Simulation of Electronic Foot Shackle with GPS

Department of Computer Science, University of Umeå
June 10, 2004

Stefan Adolfsson (c00san@cs.umu.se)
Emil Gunnarsson (c00egn@cs.umu.se)
Niklas Larsson (c00nln@cs.umu.se)

Abstract
This report presents the project “Simulation of electronic Foot Shackle with GPS” at the department of Computer Science at the University of Umeå. The report describes the two database based prototype Java-applets, the actual surveillance application and the user simulator, that together form the electronic foot shackle simulation prototype. The prototype shows how the functionality of foot shackles sentenced to some criminals could be extended by including a GPS system connected to an advanced database into the foot shackle. It also shows how such an extended system could be implemented and how the data could be represented to the police in an easy and effective way. The prototype database has been tested and proved to scale satisfiably. It also gives the user a good feel of how a real GPS based criminal surveillance system would work.

1 Introduction
“Simulation of electronic Foot Shackle with GPS” is a project in the course “Advanced Data Models and Systems, Spring 04” at the department of Computer Science at the University of Umeå. This article describes the prototype application developed to solve the following problem.

More and more criminals are sentenced to wear an electronic foot shackle instead of going to prison for certain crimes. These shackles alarm the police if some condition is broken. Currently they are only used to check if the criminal is at home at certain hours. If the criminal is not, the shackle alarms the police.

The purpose of this project was to design a database based prototype application that would extend the functionality of these foot shackles by
including a GPS system into the foot shackle. GPS is a system utilizing satelites to decide the position of a GPS transmitter. With this, the police can be alarmed in several different scenarios. For example if the criminal is not at home when he should or if the criminal is too close to a person that he has an restraining order on (a victim). The simulation prototype was then implemented to give the user a feel of how a real life implementation would work.

The database is updated at every time step with positions from the GPS foot shackle that the criminal wears and the GPS alarm device (for example) that the victim might have. The above examples trigger events that notify the police and/or the victim.

2 Approach

The first thing to be decided upon was which database system and programming language to use. Java applets are portable and quite easy to connect and send commands to databases through. Both client applications are Java applets which sends SQL-commands via JDBC to the database.

Limitations of the system were agreed upon and an EER-diagram matching its functionality was created. Thereafter SQL-code to generate the corresponding database was written. The database consists of nine tables. Some time was spent looking for an appropriate PostgreSQL driver for Java. This driver is for wrapping between ODBC and JDBC.

The project was split into three tasks: SQL-code in the database, the surveillance application and the user simulator. Since the other tasks depend heavily on the SQL-code in the database it had highest priority at the beginning of this project. The user simulator and the surveillance application was then implemented as two standalone applications using corresponding parts of the previously implemented SQL-code.

The user simulator simulates criminals, victims and their movement and updates the database in the same fashion that the GPS foot shackles would do when used in real life. This application allows the user to clear the database, add a specified number of individuals to the database, criminals and victims with random house arrests and restraining orders. There is also a field for varying the delay between updates. It is also possible to make the individuals in the database move around by pressing 'run'.

The surveillance application reads from the database two times every second. It fetches the position of all characters and whether any violations has been registered into the database. This is the part that would be used by the police force in real life. Every person in the database is shown in a user list and using a popup menu on this list, by pressing the right mouse button, the details of that user can be shown. The users are plotted on the screen as tiny people surrounded by a colored circle. The color of the circle depends on the state of the user. Currently three different states are supported. The first is the normal/default state which is shown by a black circle. A red circle means that a restraining order has been violated. Both the criminal and the victim gets a red circle. The last color used is blue and that states that the user has violated his house arrest. Feedback on
the violation is also displayed in plain text in the text field displayed on the application.

3 Results
The work described in the approach resulted in two applications, combining to form the Electronic Foot Shackle Simulation System. The two applications are shown in figure 1 and figure 2.

The Electronic Foot Shackle Surveillance Database System prototype has been tested with the user simulator application that was created for that sole purpose. A simulator application was chosen since, for obvious reasons, testing this in real life situations with real users and a real GPS system would be extremely difficult at this early stage.

The performance of the database has been tested by measuring the number of updates during 60 seconds for different number of entries in the database and by calculating the amount of disc space that each entry takes up.

For an amount of 200 persons connected to the database where the expected number of updates is 12000, the number of actual updates also is 12000. However if you duplicate the amount to 400 people, the expected number of updates is 54000 but the realized number of updates stays at 13432. For an amount of 900 persons connected to the database where the expected number of updates are 54000, the realized number of updates are 13432.

To calculate how much space a user entry takes in the database you calculate:

\[
\begin{align*}
\text{vst,vet (2 * timestamp = 2 * 8) 16 bytes} \\
\text{posx, posy (2 * double = 2 * 8) 16 bytes} \\
\text{id (1 * integer = 1 * 4) 4 bytes}
\end{align*}
\]

This makes 36 bytes for an entry in the position table. A day with an entry every tenth second makes 311040 (24*60*6*36) bytes = 304kB = 0.304MB. With 50 users and a ten second interval between updates, the database will grow by 15.2 MB in a day.

![Figure 1. The User Simulator](image_url)
4 Discussion

The database uses the relational model. This has quite a drastic effect on performance in the case when the position table grows large. Consider a query when reading the position of person X. In the relational fashion, all positions of all users are saved in the same table, hence all entries for all persons must be read. In an object model approach, each person would link to its own position object. The query would then only have to find that person and search its position table.

The system has proved to be able to handle 200 clients with an update each second. One might claim that 200 clients are not that many but in a “real world” implementation an update per minute or even less often is more likely to be used, and with that in mind the prototype scales quite good. The disc space needed for the database now averages between 10-100MB per day depending on how many clients that are connected and of course the update interval. But with the low cost for disc space nowadays that shouldn’t be a problem. If needed, a deletion of entries done more than some time ago, or entries selected by some other criteria, could be performed to keep disc space constant.

Computer security is always an issue when dealing with Java applets. This project has taken security problems quite lightly. The user who runs the applets must use a configure policy that permits applets to make network connections through the Java socket API. Also the login information to the database is stored in plain text in the code and can be easily reversely engineered. This is something that we have thought about and chose not to handle in this prototype, it will however be handled in later releases if the project would be continued. This is something that would be very important in a real life implementation of the prototype.

Many features of postgresQL such as triggers and constraints have been used. The database implemented in the project uses complex triggers and computations and is not only used for storing information. It is very convenient to use such features in SQL so that the applications connected to the database doesn’t need to make calculations on the database data.
One may think of several ways to extend the functionality of the foot shackle surveillance system. A function that would be very important in a real system of this kind is to be able to detect disfunctional GPS units. Units that has not sent any data to the database in a certain amount of time is probably broken and should be replaced. This has to be reported in some way. It could also be the case that the criminal has intentionally broken his or her foot shackle to be able to move freely around town, or even leave town, therefor the alarming of this event would be quite important. Another possibility of improving the prototype would be to implement support for checking if the criminal is at work at times when he should and thereby further control his house arrest sentence.

The map in this prototype is only shown as a background with people and houses. In future releases the map would ofcourse be replaced by a real map and the x and y positions would be displayed as latitude and longitude values. This was not implemented in this prototype as the purpose of this prototype was only to give the user a general feel of how the real future GPS based surveillance system would be used. In future implementations it would be possible to quickly see which address to find the criminal if he had broken his sentence.

5 Conclusions
This report has described our solution of creating a GPS based surveillance system for criminals sentenced to wear a foot shackle and our thoughts about the implementation.

We have created a prototype that shows a simple way to display the information that is stored in the database. The prototype clearly shows when a violation has occured, both graphically and with text.

By implementing this project we have shown that SQL is capable of complex triggers and computations and not only for storing information. We have also concluded that it is convenient to use such features in SQL so that applications using a database doesn’t need to make calculations on the database data.

The system has proved to be able to handle 200 clients with an update each second. Based on this information we consider that the database would scale quite good, used in a 'real world' implementation. However we have also concluded that an object model approach might have been a better choice for the database. Disc space should also be no problem.

Future versions of this prototype of a surveillance system for criminals would be possible to use in real life if some alterations would be made to the program. If this was to be realized ofcourse also client side programs used by the GPS foot shackles would need to be implemented.

The prototype is a good way for the user to get a feel of how a real GPS based criminal surveillance system would work.