlando, FL, January 1997. ACM Press.

[12] Jason Pascoe, David Morse, and Nick Ryan. Developing personal technology for the Field. Personal Technologies, 2(1), March 1998

[13] Albrecht Schmidt, Michael Beigl, and HansW. Gellersen. There is more to context than location. In Proceedings of Workshop on Interactive Applications of Mobile Computing (IMC'98), Rostock, Germany, November 1998. Neuer Hochschulschriftverlag

[14] Roy Want, Andy Hopper, Veronica Falcao, Jonathon Gibbons, "The Active Badge Location System", ACM Trans. Inf. Sys., Vol. 10, No. 1, 1/1992.

[15] L. Huston and P. Honeyman, "Disconnected operation for AFS." Proceedings of the USENIX mobile and location-independent computing symposium, August 2-3, 1993, Cambridge, Massachusetts. Pages 1-10.

[16] Mendel Rosenblum and John K. Ousterhout. The design and implementation of a log-structured file system. ACM Transactions on Computer Systems, 10(1): 26-52, February 1992