GVDS Assets

- In coordination models exploiting the notion of GVDS:
  - The association between the coordination context contributed by a given agent and the agent itself is now made explicit
  - The resulting style of coordination draws a distinction between the information immediately available to an agent and the one that can be requested from others
  - Still, the benefits of coordination, e.g., the decoupling of communication from behavior, are retained
- Hence, GVDS fosters a coordination style where:
  - coordination is defined entirely in terms of the coordinated agents, without reliance on some external entity
  - the coordination context is automatically and dynamically reconfigured
  - coordination is achieved through local actions that have a global effect
- The conjecture is that these characteristics are going to:
  - simplify the task of building (and reasoning about) applications that …
  - … are built out of autonomous components …
  - … whose relationships are dynamically and frequently reconfigured

Design Alternatives

- Choice of the data structure
  - Sets, bags, trees, graphs, matrices, …
  - May affect the efficiency and/or complexity of the implementation
- Choice of operations
  - Local vs. global
  - Query vs. manipulation
  - Proactive vs. reactive
  - Synchronous vs. asynchronous
- Choice of the partitioning/merging criteria
  - Superposition, union, composition, …
- Choice of the enabling condition for sharing
  - Based on connectivity
    - connectivity over space vs. connectivity over time for physical mobility
    - co-location for logical mobility
  - Possibly augmented by application constraints
    - e.g., to deal with security, or with specific application constraints

Design Alternatives – cont'd

- Degree of symmetry and transitivity
  - Is everybody "seeing" the same content?
- Degree of atomicity
  - Important when determining the semantics of operations, and their relationship to sharing
  - Determines the extent to which one can treat the GVDS as a "local" data structure
  - Simplifying the programmer's chore vs. delivering an efficient implementation
- Degree of consistency
  - Given two agents, how far can their perception of the GVDS drift?
  - The answer to this question often implies the use of caching and replication schemes
- Degree of knowledge about the system configuration
  - System information can be represented in a GVDS, too
- Degree of persistency
  - If a portion of the system is known to be stable, how can we exploit it?

Research Issues

- What is the good balance to strike among the design alternatives?
  - Relationship with other middleware approaches and results
- Is there a "unifying theory" of GVDS?
  - Is it possible to separate the issues related with distribution from those intimately connected to the data structure chosen?
  - A positive answer could lead to a middleware supporting instantiations of GVDS with different data structures
- What is the relationship between GVDS and security?
- What is the impact of the GVDS abstraction on formal reasoning and verification?

GVDS Summary

- Global virtual data structures are a novel coordination paradigm targeted at highly dynamic environments
- GVDS is not meant to be a new model by itself: instead, it is meant to be the driving concept behind a new family of coordination models
- While some incarnations of GVDS are already available, only a fraction of the design space has been explored so far

Outline

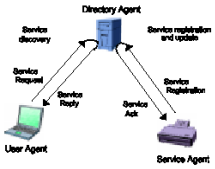
- Introduction and Major Issues
- Commercial Mobile Middleware
- Next-Generation Mobile Middleware
- Case Study – LIME
- Middleware for Wireless Sensor Networks
- Open Issues and Future Directions

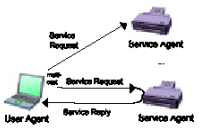
Service Discovery

- When enter a new area, problem is how to identify and connect with services
- Part of adaptability, and part of almost every mobile middleware system
  - e.g., replication for ad hoc collaborative environments used SD for identifying other devices.
  - e.g., Spectra (part of Aura) uses service discovery to find other computational devices to spread computation
- Goal: not to re-implement service discovery for every system (including middleware), but re-use a general solution
  - SLP, UPnP, SDP (Bluetooth), Jini

Service Location Protocol (SLP)

- Standards driven – developed within IETF
- Decentralized, lightweight, scalable and extensible protocol for service discovery within a site.
- Directory agent (optional), Service agent, User agent
- Active and passive DA discovery
  - DHCP option 78: distribution of DA addresses
- Services identified by URL
  - Service templates – exploit standardized vocabulary
  - Connection mechanism independent from SLP





Universal Plug and Play (UPnP)

Microsoft — Service Discovery

- Microsoft lead industry consortium (617 members as of September 2003)
- Discovery of devices reachable through TCI/IP
- No central service registry
  - Control points (e.g., your Windows ME laptop)"search" for devices (multicast query)
  - Control achieved through web page or specialized interface (e.g., control your new DVD player through its web page, rather than the remote control)
- Devices *multicast* their services
- XML based service descriptions

Service Discovery Protocol (SDP)

Bluetooth — Service Discovery

- Focuses on service discovery for ad hoc networks
  - No central service registry
- Three search types (on existing connection!):
  - By service type
  - By service attribute
  - Or service browsing
- Can retrieve attributes that detail how to connect to the service (protocol identifier, and protocol-specific parameters)
- Does not "compete" with other discovery protocols, but can have them layered on top to provide additional functionality (e.g., leasing, security, etc.)

Jini Service Discovery

- Sun Microsystems, open source
- Mechanisms for discovery of lookup server (required component)
  - *Multicast request protocol*
    - Browse for local, nearby lookup services, UDP every 5 sec.
  - *Multicast announcement protocol*
    - Lookup services advertise existence, clients listen
    - Used by new lookup services or after a failure recovery
  - *Unicast discovery protocol*
    - Join a specific community, either known, or discovered by above
- Result of service discovery is Jini object code for direct access to the server

SD Summary

- Many competing industrial standards
- Tradeoff between centralized registry and "noise" of protocol
  - Multicast increases network traffic, which is a concern for mobile, battery-powered devices
- Key component often missing is the vocabulary used for service description
  - XML is extensible, but still need shared vocabulary in order to work together

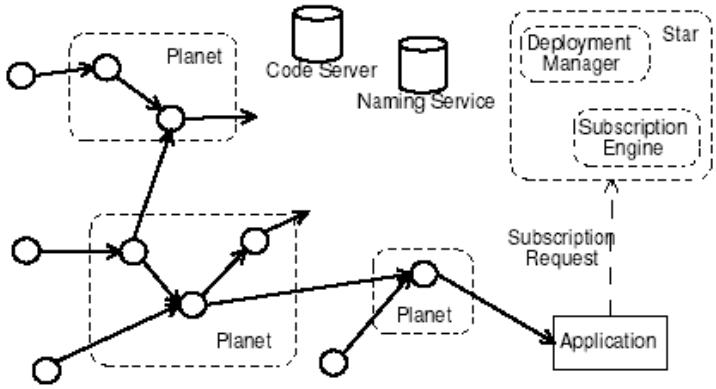
Event-Based

- Publish-Subscribe systems are asynchronous, implicit, multi-point, and peer-to-peer in communication style
- This style is suited to both traditional distributed systems and to mobile systems
- Clients and publishers are decoupled, and the infrastructure can be distributed

Solar

- Context-information collection, aggregation, and dissemination (data fusion)
  - Operator graph representation of computation allows decomposition and reuse of context aggregation primitives
  - Examples: filters, transformers (lookup mechanisms)
- Applications register operations with centralized server "Star" (the centralized dispatcher)
- Computation farmed to available hosts (planets)
  - Directly deliver events to applications

Solar Architecture



Jedi

- Provides a scalable, distributed, content-based event dispatcher
- Supports mobile agents that connect to a dispatcher
  - Clients disconnect
  - The old event dispatcher stores events for disconnected clients
  - When client reconnects, stored events are transferred and delivered
  - Partial order of events is guaranteed

Reconfiguration of PS Infrastructure

- A distributed dispatcher is potentially useful in a mobile ad hoc environment.
  - Disconnected nature of PS matches requirements and open nature of MANET
  - Each host becomes a node in the dispatching tree
- Current solutions do not address reconfiguration of the infrastructure itself
  - Reconfiguration is required when nodes fail, when links disappear, when the PS tree is partitioned
- We are examining solutions that adapt an MAODV-like tree reconstruction algorithm for fixing the PS infrastructure in MANET environments

PS Summary

- Although pub/sub features match the mobile environment well, the challenging issues have not all been addressed
- There seem to be strong connections between the tree of pub/sub and the tree of multicast MANET solutions
  - We are exploring this connection, and hope that we can develop pub/sub successfully in a MANET

Object-Oriented

- Communication among distributed objects
- Applicability of existing OO middleware to mobile setting is limited
  - Heavy computational load (need to be made lighter!)
  - Synchronous communication
  - Transparency (might be difficult to be light and adaptive without optimizing considering application needs)
- Risk to penalize performance, generate high costs and lead to unusable and non-scalable solutions

Rover

- Aims to support both transparency and awareness, and is based on two key concepts:
  - ***Relocatable dynamic objects (RDO):*** mobile objects (code and data) that can be dynamically loaded into a client from a server, or vice versa
    - Essentially used to reduce communication, e.g., to perform semantic compression
    - Methods contain also conflict resolution procedures
  - ***Queued remote procedure calls (QRPC):*** provides the core communication mechanisms among RDOs, by providing a non-blocking RPCs
- A client-server architecture is assumed, no mobile-mobile interaction supported
- The toolkit has been tested on the field by porting several common applications (e.g., `exmh`, `ical`, and other X11 applications)
  - A speed-up of about 20% has been measured, but mostly for scenarios with very low bandwidth

Using RDOs

- A Rover application is made of RDOs, that are fetched lazily from servers
  - Each RDO has a "home" server
- RDOs are cached and replicated
  - the primary copy is held at the server
  - a prefetching scheme is employed to fill the object cache while connected with objects that are selected by the application
- Updates to local copies of RDOs are done immediately, and marked tentative
  - they are lazily propagated to servers using QRPC
  - It is up to the applications to choose to use tentative RDOs
- Servers are in charge of detecting and resolving conflicts
  - this is typically achieved through methods invoked on the primary copy, by keeping version vectors for RDOs, and by examining invocation logs
  - A wide range of consistency control techniques are provided, although primary-copy, tentative-updates are preferred
- The system itself is made of RDOs, that can be loaded on-demand according to the features of the current environment

QRPC

- QRPC is used to fetch RDOs, as well as to enable the communication between client and server, necessary to keep the object copies consistent
- Invocation of a QRPC on an object returns immediately, by yielding a *promise* to the object
  - Promise object can be used to proactively check whether a result has arrived, to suspend waiting for the result, or to register callbacks
- If a mobile host is disconnected between sending the request and receiving the reply, the server will periodically try to contact the mobile host and deliver the reply
  - Different communication channels can be used for request and reply
  - Support of disconnected operation
- The queued RPCs, issued by the client, are filtered by a network scheduler module, that:
  - may decide to deliver them in a non-FIFO order, to suit the application-specified priorities and needs
  - May compress the requests, and/or send them in batches

ALICE

- Architecture for Location-Independent CORBA Environments
- Mobile hosts act as servers AND/OR clients
  - Interoperate transparently with standard CORBA applications
  - Does not require support for Mobile IP
  - Hides broken TCP connections
  - Integrates service handoff with mobile handoff
- Uses IIOP
  - Minimum ORB protocols to exchange messages between clients and servers

Object Oriented Summary

- Object oriented software develop is a common practice in distributed environments
- Supporting object relocation and interaction in the mobile environment eases application programming as long as the resulting semantics remain well-understood

Transactions

- Sequence of operations clustered together complying with ACID properties
  - Atomicity: execute completely or not at all
  - Consistency: lead from one consistent state to another
  - Isolation: isolation from concurrent transactions
  - Durability: once completed, changes cannot be undone
- Mobile Transactions:
  - Support mobile clients performing transactions on stationary servers
  - ACID may not be enforceable – high overhead

Kangaroo

- Support for transactions by mobile entities that may not start and end at the same location
- Support long-lived, hopping transactions
- Two modes
  - Split: failure of transaction at base station does not imply failure of complete transaction (previous hops)
  - Compensating: failure of a transaction at base station implies failure of complete transaction

Transaction Summary

- Most industrial transactions (e.g., Oracle Mobile agents, IBM Gold Rush) are satisfied with simple transactional support
  - Submitting remote transactions
  - Optimistically performing transactions, reporting conflicts
- Kangaroo transactions address a single form of transaction for mobile users
  - Do not solve all issues, nor provide a clean model to replace the ACID concepts
- Issues not addressed
  - Transactions and commitment in mobile ad hoc networks
  - What to do when the database itself is on a mobile device

Transport Layer

- TCP/IP does not perform well across wireless links
  - Packet loss on the lossy link cause excessive slow-start at the fixed host
- Solutions exist, including I-TCP (indirect TCP, from Rutgers U.), that remove TCPs end-to-end guarantee, but significantly improve performance for bulk transfers (1.5 times)
- Another significant problem is naming
  - All components, especially those in a disconnected MANET should not be assumed to have an IP address
  - They may not even have a distinct identifier

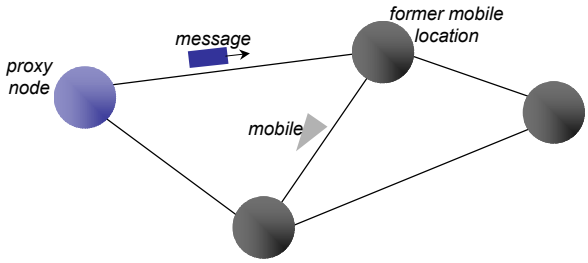
Underlying Support Algorithms

- Middleware often assumes certain existing technologies
- Often these technologies themselves are non-trivial, and deserve separate focus
- POMC – Principles of Mobile Compuitng
  - Theoretical workshop in 2001, 2002, and 2003
  - Guaranteed message delivery
  - Mutual exclusion algorithms
  - Group communication for mobile participants
  - K-cluster formation

Reliable Message Delivery to Mobile Components

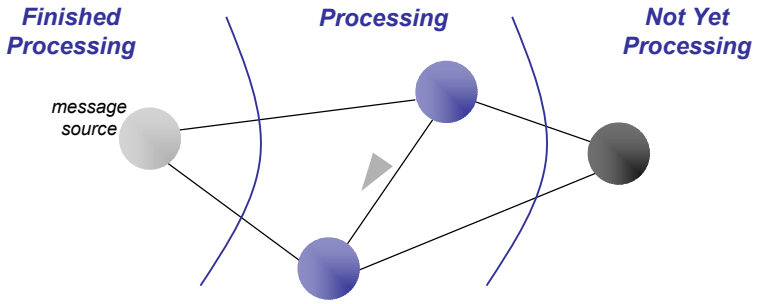
- Applications need to communicate control and data information to mobile components
- We must have a mechanism to ***reliably deliver*** these messages (unicast or multicast)
- Guaranteeing the delivery of a message to a moving unit is challenging even under the assumption of a fault-free network

Proxy



- During movement, information at the proxy is out of date and messages will chase the mobile unit
- Similar to the approach of Mobile IP

Idea: Adapt Distributed Snapshot to Message Delivery



- Snapshots provide a consistent image of the network state
- Key property: ***every message in the system appears in exactly one local snapshot***

From Snapshots To Message Delivery

Distributed Snapshot	Snapshot Delivery
Node	Mobile agent server/ Base station
Message	***Mobile unit***
Token	Application message
Record message	***Deliver application message***
Local snapshot terminates	Application message deleted

Properties of Snapshot Delivery

- Exactly-once delivery independent of mobile movement
- Can be trivially extended to multicast
- Storage required at each node for only one message round trip time
- Overhead of one message per edge
- Network neighbors must be known in advance, but extensions allow the underlying network to be dynamic

Other Algorithmic Work

- This approach adapts distributed computing algorithms to mobile computing
  - Another solutions: using diffusing computations to track mobile units, layered with message delivery
  - Other promising algorithms: randomized algorithms, self-stabilizing algorithms, epidemic algorithms
- Other approaches include developing algorithms similar to those of distributed computing by shifting the burden of computation onto fixed infrastructure
  - Checkpointing: Acharya'94
  - Reliable multicast message delivery: Acharya'96
  - Causal event ordering: Prakash'97
  - Termination detection: Matocha'98

Outline

- Introduction and Major Issues
- Commercial Mobile Middleware
- Next-Generation Mobile Middleware
- Case Study – LIME
- Middleware for Wireless Sensor Networks
- Summary and Open Questions

Summary

- Middleware for mobile computing provides abstractions for easing the development process
- Commercial middleware is targeted toward the first step of mobility, providing service access to mobile devices
- Major issues and approaches in research include
  - Replication, Adaptation, Service Discovery, Event-based, Object-Oriented, Transactional, Transport Layer, Algorithmic, Data Sharing
  - Not discussed issues include reflection, security, …

Questions:

- Will there be / should there be a single middleware for mobile computing?
  - Not all environments have the same demands and needs
  - Can there be a composable middleware that allows designers to pull in only the aspects that they need?
- Will the need for mobile middleware diminish as wireless networks become faster?
- Can middleware be shared among applications for:
  - Nomadic computing
  - Mobile ad hoc computing
  - Sensor networks