Hyperglue: Infrastructure for Distributed, Pervasive Intelligent Environments

Stephen Peters
MIT CSAIL

January 31, 2006

Talk Overview

- Scenarios and Issues
- Thesis
- Design
- Evaluation

What is Hyperglue?

- Framework for intelligent environments (IEs), assisting in human-centered computing.

Scenario 1: Free Donuts

- (video)

Scenario 2: Practice Talk

- Alan planning a practice talk; schedules a time and invites a few people (Beth, Charlie)
- Beth in a shared conference room, holding a meeting – messages should appear on a laptop
- Charlie out of the office

The Central Issue

- Common feature of scenarios – dealing with communication and coordination of many different places and people.
- Most existing IE systems don't address this
  - focus on a single location, or a single user
  - single global registry systems
  - miss core issues of privacy and end-user control

Other Problems

- How do we find out what devices are available near Beth?
  - global registry of all devices?
  - constantly advertise Beth's location?
- How can Alan and Beth share control?
  - Does Beth have an off switch? Or Caller ID?
  - continually broadcast device preferences?

Other Problems (cont'd)

- How do we avoid conflicts with Beth's activity?
  - continually update availability status?
- Who decides how information is presented?
  - Email vs. phone?
- Access control?
  - Who has privileges to control devices?

Needs

- Preferences
- Understand people's locations and situations
- Respect privacy – don't globally advertise what people are doing
- Knowledge of people and relationships
- Adaptivity

Talk Overview

- Scenarios and Issues
- Thesis
- Design
- Evaluation

Thesis – Four Pillars

- Agent communications and structures that mirror the social and physical interactions they represent
- User preferences that fuel the environment's decisions about service mapping and resource management
- Awareness of world state, used to drive context-based preference decisions
- Semantic memory that describes preferences and resources

Agent Communications

- Modeled on social and physical interactions
- Simpler programming – direct translation to the way people communicate
- Abstraction barriers – allows creation of agent societies (cf. Minsky) that are associated with a real-world object; get treated as a unit

User Preferences

- Map user-defined qualities (“urgency”, “timeliness”) to utility functions
- Allows us to find numerical solutions for “what is the best method” questions

Awareness

- Context-based preference decisions
- When in the office, display locally; when out of the office, send email
  - When out of the office on a weekend, send SMS

Semantic Memory

- Need knowledge representation (KR) for resource and preference requirements
- Contextual information, group membership, relationships, ...
- Mark up representation with semantic information that describes what is being stored

Talk Overview

- Scenarios and Issues
- Thesis
- Design
- Evaluation

Design Components

- Agent system
- Society-based communication
- Decision making system using context-based preferences and a resource cost model
- Persistent semantic network knowledge base

Design Components

- Agent system
- Society-based communication
- Decision making system using context-based preferences and a resource cost model
- Persistent semantic network knowledge base

Agent Systems

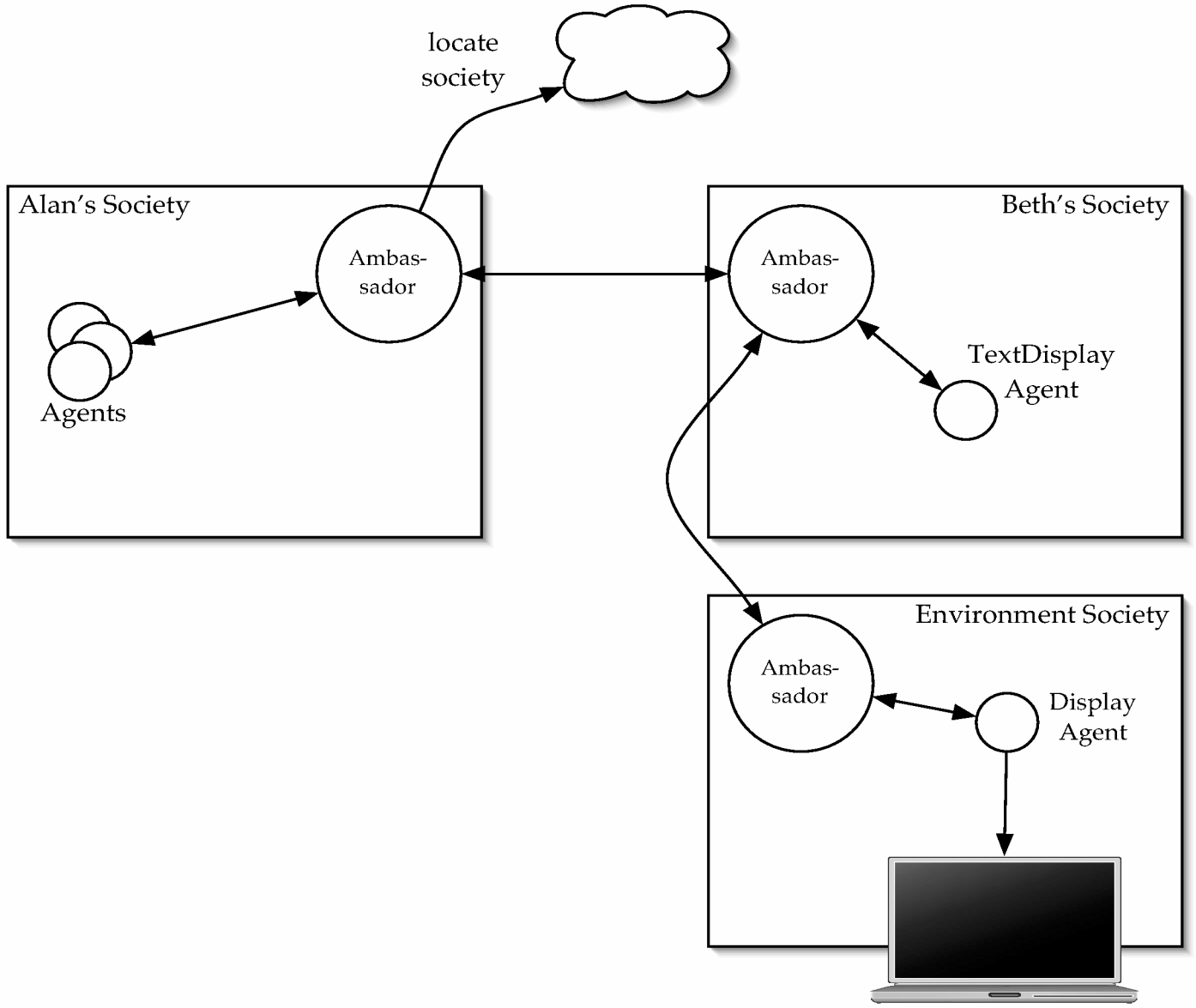
- OAA, Hive, iROS/EventHeap, Metaglue...
  - Name Mapping
  - Object Creation
  - Remote Communication
    - RPC, CORBA, RMI

Metaglue Overview

- Developed for AIRE (Coen, Phillips et al.)
- Agent identifiers including society
- Java-based, RMI communication
  - uses error-handling proxies for communication
  - dynamic replacement of agents
  - security hooks (cf. Kottahachchi)
- Notifications – multicast messaging
- GuiManager -- Separation of “user” agents from back-end

aire

Human-Centric Communication Model



Hyperglue Communication

- Agents get organized into *societies* with counterparts in the real world (Alan, Beth, Max, the conference room, offices, etc.)
- Each society contains agents for performing service mapping, resource management, semantic data management, and local catalogs
- Each society maintains an *ambassador* for communicating with other societies

Hyperglue Communication (cont'd)

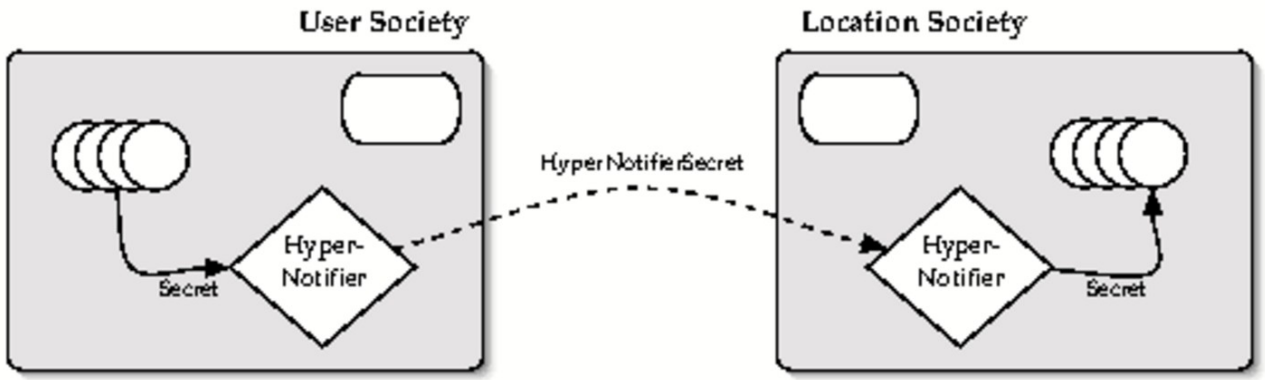
- Agents working for sender contact local ambassador
- Ambassador uses request to find remote society in an entity directory (HED), gets handle to remote ambassador
- Sender requests a service
- Remote ambassador returns several candidates, or negotiates the best possible
- Remote ambassador optionally talks to local environment in a similar fashion

Advantages

- No need for global list of all agents; just ambassadors
- Remote society can limit interaction as it sees fit
- All communication is local, so coding is easier
- Security checkpoint available in ambassador
- Possible to unite heterogeneous agent systems through public APIs

Publish-Subscribe in Society Model

- Need notifications to cross societal barrier
- Global clearinghouse society for handling notifications?
  - message congestion
  - need good addressing techniques, high-throughput notification handling
- Hyperglue solution – localized notifiers:



Group Societies

- Donuts scenario -- sending a message to a workgroup?
  - Don't want to leak membership lists to world
- Create society for the collection of people associated with a workgroup
  - ambassador becomes designated point of contact for the group members
- Ad hoc collections for collaboration
  - Chat application; who is the owner of the conversation?

Society Hierarchies (future work)

- Group societies serve as aggregates for child societies
- Can have groups of groups – natural focus for hierarchical HED search mechanism

Design Components

- Agent system
- Society-based communication
- Decision making system using context-based preferences and a resource cost model
- Persistent semantic network knowledge base

Service Mapping

- Focus on high-level goals as opposed to resources and procedures
  - users want more light, not “turn on lamp #45”
- Decision-theoretic approach to decision making
- Plan library for known goal types
  - what resources the plan requires
  - what quality of service is provided

Service Quality Parameters

- Services are abstract, so “quality of service” is also an abstract notion
- Should be specified at the level the user cares about
  - natural vs. artificial light
  - good vs. bad privacy (open drapes have bad)
- Service quality parameters used to create utility functions

Utility functions

- Provide *ceteris parabus* preferences for parameters user cares about
  - "Other things being equal, I prefer natural light to artificial."
- Turn preference rules into a utility function
- Function returns numerical value representing *utility* of a given set of parameters

Function Generation

- Modified McGeachie-Doyle algorithm
  - Create graph of all possible parameter values
  - Draw arcs between nodes as specified by the preference rules
  - Calculate node value based on distance to leaf node

Modifications to McGeachie-Doyle

- Multivariate parameter values instead of boolean
  - "soft", "medium", and "bright" illumination, not just "good" or "bad"
  - causes $\log_2 |E|$ nodes for each parameter with E values
- Allow for preference weighting
  - "Natural light is twice as good as bright light."
  - Doesn't really need to be exact, but allows user to weight their own preference set
  - adds weights to graph arcs

Using Utility Functions

- Generate functions based on user preferences
- Use service request to find all possible service plans that satisfy the request
- For each plan, use functions to evaluate based on how well the plan's resources satisfy the parameters
- Scale utility and offset by “cost” of using resources to get a net benefit
- Order the plans by net benefit, and return the best

Contextual Preferences

- Natural light might be better in the morning, but artificial in the afternoon
- Creates complex rules:

```
((source natural time-of-day morning >> source artificial time-of-day morning)
 (source artificial time-of-day afternoon >> source natural time-of-day afternoon))
```

- Solution: set up preference sets based on context (location, time of day, current task)
  - Use current context to index into user's preference sets
  - Multimethod dispatch

Extension to Utility Algorithm

- Parameters are often under-specified
- Calculating utility can return a set of bounds, describing the possible minimum and maximum values
- Given a hierarchy of services and sub-services, these values can be used to perform a depth-first branch-and-bound search to simplify calculations

Design Components

- Agent system
- Society-based communication
- Decision making system using context-based preferences and a resource cost model
- Persistent semantic network knowledge base

Semantic Network-Based Knowledge Representation

- Information on system's users
- Presence information (Beth in a conference room, Max is elsewhere)
- Context (Beth's room is currently in a meeting)
- Preferences
- Device information
- Other
  - Meeting attendees, agenda items, discussion points...

SEMANTIC Design

- Other Semantic Network implementations
  - High-performance databases
- Built to provide a storage framework for Metaglue that was
  - Persistent
  - Java-based
  - Reasonably fast
  - Transparent
  - Inferencing platform
  - More robust than other Java persistence solutions

Persistence

- Backed by MySQL (or PostgreSQL)
- Simple storage – tuples based on object identifiers, keys, values

<10342, is-a, semantic.Person>
<10342, name, “Bruce Wayne”>
<10342, email, bwayne@gotham.gov>
<10342, birthdate, “May 27, 1939”>

Java-Based

- All SEMANTIC types inherit from UniqueID superclass
- Uses reflection to determine key names from Java fields
- Special handling for primitive types, collections, some Metaglue types.
- Easy mechanisms for searching all objects:
  - Person test = new Person();
    test.setEmailAddress("bwayne@gotham.gov");
    List l = test.findSimilarObjects();

Query Languages for Inferencing

- Using Prolog-style pattern-matching code to find linked objects:
  - (ask '((?x is-a semantic.Group)
    (?x name MyGroup)
    (?y is-a semantic.MemberOf)
    (?y fromNode ?x)
    (?y toNode ?z)))
- Also Java-based versions for Lisp-wary
- Triggers available for callbacks to agents when updates occur

Talk Overview

- Scenarios and Issues
- Thesis
- Design
- Evaluation

aire

Evaluation

- How well can this system implement the scenarios?
- Do we protect people's ownership rights?
- Is the organization “natural”? How difficult would it be to modify the system for a new paradigm?

Summary

- Designed a system based on
  - Agent communications and structures that mirror the social and physical interactions they represent
  - User preferences that fuel the environment's decisions about service mapping and resource management
  - Awareness of world state, used to drive context-based preference decisions
  - Semantic memory that describes preferences and resources