CSCI {4,6}770: Ubiquitous Computing

Instructor: Surendar Chandra (surendar@cs.uga.edu)

219B Boyd GSRC (542-3478)

Office Hours: 4:00-5:00 (Wed,Thu)
(other times, by email appt)

Course Web:
http://greenhouse.cs.uga.edu/courses/spr02/ubicomp/

Mailing list: ubicomp@greenhouse.cs.uga.edu

Subscription URL:
http://greenhouse.cs.uga.edu/mailman/listinfo/ubicomp

Review forum URL:

http://greenhouse.cs.uga.edu/courses/spr02/ubicomp/Ubiq/edit-member.pl?new

Newsgroup: uga.cs.courses.ubicomp

Outline for today

- Brief Introduction
  - What is ubiquitous computing?

- Course policies:
  - Course organization and expectation
  - Grading policy, late policy, reevaluation policy
  - Academic honesty

- Assignment I:
  - Location problem

What is Ubiquitous Computing?

- Era 1: Mainframes (past)

  Central, “powerful” and expensive computer

  Many users access a single computer from “dumb” terminals

  Used for enterprise data processing

    - Cobol, data bases, JCL etc..

  Computer not easily accessible

  E.g. OASIS

What is Ubiquitous Computing (cont)

- Era II: Personal computers (present)
  - "Powerful" and relatively inexpensive computers
  - At least one machine per user
  - Used for word processing, personal productivity applications, video, audio etc
    - Powerpoint, MS Word, Web browser etc..
  - Computer still not that accessible

What is Ubiquitous Computing?

- Era III: Post-PC (future)
  - Explosion in number and variety of computing devices
  - A number of devices/machines share one user
  - Devices (inexpensive) vary in complexity and function
  - Used to make “our lives better”
  - Computers become “invisible”

Key requirements for ubiquitous computing

- All the devices are network capable, though not necessarily connected all the time. The devices can communicate with each other.
- Thus a computer in your washer, microwave, car emission controller, laptop projector etc are not ubiquitous devices unless you can connect and modify their behavior.
- Devices communicate to seamlessly simplify our lives

Example Ubiquitous scenario

You want to setup an appointment to meet me. You walk up to me and make an appointment

Underlying processes:

- Location: So that I can figure out your computer incarnation
- Communication: So we can decide on how to talk to my appointment scheduler. We have to figure out the mechanisms (Bluetooth), protocols (XML over tcp)
- Authentication: So that I can know who you are and let you see my “appropriate” calendar
- Authorization: So that I can let you make an appointment
- Consistency management: So that the appointment that I made is reflected “everywhere”

Another ubiquitous scenario

My flight gets delayed and so I can't keep the appointment. Our appointment is converted to a telephone call, I automatically call you from the InFlight phone to a pay phone near you

- Authentication
- Authorization: so I can know where you are and entities around you
- Location management: so I know how to contact you (Inflight phone and pay phone near you)
- Socially appropriate: don't route the lovers call to the spouses' home phone!!

Course Organization

- Fairly new and emerging area – no “right solution”
- I encourage open discussion about the technologies that comprise ubiquitous computing
- Research papers will complement class lectures

Systems philosophy

1. Your own implementation

2. Your own simulation

3. Implementation results from a paper

4. Simulation results from a paper

5. Everything else (hearsay, rumors, “I think so”, “I think that it is how it should work” etc)

Syllabus

- Ubiquitous Computing Vision
- Distributed and P2P System Architectures
  - Naming and Location management
  - Replication Services:
    - Synchronization and consistency
    - Caching, Prefetching and Hoarding
  - Security
- Client side issues:
  - Communication protocols
  - Battery power management
  - Adaptive content distribution
  - OS Support for small devices
- Next generation applications:
  - MEMS/Microsensors
  - Convergence (computers/telephones/multimedia/home entertainment)

Grade distribution

- Class participation – 8 %
- Paper evaluation – 6 %
- Midterm – 20%
- Final – 20%
- Homework projects – 24%
- Course project/Final oral exam – 22%

Class participation and paper evaluation

- Class participation is very important. You will be graded on your involvement in class discussions. There are no “dumb” questions. You will only be penalized for “no” questions/comments.
- To foster interaction I will randomly pick name – more than two unexcused absences is grounds for administrative withdrawal.
- Over the course of the semester, you will read and critique research papers. You will discuss your criticisms during the class discussion. Critiquing other work is a good way to start thinking of a better solution
- You should register as soon as possible to provide online critiques

Midterm and Final

- One open book, open notes, in class exam

Homework projects

- During the first half of the semester, you will assigned three programming project that are due every three to four weeks.
- **Projects are individual efforts**.
- Each project should be electronically turned in with a succinct report on your implementation strategy and what you learned.
- Projects should compile without any modifications. If you need a specific OS, you should make arrangements beforehand
- I will randomly select submissions for an one-on-one oral interview

Course Project

- Teams of 2-3 students. Start thinking about what you want to do/partners etc. now.
- Individual grades influenced by the final oral examination and presentation.
- I will provide a list of project ideas
- You will explore a substantial topic and show its validity with an implementation
- At the end of the semester, there will be a mini-symposium where you will publically present your work. Best paper and presentation will be recognized.

Mini-symposium 2001

Reevaluation policy

- Arithmetic errors, missed grading will be reevaluated.
- I encourage you to discuss concerns with your solution with me
- I discourage re-evaluation of partial credits:
  - Football penalty policy:

    If you think you deserve a better partial grade, write down the reason why you think that you deserve a better grade and how many extra points you think you deserve. If I agree, you could get up to this many extra points. If I disagree, you will lose this much points.

Late policy

- None – Projects/homework/critiques are due at 11:00 am (right before the beginning of class). **I do not accept late submissions** (not even a second)

- I use the clock in gemini.cs.uga.edu for reference

- Please contact me regarding unforeseen emergencies

Academic Honesty

- Freedom of information rule:
  - Collaboration is acceptable
  - To assure that all collaboration is on the level, **you must always write the name(s) of your collaborators on your assignment.** Failure to adequately acknowledge your contributors is at best a lapse of professional etiquette, and at worst it is plagiarism. Plagiarism is a form of cheating.

Academic Honesty – No sponge rule

In intra-team collaboration where the group as a whole produces a single "product", each member of the team must actively contribute. Members of the group have the responsibility

1. to not tolerate anyone who is putting forth no effort (being a sponge)
2. to not let anyone who is making a good faith effort "fall through a crack" (to help weaker team members come up to speed so they can contribute)

I want to know about **dysfunctional group situations** as early as possible. To encourage everyone to participate fully, we make sure that every student is given an opportunity to explain and justify their group's approach.

Academic Honesty – Gilligans Island Rule

- This rule says that you are free to meet with fellow students(s) and discuss assignments with them. Writing on a board or shared piece of paper is acceptable during the meeting; however, you may **not take any written (electronic or otherwise) record away from the meeting**. This applies when the assignment is supposed to be an individual effort. After the meeting, engage in half hour of mind-numbing activity (like watching an episode of Gilligan's Island), before starting to work on the assignment. This will assure that you are able to reconstruct what you learned from the meeting, by yourself, using your own brain.

Assignment I – Due 02/05/2002

- Resource Location:
  - Locating what is “around” you is an important problem
  - Need this functionality to access “services”
    - need to know who has a laptop/palm top in this room so that you can communicate with them
    - need to know the location and security key for the wireless access point to use it
    - need to know the printers that are closeby
    - etc..

Beacons

- Let's call this location system as a beacon
- Everything has a beacon running in it so that it can know what else is around it and also enable others to locate it.
  - There will be a beacon in your laptop/palmtop/wireless access point/printers etc.

- Your goal is to write these beacons
- You do not have to worry whether the beacon is close to you. For example, you might locate a printer beacon in Boyd even though it is not "close" to us.

Beacon Structure

```
typedef struct identification {
     // Identify who we are
char name[32];           /*Name of the current client*/
     // Specify how we can be contacted.
in_addr_t location;    /* IP address of the client */
in_port_t port;        /* port where the client is listening */
     // Specify my credentials
char key[32];          /* My authentication key. Unused for
                          this project */
     // Specify how to talk to us. Unused for this project
unsigned int type;  /* whether we talk XML, HTTP, Corba protocols */
unsigned int length;    /* Length of protocol specific data
                           that follows in buf */
void *buf;             /* Protocol specific buffer */
} identification_t;
```

P2P Beacon identity exchange scheme

- Beacons use broadcast/multicast to find other beacons
  - When you broadcast a packet, every other computer in your network receives your query
  - When you multicast your query, only interested beacons that are a member of a multicast group receive the query
  - E.g. gnutella

Beacon service

- Beacons will provide the following service in their service port (chosen by you):
  - **open(passwd)** All the requests to a beacon should be preceded by the open function. We assume that any *passwd* is always valid.
  - **get(token, key)** This service will send the *value* associated with a given *key*. The key should be among the keys listed in the *list* service. Requests for a key that is not available should be denied.
  - **set(token, key, value)** This service will associate the *value* with the *key*. Existing values are overwritten with the new contents.
  - **list(token)** This service will list all the keys that are available at the beacon using earlier *set* operations.
  - **close(token)** This service will end the session with this particular beacon.

Assignment 1

- You are free to choose any implementation language/ mechanism. For ease of grading, please do not use Java GUI's for your program.
- Sample code from Richard Stevens book (url in handout)
- Submit code and a report detailing your approach, compilation instructions and how it solves:
  - Interoperability
  - Scalability
  - Consistency
- Remember: I will not accept late home works