Location Tracking and Ubisense

Unit 03
October 28, 2010
Alois Ferscha, Michael Matscheko

Pervasive Computing Infrastructure
WS 2010/11
What is position / location?

- **Absolute** position
  - all objects share a common reference frame
  - e.g. GPS, Ekahau (WLAN), …

- **Relative** location
  - each object has its own reference frame
  - e.g. near RFID scanner, 1 m in front of door, …

- **Symbolic** location
  - encompasses abstract ideas where sth. is located
  - e.g. rooms, zones, countries, …

**Related information**
- **Orientation** (roll, pitch, yaw)
- **Velocity, Acceleration**, …
Positioning – Geometric Model

Geometric Model

A location is specified as a three dimensional coordinate, e.g., the latitude, longitude and altitude

or a set of coordinates determining an area’s bounding geometric shape (such as a polygon)

- Point Geometry
- Curve Geometry
- Segmented Curve
- Geometric Surface
- Cartesian Coordinate System (2D, 3D)
Tracking vs Positioning

Two methods for location sensing

- **Tracking**: a sensor network tracks the positions of moving objects, usually with the help of certain tags, mounted on or carried by the tracked objects or persons.

  Can use unpowered or at least low-powered tags.

  e.g. Ubisense, (multi-)camera motion trackers, …

- **Positioning**: the moving object determines its position itself, by sensing environmental properties, usually supported by transmitters in the infrastructure.

  Can use pre-existing or global infrastructure.

  e.g. GPS, Ekahau (WLAN), …
A Survey and Taxonomy of Location Systems

- Classification of location systems
  - **Satellite-based** location systems (require line-of-sight)
  - **Indoor** location systems (poor deployment at the moment)
    - Complex and expensive infrastructures, required wearing of badges
  - **Infrastructure-based** location systems (based on existing infrastructures)
Positioning Systems – Accuracy and Applicability

Indoor GPS, Ubisense, Ekahau, ... cm - m
Features and Limitations

Setup

.NET API

TCP/IP Location Provider
Ubisense

Highly scalable tracking system

- Tags emit ultra-wideband (UWB) radio pulses
  - very short pulse in broad frequency spectrum
  - better accuracy in cluttered and multipath environments
  - less interference with other RF transmitters

- Sensor infrastructure determines location, based on both Angle-of-Arrival (AoA) and Time-Difference-of-Arrival (TDoA)

- Tag has to be in range of at least 2 sensors (better 3 or more)

- Accuracy up to 15 cm (3D), in practice about 1 m
Ubisense Tags

Series 7000 Compact Tag

- 38 x 39 x 16.5 mm, 25 g
- 1 Button, 1 LED
- UWB radio (6 – 8 Ghz)
- Two way communication with an additional 2.4 GHz radio. Allows dynamic update rate changes, battery status checks, LED control and user interaction with the button
- mechanically robust, dust- and waterresistant
Ubisense Sensors

Series 7000 Sensor

- 200 x 130 x 60 mm, 650g
- Networking through **wired** ethernet or **wireless** LAN. Power-over-Ethernet support
- Remote controlled over network
- **Update rate 135 Hz**, can theoretically provide a location update every 7.4 ms (e.g. about 5 updates per sec for 25 tags, in practice much less)
- **Modular cell structure**, configured by software. Multiple cells allow almost unlimited area coverage and amount of tags
Ubisense - TDoA with RF signals

Time Difference of Arrival (TDoA)

- Different arrival times at multiple sensors (with known location) allow calculation of position.

- Ubisense sensors have an approx. 90° reception cone. At least 2 sensors are necessary for a position estimation.

- RF/EM signals travel at speed of light (3x10^8 m/s). To measure different arrival times at just a few meters distance, we need extremely well synchronized clocks.

- Ubisense sensors require an additional set of cables just for timing synchronization.
Ubisense - Angle of Arrival

Angle of Arrival (AoA)

- Position calculated from the angles of arrival of a single signal at a minimum of 2 sensors
- Accurate estimation of angle of arrival is difficult (hardware limitations)
- Angle is heavily affected by signal reflections
- Low confidence
Ubisense – Software Framework

Platform Control

- Service startup or stop
- Dataset export (server migration)
Ubisense – Software Framework

Site Manager

- Room / Cell / Area / Object / Tag ... configuration
Ubisense – Software Framework

Site Manager

- Sensor setup and calibration (automatic, assisted, manual)
Ubisense – Multicell Setup
Ubisense out of the lab
Ubisense - .NET API

Extensive API for .NET 2.0

- **UbisensePlatform.dll**
  The definition of core Ubisense objects.

- **UbisenseLocationEngine.dll**
  Modules to configure and use the actual real time location engine hardware.

- **UbisenseLocationServices.dll**
  Its modules manage site tasks, like spatial monitoring and cell extents.

- **UbisenseVisualization.dll**
  Modules that give the visualization functionality.

Check API documentation for more info.
Ubisense - .NET Sample Application

Simple console application

- Tells us when a person enters or leaves a zone
- Persons / Zones are defined in the Site Manager
Ubisense - .NET Sample Application

```csharp
// Create an example event client for a USpatial::Cell
    Ubisense.USpatial.Monitor.Schema(true);

// Add the insert/delete handlers for the monitor_schema.
// These will be called when rows are inserted/deleted into/from
    monitor_schema, OnInsert);
    monitor_schema, OnDelete);

static void OnInsert(Ubisense.USpatial.Interaction _i)
{
    // This code will execute whenever a person enters the 'Alarm Zone'.
    // More precisely, it will execute whenever a row is inserted into the
    // Ubisense.USpatial.Interaction, and _i is the row that has been inserted.
}

static void OnDelete(Ubisense.USpatial.Interaction _i)
{
    // This code will execute whenever a person leaves the 'Alarm Zone'.
    // More precisely, it will execute whenever a row is deleted from the
    // Ubisense.USpatial.Interaction, and _i is the row that has been deleted.
}
```

Enter Event

Leave Event
Ubisense – Location Provider

TCP/IP Ubisense Location Server

- Provides location data to any number of tcp/ip clients
- Server will transmit a location message to all connected clients whenever a location changes (not in fixed time intervals)
- Clients never need to send anything (just connect)
- Location message format (plain ASCII, space separated)
  "<tag-id> <time> <x> <y> <z>\n"

  <tag-id> String, ID of the tag (number on the back of each tag, like 020000135253)
  <time> long, ms since 1970-01-01 GMT
  <x> double, x coordinate in m
  <y> double, y coordinate in m
  <z> double, z coordinate in m
public class LocationMessage implements Serializable {

    private final double x;
    private final double y;
    private final double z;
    private final String tagId;
    private final Calendar time;

    public LocationMessage(String s) {
        final String[] elems = s.split("\s");
        assert elems.length == 5;
        tagId = elems[0];
        time = Calendar.getInstance();
        try {
            time.setTimeInMillis(Long.parseLong(elems[1]) - time.getTimeZone().getRawOffset());
            x = Double.parseDouble(elems[2]);
            y = Double.parseDouble(elems[3]);
            z = Double.parseDouble(elems[4]);
        } catch (NumberFormatException e) {
            throw new IllegalArgumentException("Error parsing location message string", e);
        }
    }

    // getter methods ...
}
public class UbisenseClient extends Thread {

    private BufferedReader reader = null;

    public UbisenseClient(String host, int port) throws IOException {
        Socket socket = new Socket();
        socket.connect(new InetSocketAddress(host, port), 5000);
        reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));
        start();
    }

    @Override
    public void run() {
        String line = null;
        while (!interrupted()) {
            try {
                line = reader.readLine();
                if (line != null) {
                    LocationMessage message = new LocationMessage(line);
                    System.out.println(message);
                    // do something useful ...
                } else
                    interrupt();
            } catch (Throwable e) {
                interrupt();
            }
        }
    }
}

Constructor: Open a TCP/IP socket to the server

Client-Thread: Whenever a new line arrives, parse the message, and do something useful...
Schedule

VO
Mi 03. 11. 2010  08:00 – 10:00  K 012D
Mi 10. 11. 2010  08:00 – 10:00  K 012D
Mi 17. 11. 2010  08:00 – 10:00  K 012D
Mo 15. 11. 2010  15:30 – 17:00  to be announced

UE
Do 04. 11. 2010  14:30 – 15:15  PCL  Sensor Technology 4: Demo, Topics
Do 11. 11. 2010  14:30 – 15:15  PCL  Presentation and Discussion of Topics
Mo 15. 11. 2010  14:30 – 15:15  PCL  Presentation and Discussion of Topics
Do 18. 11. 2010  14:30 – 15:15  PCL  Implementation
Do 25. 11. 2010  14:30 – 15:15  PCL  Presentation System Design
Do 02. 12. 2010  14:30 – 15:15  PCL  Implementation
Do 09. 12. 2010  14:30 – 15:15  PCL  Implementation
Do 16. 12. 2010  14:30 – 15:15  PCL  Competition / Presentation
Do 13. 01. 2011 and Do 20. 01. 2011  Reserved