Buddy Tracker

CSE 403 - Software Engineering
Spring 2004

~ TEAM MEMBERS ~
Darby Wong / Brad Woodward - User Interface
Anthony Wu - Data Structures & Integration
Shengli Zhou / Joseph Lai - Server and Database
Devon Kim / Kanji Furuhashi - Location-aware services

Product Overview

- Why? - Evolutionary step in personal communication applications in ubiquitous computing environments
- What will be built? - Stable, user friendly client capable of messaging and locating buddies
- How will it be done? - Using high level .NET tools and provided web services to enable rapid development of features

Target Users

- 18-25 year olds - eager adopters of new technology
- small businesses - keep in touch with traveling employees
- parents - keep an eye on the kids
- many other possible uses, but we will not cater to those users to stay focused

User Interface

- Will use .NET UI toolkit to build our interface
- Login Screen
  - authenticate user to our service
  - option to auto-login to MapPoint account
- Three Views
  - map
  - chat & messages
  - user options

Map operations

- Tap and Hold points of interest for contextual info
- Single/Double tap for zooming in/out
- Single tap or tap and hold to scroll

Chat operations

- Drop down menu of current chat buddies for quick switching
- Text box for displaying conversations
- Text box for entering messages

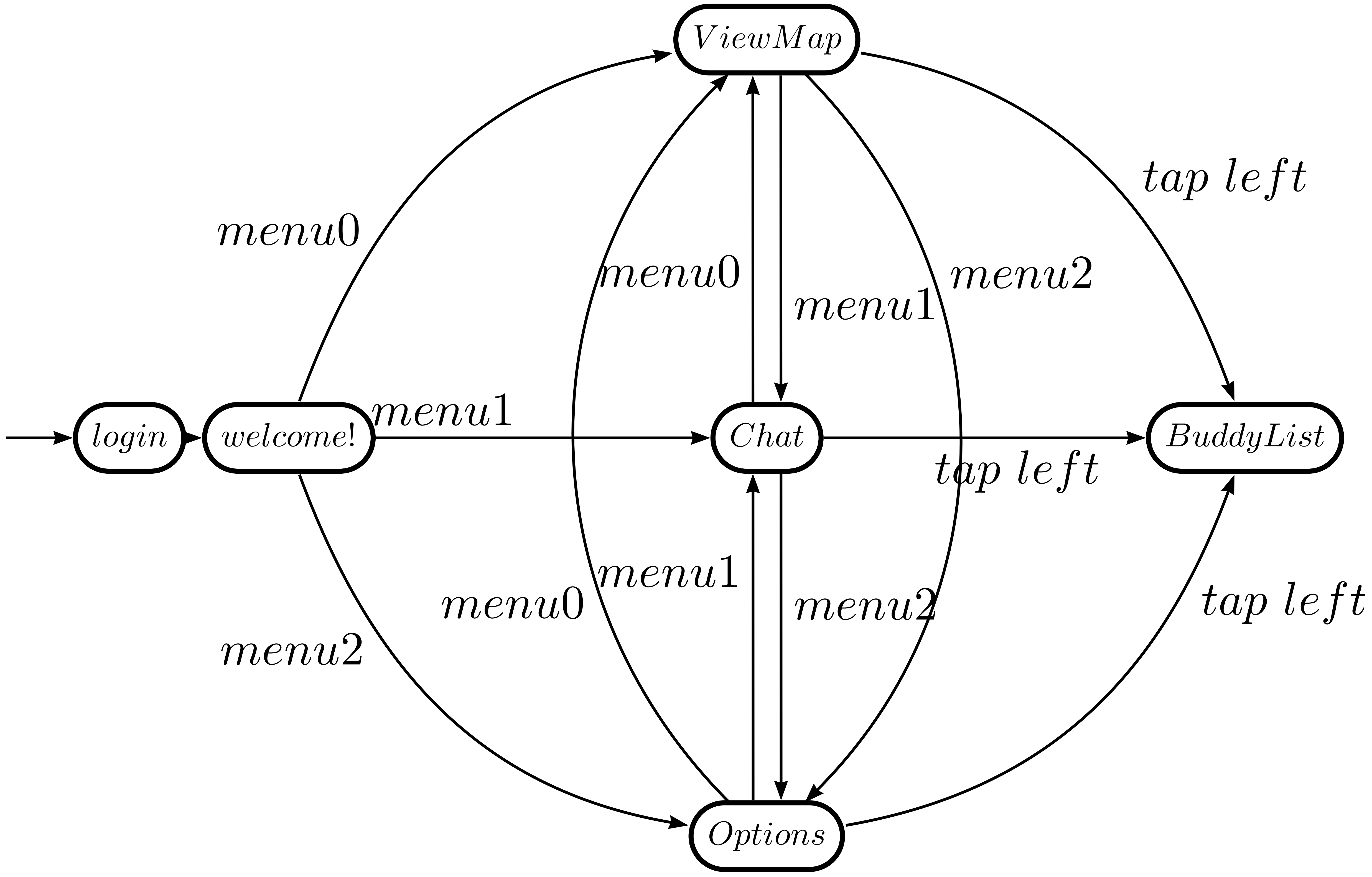
Buddy List

- Accessible from all screens
  - Hidden away in a auto-hide panel
- Organized in a tree structure
  - Groups ==> Indiv. buddies
- Quick privacy management through boolean on/off controls
- Buddy status indicators

Buddy List Operations

- Tap in map view to jump to map location
- Tap in chat view to bring up conversation
- Tap and hold buddy name to bring up buddy management options

UI NFA


login
welcome!
ViewMap
Chat
BuddyList
Options
menu0
menu1
menu2
menu0
menu1
menu2
tap left
tap left
tap left
menu0
menu1
menu0
menu2

Server Specs

- Direct dialog and chat between two clients
- Provide authentication and login security for subscribers
- Store database of user information / profile
- Provide web service for clients
- Accept update calls from clients

Server Specs (cont.)

- Handle heavy server traffic
- Avoid redundant SQL queries
- Keep index of frequently queried tables
- Encrypted packets w/RSA

Server Architecture Frontend

- Different types of requests inherit from superclass “Request”
- RequestListener sits in a loop to continually respond to client requests
- SQLInterface bridges front and back ends

Server Architecture Backend

- Microsoft SQLServer database
- User ID is the primary key
- Dynamic data will be stored in local data structure to prevent redundant queries

Location Service Specs

- Use MapPoint to provide location-aware services and data
  - Render various types of maps
  - Obtain points of interests
  - Security implemented by .NET-provided SOAP

Location Service Arch.

- User RenderServiceSoap class method GetMap to request a map with a known center point or GetBestMapView
- Wait for response from MapPoint
- Process the MapImage(s) returned by GetMap
- Use “pushpin” objects to integrate points of interest into the map

Client-side PlaceLab Specs

- Will use Intel's PlaceLab API to detect WiFi access points and to retrieve their MAC addresses
- Automatically update at specified intervals
- Client will trigger location updates
- No big security concerns for this type of public information

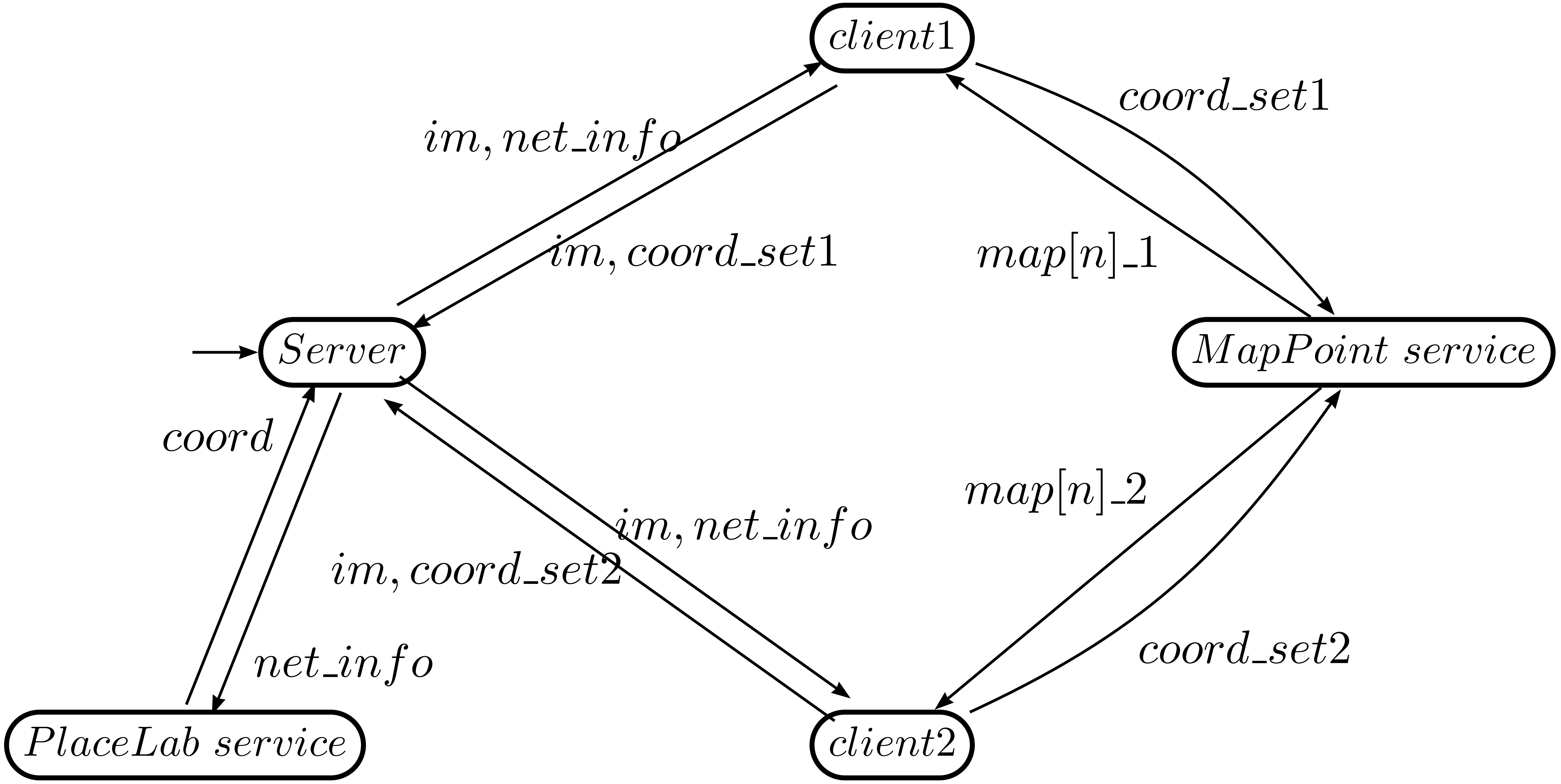
Server-side PlaceLab Specs

- Use Intel's PlaceLab database to translate from access point MAC addresses to location
- Server will sent the result location information to the server-side BuddyTracker web service
- GUI for this service is optional

PlaceLab Architecture

- PlaceLab is implemented with Java
- Will use XML RPC socket passing to communicate with our .NET components
- API:
  - org.placelab.core.WifiSpotter
  - org.placelab.core.Measurement
  - org.placelab.core.Tracker
  - org.placelab.core.WifiMapper
  - org.placelab.core.Beacon

System NFA



$client1$
im, net_info
$im, coord\_set1$
$coord\_set1$
$map[n]\_1$
$Server$
$MapPoint\ service$
$coord$
im, net_info
$im, coord\_set2$
$map[n]\_2$
$coord\_set2$
net_info
$PlaceLab\ service$
$client2$

Project Milestones

- May 16 - Feature Complete Alpha
  - User Interface Frozen
  - Server fully implemented
  - PlaceLab correctly locates clients
  - Two weeks of testing
- May 30 - Stable Beta, bug fixes
- June 6 - Polished, deployable application