Agents and Electronic Commerce

Munindar P. Singh
singh@ncsu.edu
http://www.csc.ncsu.edu/faculty/mpsingh/

(Slides coauthored with Michael N. Huhns)

Outline

- Electronic commerce
- Agents
- Agents for electronic commerce
- Electronic commerce for agents
- Synthesis

Agents for EC

- Basic match
- Service location
- Service evaluation
- Negotiation
- Markets

EC for Agents

- Commerce as a metaphor for
  - cooperative activity
  - resource allocation

Kinds of Networks

- Internet
- Intranet: network restricted within an enterprise
- Extranet: private network restricted to selected enterprises
- Virtual Private Network (VPN): a way to realize an intranet or extranet over the Internet.

Open Environments: Characteristics

- Cross enterprise boundaries
- Comprise autonomous resources that
  - Involve loosely structured addition and removal
  - Range from weak to subtle consistency requirements
  - Involve updates only under local control
  - Frequently involve nonstandard data
- Have intricate interdependencies

Open Environments: Technical Challenges

- Coping with scale
- Respecting autonomy
- Accommodating heterogeneity
- Maintaining coordination
- Getting work done
  - Acquiring, managing, advertising, finding, fusing, and using information over uncontrollable environments

Electronic Commerce

- Introduction
- B2C
- B2B
- Projections
- Challenges
- Tasks from different perspectives
  - Merchant
  - Customer
  - Dealmaker

EC Applications: 1

Interestingly, most applications of agents relate to electronic commerce.

- Information gathering, presentation, and management
  - University of Michigan auctions access to information for digital libraries
- Personalization
  - Firefly, AgentSoft, Verity, and Amulet offer agents that adapt to users' information needs and proactively retrieve + organize targeted information
  - Siemens (Germany) provides personalized telecom services
  - Amazon and Barnes & Noble help customers purchase books on-line
- Business processes and optimization
  - Sainsbury's Supermarkets (UK) simulates customers with agents
  - Inventory management and logistics

EC Applications: 2

- Logistics
  - US Postal Service includes smart-card agents on packages to track deliveries
- Energy distribution and management
  - Sydkraft (Sweden) controls electricity distribution
- Telecommunications
- Intelligence, in general
  - France Telecom and Deutsche Telekom diagnose circuit faults and route message traffic
  - Smart vehicles and smart highways
  - Raytheon/TI sensors cooperate in target detection

Properties of EC Environments

Autonomy

Independence of users.

- Political reasons
  - Ownership of resources
  - Control, especially of access privileges
  - Payments
- Technical reasons
  - Opacity of systems with respect to key features, e.g., precommit

Heterogeneity

Independence of component designers and system architects.

- Political reasons
  - Ownership of resources
- Technical reasons
  - Conceptual problems in integration
  - Fragility of integration
  - Difficult to guarantee behavior of integrated systems

Locality

- Global information (data, schemas, constraints) causes
  - Inconsistencies
  - Anomalies
  - Difficulties in maintenance
- Global information is essential for coherence
  - Locations of services or agents
  - Applicable business rules
- Relaxation of constraints works often
  - Obtain other global knowledge only when needed
  - Correct rather than prevent violations of constraints: often feasible
  - When, where, and how of corrections must be specified, but it is easier to make it local

Dynamism

- Entities change dynamically in their
  - Composition
  - Behavior
  - Interactions

Simple B2C Example

Suppose you want to sell cameras over the web, debit a credit card, and guarantee next-day delivery

- Your application must
  - record a sale in a sales database
  - debit the credit card
  - send an order to the shipping department
  - receive an OK from the shipping department for next-day delivery
  - update an inventory database

B2C Challenges

- Problems
  - What if the order is shipped, but the debit fails?
  - What if the debit succeeds, but the order was never entered or shipped?

Approach for Closed Environment

- Transaction processing (TP) monitors (such as IBM's CICS, Transarc's Encina, BEA System's Tuxedo) can ensure that all or none of the steps are completed, and that systems eventually reach a consistent state
- But what if user's modem is disconnected right after he clicks on OK? Did order succeed? What if line went dead before acknowledgement arrives? Will the user order again?
- The TP monitor cannot get the user into a consistent state!

Approach for Open Environment

- Server application could send email about credit problems, or detect duplicate transactions
- Downloaded Java applet could synchronize with server after broken connection was reestablished, and recover transaction; applet could communicate using http, or directly with server objects via CORBA/IIOP or RMI
- If there are too many orders to process synchronously, they could be put in a message queue, managed by a Message Oriented Middleware server (which guarantees message delivery or failure notification), and customers would be notified by email when the transaction is complete

EC Challenge: Teamwork

To perform even simple trades reliably we must ensure that the parties to an interaction agree on its current state and where they desire to take it

- Requires elements of teamwork and collaboration through
  - persistence of the computations
  - ability to manage context
  - retrying

EC Challenge:
Information System Interoperation

Supply Chains: manage the flow of materiel among a set of manufacturers and integrators to produce goods and configurations that can be supplied to customers.

- Requires the flow of information and negotiation about
  - product specifications
  - delivery requirements
  - prices

EC Challenge:
Distributed Decision-Making

Manufacturing Control: manage the operations of factories

- Requires intelligent decisions to
  - plan inflow and outflow
  - schedule resources
  - accommodate exceptions

EC Challenge: Autonomous Interests

Automated Markets to conduct trades under various kinds of mechanisms.

- Requires abilities to
  - set prices
  - place bids
  - accept or reject bids
  - accommodate risks

EC Challenge: Personalization

Consumer dealings to make the shopping experience a pleasant one for the customer.

- Requires
  - learning and remembering the customer's preferences
  - offering guidance to the customer (best if unintrusive)
  - acting on behalf of the user without violating their autonomy

EC Challenge: Service Location and Assessment

Recommendations to help customers find relevant and high quality services.

- Requires a means to
  - obtain evaluations
  - aggregate evaluations
  - find evaluations

EC Challenge: Exception Conditions

Virtual Enterprises to construct enterprises dynamically to provide more appropriate, packaged goods and services to common customers.

- Requires the ability to
  - construct teams
  - enter into multiparty deals
  - handle authorizations and commitments
  - accommodate exceptions

Agents and MAS

- Background
- Principles
- Technologies

Tremendous Interest in Agent Technology

Evidence:

- 400 people at Autonomous Agents 1998 in Minnesota
- 550 people at Agents World 1998 in Paris

Why?

- Vast information resources now accessible
- Electronic commerce
- Ubiquitous processors
- New interface technology

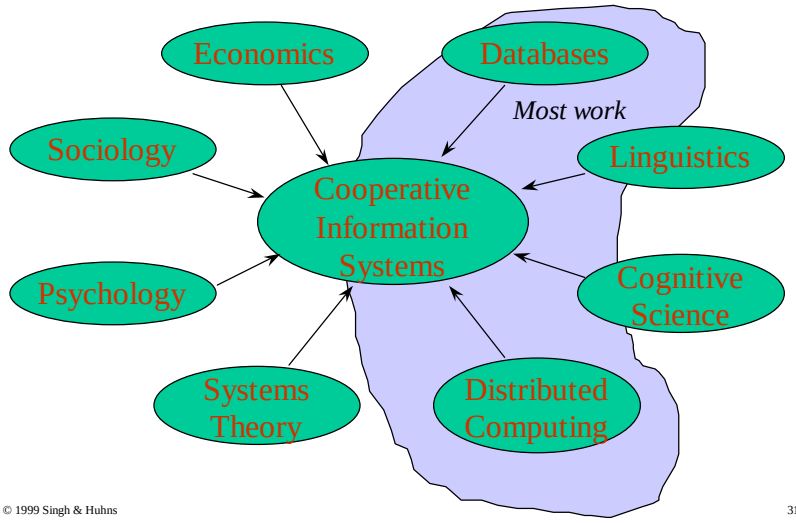
What is an Agent?

- The term agent in computing covers a wide range of behavior and functionality.
- In general, an agent is an active computational entity
  - with a persistent identity
  - that can perceive, reason about, and initiate activities in its environment
  - that can communicate (with other agents)
- It is the last feature that makes agents a worthwhile metaphor in computing

Attributes of MAS

- Decentralization
- Complex components, often best described at the knowledge level
- Adaptive behavior
- Complex interactions
- Coordination

Heritage of MAS



Characteristics of MAS Applications

- Inappropriate for conventional distributed computing:
  - local data may be incomplete or inaccurate
  - local problem solving is prone to error
  - the nodes are complex enough to be agents
- Inappropriate for conventional AI:
  - local autonomy is critical
  - strong semantic constraints exist among agents

Benefits of MAS: 1

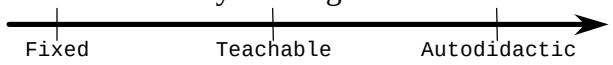
- Due to Distributed Computing
  - Modularity: many problems are inherently decentralized; large problems are easier if they are decomposed and distributed
  - Speed
  - Reliability

Benefits of MAS: 2

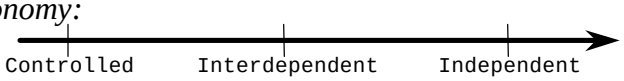
- Due to AI
  - Maintaining systems becomes harder as they scale up
    - mix and match parts: easy, if they were designed to cooperate
    - extend capabilities: easier if you can just add more players to a team
  - Knowledge acquisition: use many narrow experts
  - Reusability
  - Ease of requirements acquisition
  - Platform independence

Dimensions of MAS: Agent

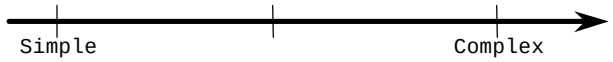
Dynamism is the ability of an agent to learn:



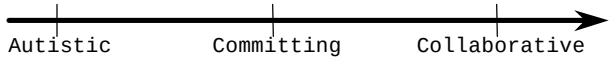
Autonomy:



Interactions:

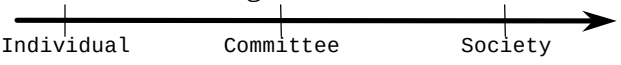


Sociability (awareness):

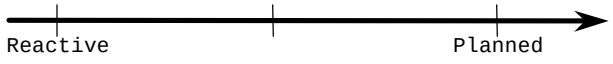


Dimensions of MAS: System

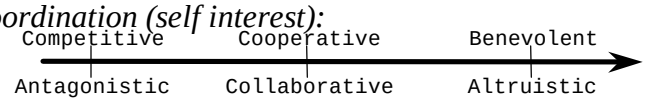
Scale is the number of agents:



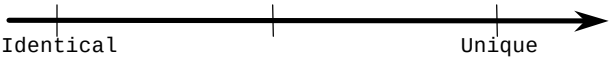
Interactions:



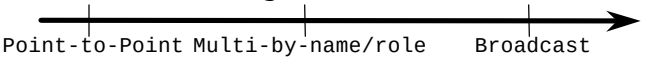
Coordination (self interest):



Agent Heterogeneity:



Communication Paradigm:



Basic Problems of MAS

- Description, decomposition, and distribution of tasks among agents
- Interaction and communication among agents
- Distribution of control among agents
- Representation of goals, problem-solving states, and other agents
- Rationality, consistency maintenance, and reconciliation of conflicts among agents

Pure Agent Approaches

Materiel Management

Distributed, autonomous agents enable

- Robust inventory control
- Decentralized logistics

based on a "Commuter" approach

The Logistics Problem

Efficiently and rapidly moving large quantities of equipment, personnel, and supplies is a massive problem in planning and scheduling

Current solutions to this problem are centralized and top-down, based on hierarchical decomposition

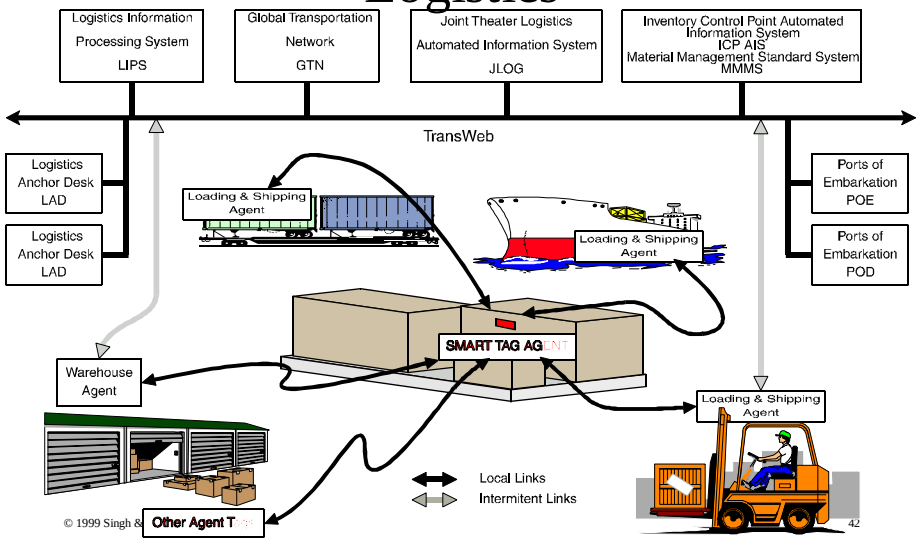
- Such solutions are
  - extremely complex
  - unable to deal with unforeseen delays and breakdowns
  - unable to take advantage of synergism among tasks
  - susceptible to single-point failures
  - difficult to change once started!

The USC Approach: a "commuter" paradigm

Imagine that each item of materiel is an intelligent agent whose sole objective is to reach its assigned destination. Just like a person commuting to work, this agent would *dynamically*

- decide its means of conveyance
- contend for storage and transportation resources
- avoid or resolve conflicts with other agents
- make *local* decisions as it wends its way through a distribution network

Agent-Based Decentralized Logistics



Hardware/Software Architecture

- Each item of materiel would have a "smart card" containing
  - a mechanism for communicating locally and globally
  - a reasoning engine
  - a knowledge base with information about routes, conveyances, and ways to resolve conflicts
  - its objective, priority, needs, and relationships to other items
- Each part of the distribution network would have a scanner to interrogate and command the items

Technical Challenges

- Setting-up a deployment
  - centralized planning is still required to assign objectives, priorities, and responsibilities to the items
- Controlling a deployment: maximize responsiveness and minimize resource usage
  - a market approach would allow items to compete fairly and intelligently, while cooperating altruistically when appropriate
  - items would continually and optimally replan during execution
- Monitoring a deployment
  - intelligent items would be able to state when they might reach their destinations, not just where they are at the moment

Consistency Maintenance Applied to Consultation and Collaboration

What Is a TMS?

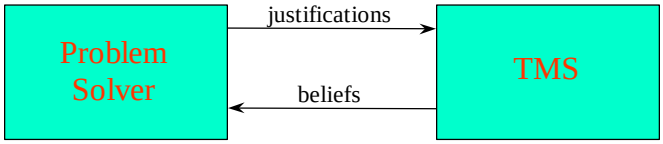
A truth maintenance system

- performs some form of propositional deduction
- maintains justifications and explains the results of its deductions
- updates beliefs incrementally when data are added or removed
- uses its justifications to perform dependency-directed backtracking

TMSs are important because they

- deal with atomicity
- deal with the frame problem
- lead to efficient search

Architecture of TMS-Based Agent



- The problem solver represents domain knowledge in the form of rules, procedures, etc. and chooses what to focus on next
- The TMS keeps track of the current state of the search for a solution. It uses constraint satisfaction to maintain consistency in the inferences made by the problem solver

Knowledge Base Integrity

- *Stability:* believe everything justified validly; disbelieve everything justified invalidly
- *Well-Foundedness:* beliefs are not circular
- *Logical consistency:* logical contradictions do not exist
- *Completeness:* a system will find a consistent state if it exists, or report failure

Problems arise when knowledge is distributed

Kinds of Inconsistency

- Both a fact and its negation are believed
- A fact is both believed and disbelieved
- An object is believed to be of two incompatible types, i.e., two terms are used for the same object
- Two different objects are believed to be of the same type, i.e., the same term is used for two different objects
- A single-valued fact is given more than one different value; e.g., (age Bob 7) and(age Bob 8)

Separate TMSs could be used for

- domain knowledge, control knowledge, know-what, and know-how

Degrees of Logical Consistency

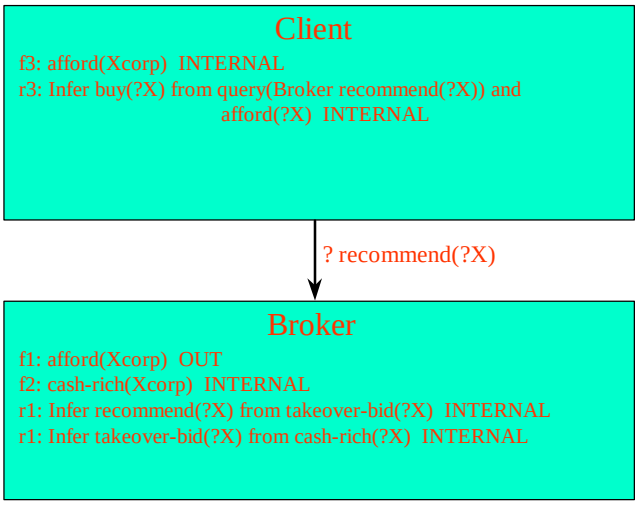
- *Inconsistency*: one or more agents are inconsistent
- *Local Consistency*: agents are locally consistent
- *Local-and-Shared Consistency*: agents are locally consistent and all agents agree about shared data
- *Global Consistency*: agents are globally consistent

The RAD DTMS maintains local-and-shared consistency and well foundedness

Distributed TMS

- Each agent has a justification-based TMS
- Each datum can have status OUT, INTERNAL (valid local justification), or EXTERNAL. A shared datum must be INTERNAL to one of the agents that shares it
- When a problem solver adds or removes a justification, the DTMS
- Unlabels data based on the changed justification
- Labels all unlabeled shared data
- Chooses labels for remaining unlabeled data; if this fails, it backtracks by unlabeling additional data and iterating

DTMS Example



DTMS Example (cont.)

DTMS Example (cont.)

Client

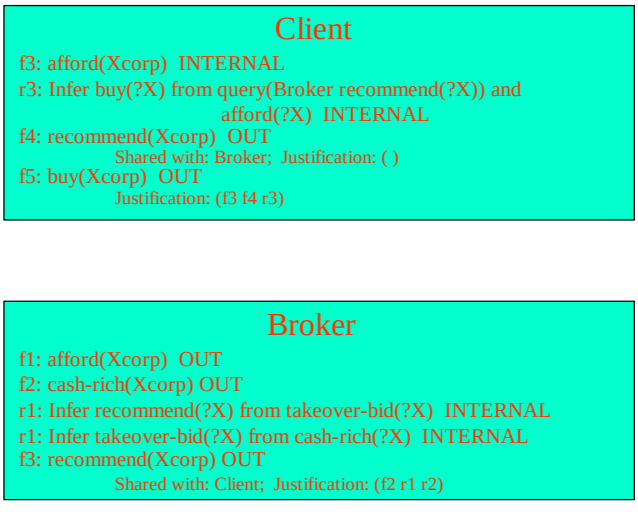
f3: afford(Xcorp) INTERNAL
r3: Infer buy(?X) from query(Broker recommend(?X)) and
afford(?X) INTERNAL
f4: recommend(Xcorp) EXTERNAL
Shared with: Broker; Justification: ( )
f5: buy(Xcorp) INTERNAL
Justification: (f3 f4 r3)

Broker

f1: afford(Xcorp) OUT
f2: cash-rich(Xcorp) INTERNAL
r1: Infer recommend(?X) from takeover-bid(?X) INTERNAL
r1: Infer takeover-bid(?X) from cash-rich(?X) INTERNAL
f3: recommend(Xcorp) INTERNAL
Shared with: Client; Justification: (f2 r1 r2)

DTMS Example (cont.)

DTMS Example (cont.)



Distributed ATMS

- Agents are locally, but not globally, consistent, based on a local ATMS
- Agent interactions are limited to result sharing
- Agents communicate only their own results
- Agents believe only results they can substantiate locally
- Agents communicate inconsistent assumption sets, termed "NOGOODS," which receiving agents use to disbelieve any results that have been obtained from the sending agent and that are justified by one of these sets
- *[Mason and Johnson]*

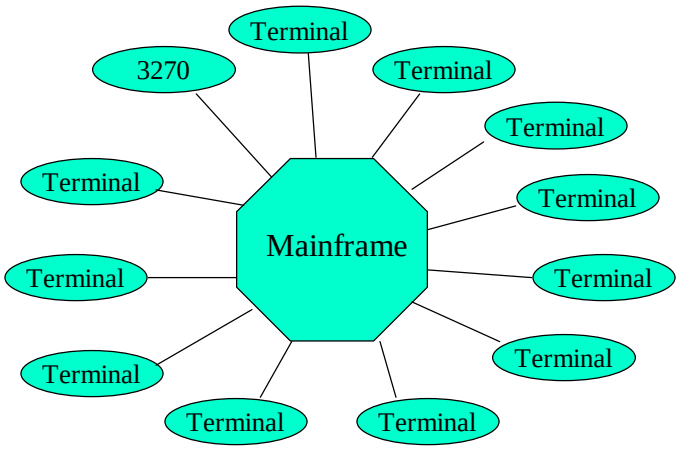
Cooperative Information Systems

Principles of Agent Systems

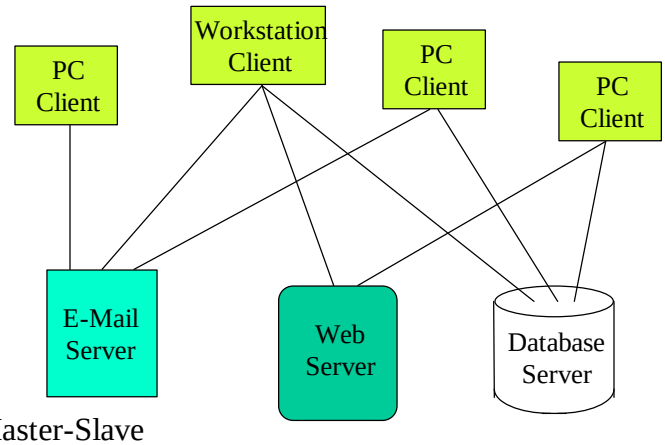
We must understand the principles to engineer agent-based systems

- System architecture
- Databases and information systems
- Interoperation
- Protocols and compliance
- Underlying frameworks and implementations

Information System Architectures: Centralized

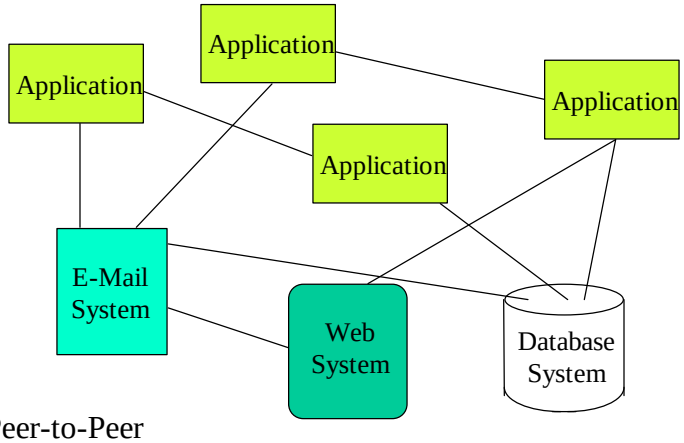


Information System Architectures: Client-Server



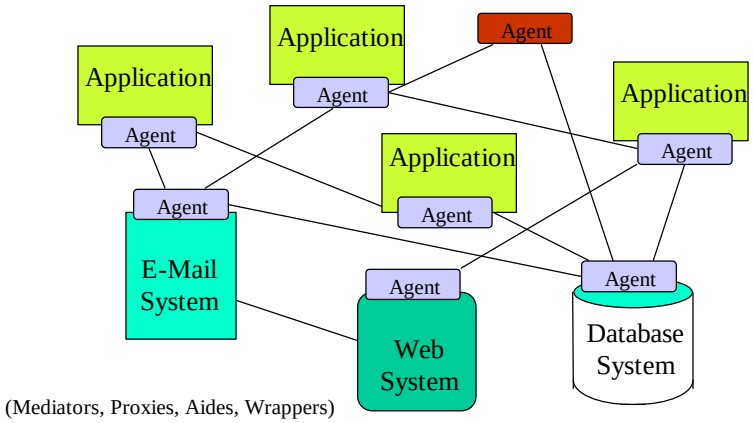
Master-Slave

Information System Architectures: Distributed



Peer-to-Peer

Information System Architectures: Cooperative



(Mediators, Proxies, Aides, Wrappers)

Cooperative Information Systems

MAS applied to decentralized information system components

- The study of how agents (as components) should coordinate their activities to achieve their goals:
  - cooperate when pursuing common or overlapping goals
  - compete intelligently when pursuing conflicting goals

When is CIS Applicable?

CIS is appropriate whenever the following hold:

- information is distributed, as in interacting businesses
- metadata is heterogeneous, as in schema integration
- data sources are distributed, as in energy management
- rewards are distributed, as in automated markets
- diverse interests must be represented, as in negotiation
- decisions are distributed, as in manufacturing control

Interoperation in Traditional and Virtual Enterprises

Cooperation in Information Systems

- *Connectivity*: ability to exchange messages
- *Interoperability*: ability to exchange messages to request and receive services, i.e., use each other's functionality
- *Cooperation*: ability to perform tasks jointly

Levels of Interoperation

Require surmounting a series of challenges

- Transport: IP
- Messaging: JDBC
- Data and structure: XML
- Process
- Policy
- Semantics: ontologies
- Dynamism: agents

Database Integration

Dimensions of Integration

- Existence of global schema
- Location transparency: same view of data and behavior at all sites
- Uniform access and update language
- Uniform interaction protocols
- Opacity of replication
- Strict semantic guarantees of overall system

Full Integration

Distributed databases are the most tightly integrated

- They provide
  - A global schema and a unique way to access data through that schema
  - Location transparency
  - Replication managed automatically
  - ACID transactions (explained below)

Federation

Less than full integration

- Local schemas and a global schema coexist: access may be through either
- ACID transactions are optional, but possible if the local transaction managers are open—problems of hidden conflicts must be solved
- Location transparency

Multidatabases

Multidatabases are a loose form of federation

- No global schema
- A uniform language to access the DB
- The locations of the data are visible to the application
- There may be some support for semantic constraints, depending on the underlying systems

Database Interoperation

Interoperation is the loosest form of integration, in that there is no real integration of the databases

- There might be no global schema, so heterogeneous ways to access the DB must coexist
- Location transparency is not easy to achieve
- Different query languages might be required at different databases (SQL, OQL)
- Applications must handle all semantic constraints

Legacy Systems

A pejorative term for computing systems that

- run on obsolete hardware and nonstandard communication networks
- run poorly documented, unmaintainable software created by ad hoc patches to handle bugs, and changing regulations and business needs
- consist of poorly modeled databases, often on hierarchical or network DBMSs
- support rigid user interfaces

Legacy Systems: Positive

- Fulfill crucial business functions
- Work, albeit suboptimally; Run
  - airline reservation systems
  - banks
  - business data processing
- Represent huge investments in time and money

Legacy Systems: Challenge

How to interoperate modern applications with legacy systems

- share data
- preserve integrity

Dimensions of Abstraction/1

Information resources are associated with abstractions over different dimensions. These may be thought of as constraints that must be discovered and represented.

- Data
  - domain specifications
  - value ranges, e.g., Price >+= 0
  - allow/disallow null values

Dimensions of Abstraction/2

- Structure
  - schemas and views, e.g., securities are stocks
  - specializations and generalizations of domain concepts, e.g., stocks are a kind of liquid asset
  - value maps, e.g., S&P A+ rating corresponds to Moody's A rating
  - semantic data properties, sufficient to characterize the value maps, e.g., prices on the Madrid Exchange are daily averages rather than closing prices
  - cardinality constraints
  - integrity constraints, e.g., each stock must have a unique SEC identifier

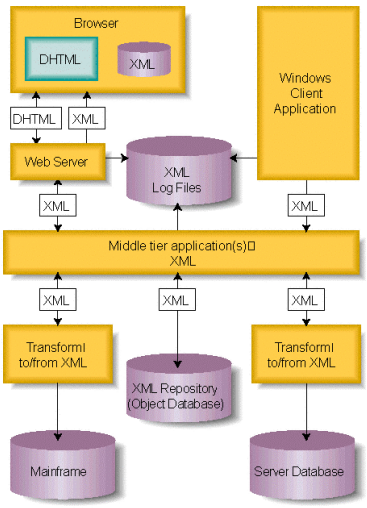
Dimensions of Abstraction/3

- Process
  - procedures, i.e., how to process information, e.g., how to decide what stock to recommend
  - preferences for accesses and updates in case of data replication (based on recency or accuracy of data)
  - preferences to capture view update semantics
  - contingency strategies, e.g., whether to ignore, redo, or compensate
  - contingency procedures, i.e., how to compensate transactions
  - flow, e.g., where to forward requests or results
  - temporal constraints, e.g., report tax data every quarter

Dimensions of Abstraction/4

- Policy
  - security, i.e., who has rights to access or update what information? (e.g., customers can access all of their accounts, except blind trusts)
  - authentication, i.e., a sufficient test to establish identity (e.g., passwords, retinal scans, or smart cards)
  - bookkeeping (e.g., logging all accesses)

XML-Based Information System

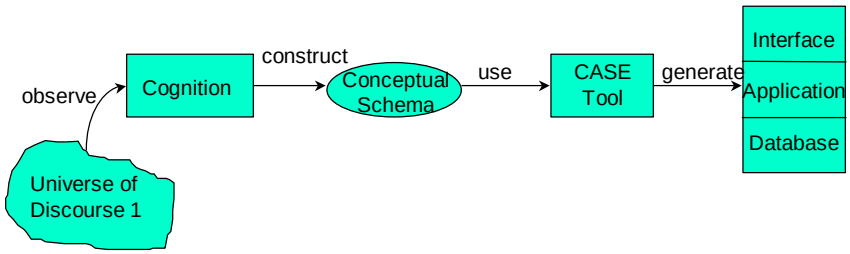


Enterprise Modeling

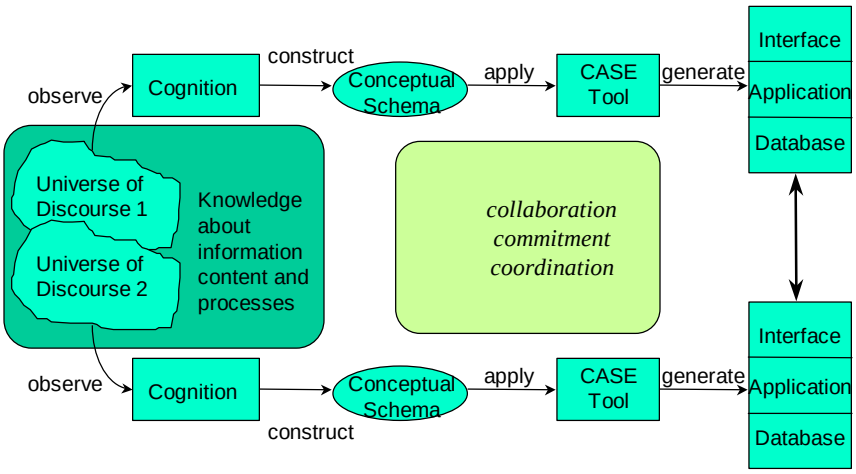
Model static and dynamic aspects of enterprises

- Models document business functions
  - databases
  - applications
  - knowledge bases
  - workflows, and the information they create, maintain, and use
  - the organization itself
- Models enable
  - reusability
  - integrity validation
  - consistency analysis
  - change impact analysis
  - automatic database and application generation

Building a System



Building Cooperating Systems



Process Abstractions

Task coordination covers the process dimensions of abstraction.

- Tasks execute on multiple client, server, and middleware systems
- Coordinate them
  - *for* distributed queries and transactions, and general workflow
  - *by* constraining control and data among them

Policy Abstractions

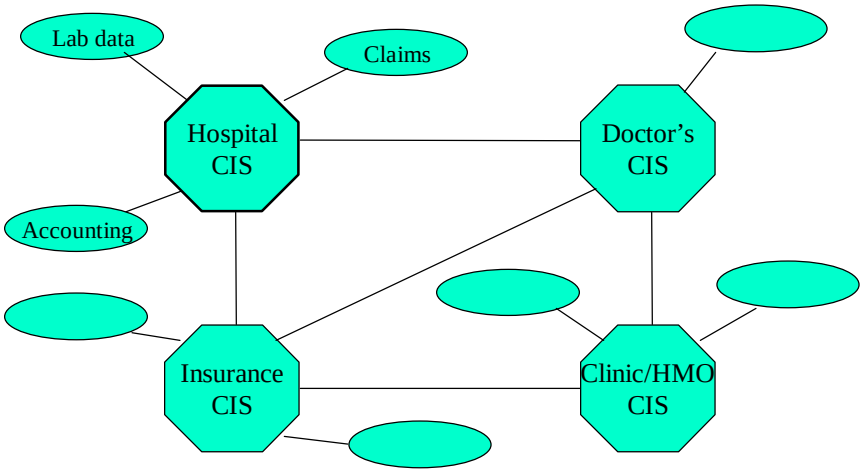
How may security, authentication, and bookkeeping policies be enforced?

- Alternatives:
  - Each resource applies them locally
  - Service level agreements capture the policies

Semantic Interoperability

- The goal for application development is to develop new applications that
  - extend over multiple new and legacy systems
  - respect the semantics of the various resources
- The approach is to
  - infer process business rules and integrity constraints
  - generate structure and process bindings in order to develop the given application

Information System Architectures: Organizational

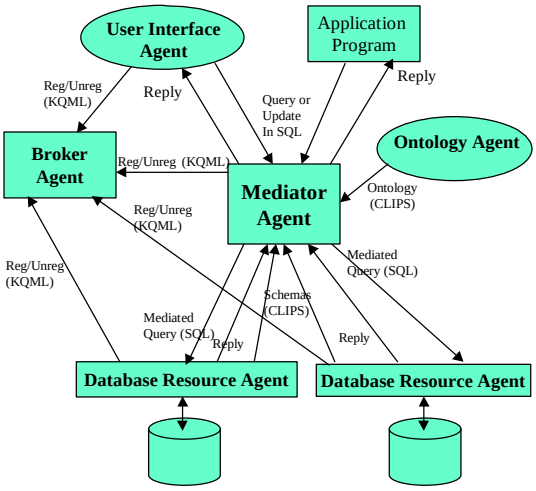


Standard MAS Architecture

Agent Environments

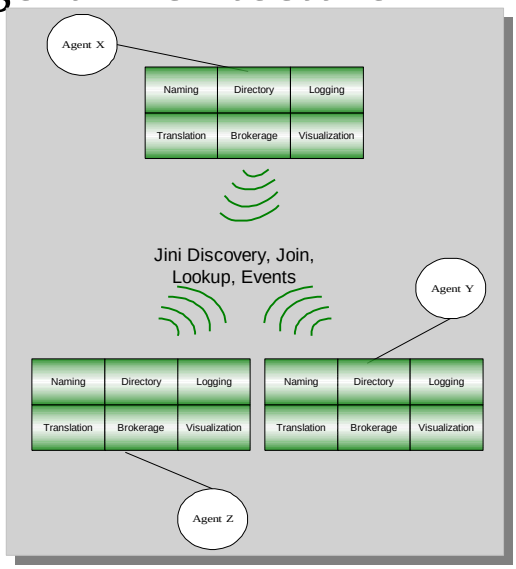
- Communication Infrastructure
  - Shared memory (blackboard)
  - Connected or Connectionless (email)
  - Point-to-Point, Multicast, or Broadcast
  - Directory Service
- Communication Protocol
  - KQML
  - HTTP and HTML
  - OLE, CORBA, DCOM, etc.
- Interaction Protocol
- Mediation Services
- Security Services (timestamps/authentication/currency)
- Remittance Services
- Operations Support (archiving/billing/redundancy/restoration/accounting)

(de facto) Standard Agent Types

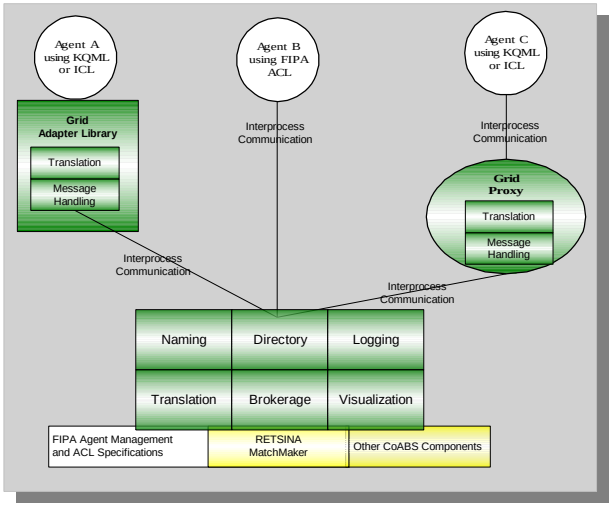


Intranet Agent Architecture

- Intranet services enable the federation of distributed heterogeneous agents to interoperate and collaborate
- Agents register their names and capabilities using
  - naming service
  - directory service
- Agents use a brokerage service to find other agents that can deal with a specified request
- Agent activity within the net is recorded by a logging service
- Agent status and interactions are shown by a visualization service
- Interactions via an ACL

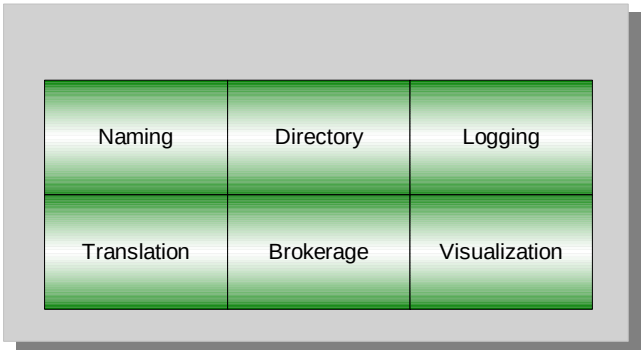


Access to Services



Network Services

- 1 instance of the Naming Service per LAN
- 1..n instances of Directory and Brokerage Service per LAN



Naming Service

- Architecture requires scalable, symbolic name resolution
- Alternative naming protocols
  - FIPA
  - LDAP
  - Jini
  - CORBA Naming Service
  - JNDI

Directory Service

- Simple yellow pages service
- Registered agents advertise their services by providing their name, address, and service description
- Agents request recommendations for available services (provided by other registered agents or services)
- A simple database-like mechanism that allows agents to
  - insert descriptions of the services they offer
  - query for services offered by other agents.
- 1..n Directory Service Agents on a LAN
- Brokerage, recruitment and mediation services are not provided by Directory Service

Brokerage Service

- Cooperates with the Directory Service
- An agent requests the Brokerage Service to recruit one or more agents who can answer a query
- Brokerage Service uses knowledge about the requirements and capabilities of registered agents to
  - determine the appropriate agents to which to forward a query
  - send the query to those agents
  - relay their answers back to the original requestor
  - learn about the properties of the responses it passes on
    - example: Brokerage agent determines that advertised results from agent X are incomplete and seeks out a substitute for agent X

Agents for Messaging and Structure Interoperation

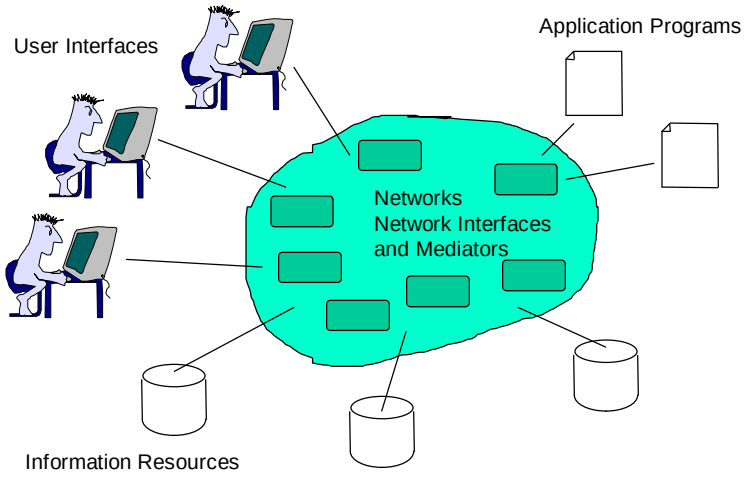
- Mediators [Wiederhold]
- Aides [Carnot DCA]
- Database and Protocol Agents [Carnot ESS]
- Heads [Steiner]
- Brokers [OMNI]
- Knowledge handlers [COSMO]
- Intelligent information agents [Papazoglou]
- Facilitators [ARPA Knowledge Sharing Effort]

Mediators

Modules that exploit encoded knowledge about data to create information for higher-level applications. Mediators, thus,

- provide logical views of the underlying information
- reside in an active layer between applications and resources
- are small, simple, and maintainable independently of others
- are declaratively specified, where possible, and inspectable by users
- come in a wide range of capabilities, from database and protocol converters, to intelligent modules that capture the semantics of the domain and learn from the data

Mediator Architecture



Mediator Interfaces

- Mediators should be separate from databases
  - mediators contain knowledge beyond the scope of a database
  - mediators contain abstractions that are not part of a database
  - mediators must deal with uncertainty
  - mediators access multiple databases to combine disjoint data
- Mediators should be separate from applications
  - their functions are different in scope than those of applications
  - separate mediators are easier to maintain
- Because mediators are stable and small, they can be mobile
  - they can be shipped to sites where large volumes of data must be processed

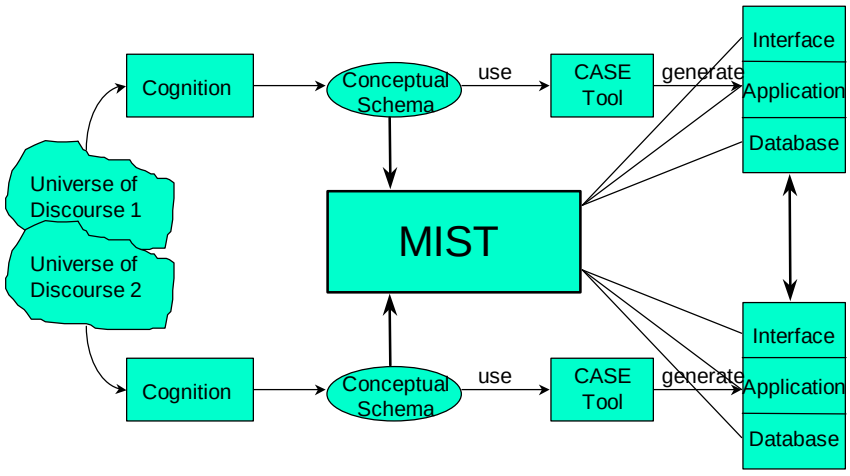
Type Brokers

A low-level means to manage structure and semantics of information and query languages. Brokers

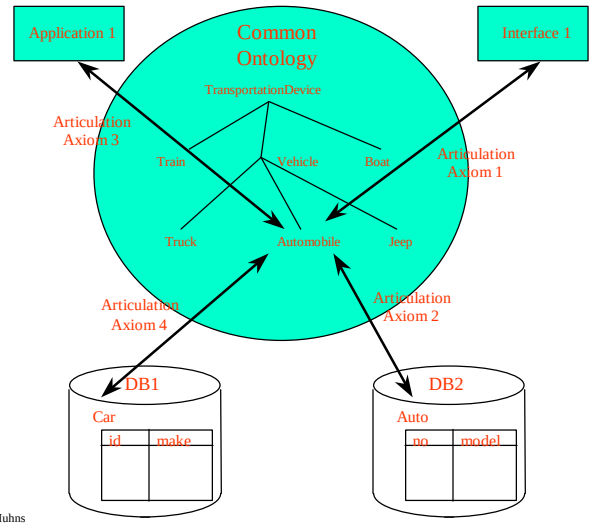
- define standard types by which computations can communicate
- distribute the type information—an application uses the broker to find a service, and then communicates directly with the service
- give slightly more semantics than directories—the type signature of methods, not just their names

With more sophisticated notions of service semantics, brokers could be more useful

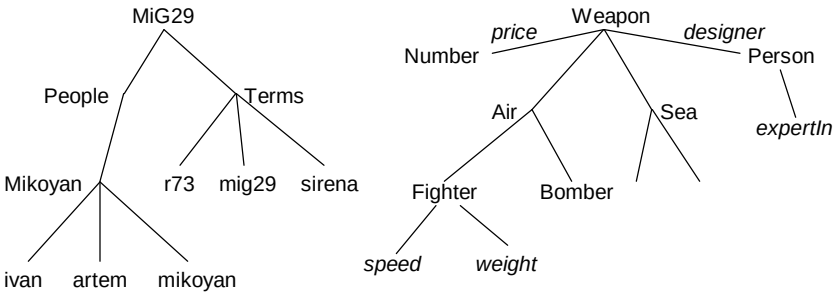
Creating Semantic Mappings



Ontologies and Articulation Axioms



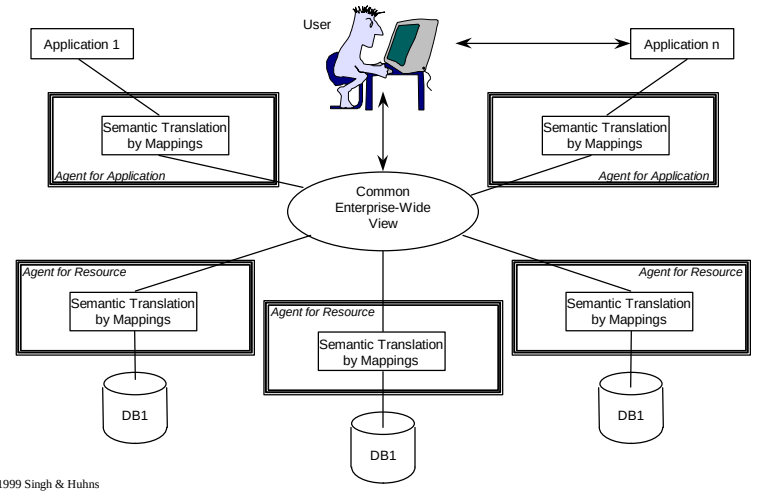
Topic Trees, Ontologies, and Database Schemas



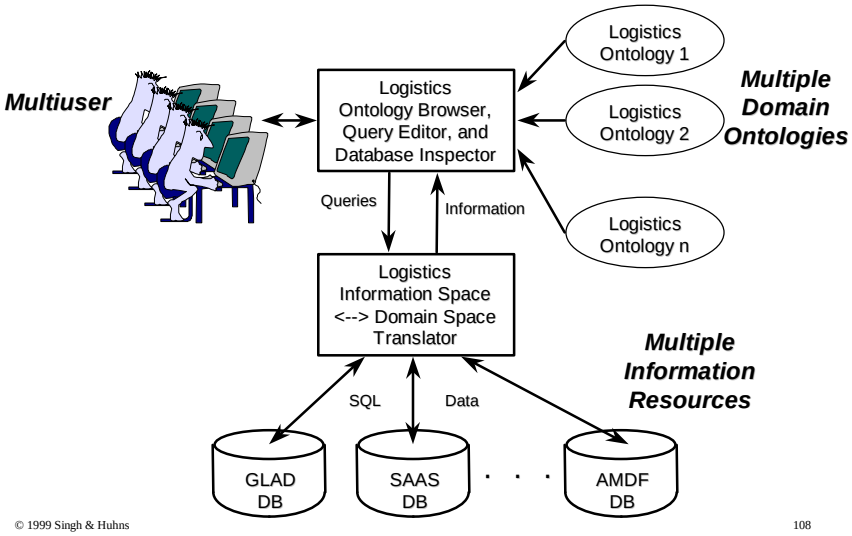
Person	DOB	Specialty

Fighter	Speed	Weight	Price

Semantic Translation



Logistics Information Management



Database Abstractions for Computation

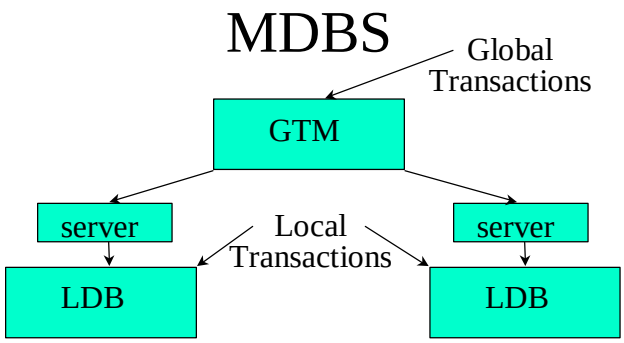
Traditional Transactions

DB abstraction for activity

- ACID properties
  - *Atomicity*: all or none
  - *Consistency*: final state is consistent if initial state is consistent
  - *Isolation*: intermediate states are invisible
  - *Durability*: committed results are permanent

In distributed settings, use mutual (e.g., two-phase) commit to prevent violation of ACID properties

x:=x-a y:=y+a



3 levels of autonomy are possible

- *design*, e.g., LDB software is fixed
- *execution*, e.g., LDB retains full control on execution even if in conflict with GTM
- *communication*, e.g., LDB decides what (control) information to release

Global Serializability

Transactions throughout the MDBS are serializable, i.e., the transactions are equivalent to some serial execution

- What the GTM can ensure is that the global transactions are serializable
- This doesn't guarantee global serializability, because of *indirect* conflicts:
  - GTM does T1: r1(a); r1(c)
  - GTM does T2: r2(b); r2(d)
  - LDB1 does T3: w3(a); w3(b)
  - LDB2 does T4: w4(c); w4(d)
  - Since T1 and T2 are read-only, they are serializable.
  - LDB1 sees S1=r1(a); c1; w3(a); w3(b); c3; r2(b); c2
  - LDB2 sees S2=w4(c); r1(c); c1; r2(d); c2; w4(d); c4
  - Each LDB has a serializable schedule; yet jointly they put T1 before and after T2

Global Atomicity

This arises because some sites may not release their prepare-to-commit state and not participate in a global commit protocol

Global Deadlock

Easy to construct scenarios in which a deadlock is achieved. Assume LDB1 and LDB2 use 2PL. If a deadlock is formed

- solely of global transactions, then the GTM may detect it
- of a combination of local and global transactions, then
  - GTM won't know of it
  - LDBs won't share control information

Tickets

Global serializability occurs because of local conflicts that the GTM doesn't see

- Fix by always causing conflicts--whenever two GTs execute at a site, they must conflict there. Indirect conflicts become local conflicts visible to the LDB
  - Make each GT increment a ticket at each site
- Downside:
  - Causes all local subtransactions of a global transaction to go through a local hotspot
  - GTs are serialized but only because lots are aborted!

Rigorous DBMS

Rigorous = Strict.

- Check that this prevents the bad example.
- The GTM must delay all commits until all actions are completed
  - possible only if allowed by LDB
  - requires an operation-level interface to LDB
- Downside:
  - Causes all sites to be held up until all are ready to commit
  - Essentially like the 2PC approach

Global Constraints

- When no global constraints, local serializability is enough
- Can split data into local and global
  - LDB controls local data
  - GTM controls global (local read but only write via GTM)
- Downside: doesn't work in all cases

Atomicity & Durability

What happens when a GT fails?

- The local sites ensure atomicity and durability of the local subtransactions
- With 2PC, GTM can guarantee that all or none commit

Otherwise,

- redo: rerun the writes from log
- retry: rerun all of a subtransactions
- compensate: semantically undo all others

B2B Workflows

Workflows

- Tasks include queries, transactions, applications, and administrative activities
- Tasks decompose into subtasks that are
  - distributed and heterogeneous, but
  - coordinated
- Subtasks have mutual constraints on
  - order
  - occurrence
  - return values

Workflow Applications

- Loan application processing
- Processing admissions to graduate program
- Telecommunications service provisioning often requires
  - several weeks
  - many operations (48 in all, 23 manual)
  - coordination among many operation-support systems and network elements (16 database systems)

Why Workflows?

- ACID transactions are applicable for
  - brief, simple activities (few updates; seconds, at most)
  - on centralized architectures
- By contrast, open environments require tasks that are
  - Complex, i.e., long-running, failure-prone, update data across systems with subtle consistency requirements
  - Cooperative, i.e., involve several applications and humans
  - Over heterogeneous environments
  - Have autonomous unchangeable parts

Workflow Challenges

- Modeling a workflow
  - Notion of correctness of executions
  - Notion of resource constraints
- Interfacing a workflow interface with underlying databases?
  - Concurrency control
  - Recovery
- Exception handling: normal executions are often easy—just a partial order of activities
- Handling revisions

Extended Transactions

Numerous extended transaction models that relax the ACID properties in set ways. They consider features such as

- Nesting
  - traditional: closed (ACID)
  - newer: open (non-ACID)
- Constraints among subtransactions, such as
  - commit dependencies
  - abort
- Atomicity, e.g., contingency procedures to ensure “all”
- Consistency restoration, e.g, compensation

Extended Transaction Models

- Sagas
- Poly transactions
- Flex transactions
- Cooperative transactions
- DOM transactions
- Split-and-join transactions
- ACTA metamodel
- Long-running activities
- ConTracts

Scheduling Approaches

Ensure how activities may be scheduled, assuming that desired semantic properties are known

- *Significant events* of a task are the events that are relevant for coordination. Thus a complex activity may be reduced to a single state and termination of that activity to a significant event
- Workflows can be modeled in terms of *dependencies* among the significant events of their subtasks

Example: If the booking assignment transaction fails, then initiate a compensate transaction for the billing transaction

B2B Process Interoperation

Organizational Treatment of Interoperation

Spheres of Commitment

- Define abstract societies. Each role
  - requires capabilities
  - imposes commitments
  - grants authorities
- Agents instantiate spheres of commitment
  - autonomously or when configured by humans
  - behave according to the commitments

Social Abstractions

- Commitments: social, joint, collective, ...
- Organizations and roles
- Teams and teamwork
- Mutual beliefs and problems
- Joint intentions
- Potential conflict with individual rationality

Coherence and Commitments

- Coherence is how well a system behaves as a unit. It requires some form of organization, typically hierarchical
- Commitments among the agents are a means to achieve coherence

Kinds of Commitment

- Psychological or mental: an agent's state of being committed to a belief or an intention
- Joint: agents' commitments to the same intention or belief
- Mutual: agents' commitments to one another with respect to the same condition

Social Commitments

An agent's commitment to another agent

- unidirectional
- arising within a well-defined scope or *context*, which is itself a MAS
- revocable within limits

Organizations

Whenever agents work together in a shared environment and with some structure to their interactions. Typically,

- larger-scale than single agent
- engaged in tasks
- goal-oriented
- with knowledge and memory beyond individual

Motivation for Organizations

Organizations help overcome the limitations of agents in various respects (recall definition of an agent)

- reasoning
- capabilities
- perception
- lifetime and persistence
- shared context essential for communicating

Modeling Organizations

- Abstractly, organizations
  - consist of roles
    - requiring certain capabilities
    - offering certain authorities
  - involve commitments among the roles
- Concretely, organizations
  - consist of agents
  - acting coherently

Sphere of Commitment

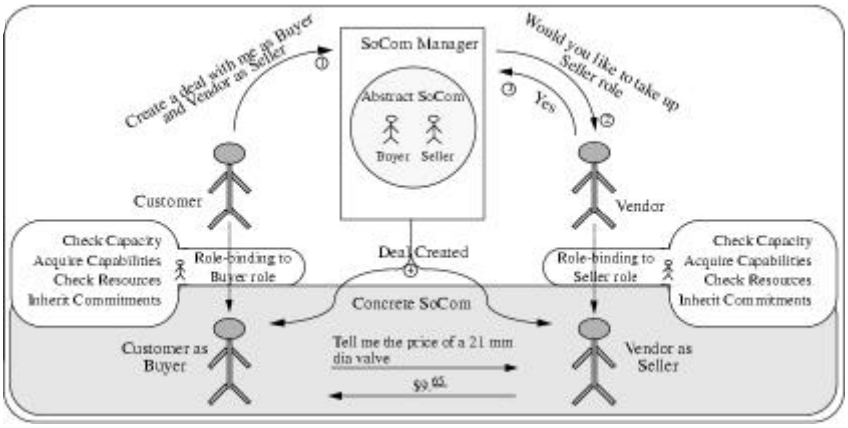
SoCom: an organization that provides the context or scope of commitments among relevant agents

- the SoCom serves as a witness for the commitment
- helps validate commitments and test for compliance
- offers compensations to undo members' actions

Example: Buying and Selling

- Define an abstract sphere of commitment (SoCom) consisting of two roles: buyer and seller, which require capabilities and commitments about, e.g.,
  - requests they will honor
  - validity of price quotes
- To adopt these roles, agents must have the capabilities and acquire the commitments.

Buyer and Seller Agents



SoComs provide the context for the concepts represented & communicated.

Example: Buying and Selling

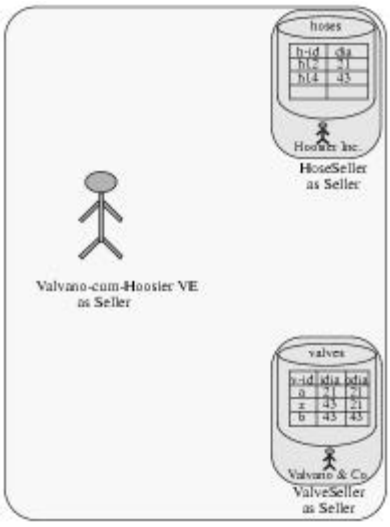
- Agents can join
  - during execution—requires publishing the definition of the commerce SoCom
  - when configured by humans
- The agents then behave according to the commitments
- Toolkit should help define and execute commitments, and detect conflicts.

Virtual Enterprises (VE)

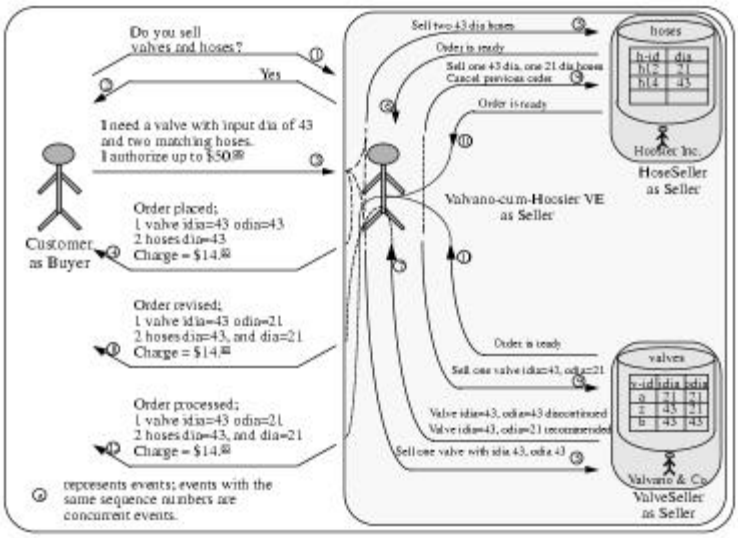
Two sellers come together with a new proxy agent called VE.

Example of VE agent commitments:

- notify on change
- update orders
- guarantee the price
- guarantee delivery date



A Selling VE



Social Commitments

- Operations on commitments (instantiated as social actions):
  - create
  - discharge (satisfy)
  - cancel
  - release (eliminate)
  - delegate (change debtor)
  - assign (change creditor).

Policies and Structure

- Spheres of commitment (SoComs)
  - abstract specifications of societies
  - made concrete prior to execution
- Policies apply on performing social actions
- Policies related to the nesting of SoComs
- Role conflicts can occur when agents play multiple roles, e.g., because of nonunique nesting.

Teams

Tightly knit organizations

- shared goals, i.e., goals that all team members have
- commitments to help team-members
- commitments to adopt additional roles and offer capabilities on behalf of a disabled member

Teamwork

When a team carries out some complex activity

- negotiating what to do
- monitoring actions jointly
- supporting each other
- repairing plans

Joint Intentions

Traditional accounts of teams are based on joint intentions and mutual beliefs

- Team-members jointly intend the main goal of the team, which means that they
  - all intend it and mutually believe that they intend it
  - each will notify the others if it drops out and mutually believe this notification requirement

Problems with Joint Intentions: 1

- Joint intentions appear useful, although somewhat complicated. More importantly,
  - Several agents mutually believe a proposition p if and only if each believes p, each believes that each believes p, and so on ad infinitum
  - Thus mutual beliefs cannot be attained (unless hardwired in) through asynchronous communication with unreliable or unbounded messages

Problems with Joint Intentions: 2

  - The same construction can be made for joint intentions
  - In other words, such idealized teams cannot exist

The intuitive problem is that this approach tries to simulate social structure purely through mental concepts.

Multiagent Workflow at NCSU

Workflows are composite, long-running activities

- exceptions
- revisions
- flexible negotiation

The NCSU approach involves

- roles with patterns of commitment as statecharts
- reflective use of capabilities

JESS + JATLite prototype implementation

Service Location and Evaluation

Retrieval versus Discovery

What or who?

- Information retrieval is concerned with
  - obtaining information
  - from a specific set of servers
  - with a specific query
  - correctness matters
- Resource discovery and location is concerned with
  - finding where to get the information
  - incompleteness might matter
  - relevance matters

Retrieval is almost as difficult in closed environments as in open ones. Discovery is not as difficult a problem in closed environments

Network Navigation

Organize servers into a network so that you can reach appropriate ones from each other (e.g., WWW)

Typically a matter of following pointers through a friendly, but not very helpful, interface:

- No semantics or notion of relevance
- No support for query routing
- No support for making links

Resource Discovery Challenges

- Scalability (w.r.t. network and server loads)
  - number of services
  - complexity of service description
  - number of users
- Efficient and effective indexing of information sources
  - high relevancy
  - low volume, i.e., high precision

Harvest

Main architectural components:

- *Provider*, which runs a service (resource)
- *Gatherer*, which resides on the provider's site and monitors it for indexing info; specialized for a topic
- *Broker*, which gives an indexed query interface to a number of providers
- *Replicator*, which manages a wide-area file-system (with eventual consistency)
- *Service Registry*, which knows about all gatherers and brokers

Brokers can be nested

Users can get to succinct indexes from which they can find the server sites with the best fit for their query

No semantic support; topic-specific indexing can be demanding, but is still a good solution

Content Routing

Associates a content label with each document or collection (may be recursive). The label abstractly specifies the contents of its document or collection

- Queries (and labels) are boolean combinations of attributes
- User begins a query at a server; a standard server exists for novices
- The query is matched against labels to select good collections; this is done repeatedly until a base collection (one with documents) is found

No semantics to the labels

Applying Taxonomies

A network of specialized information agents (in SIMS)

- Knowledge and queries are represented in a taxonomic (concept) language (LOOM)
  - selection of information sources
  - reformulation of query and generation of query plans
- The representation support heuristics to
  - generalize or specialize concepts
  - partition concepts
  - decompose relations

Recommender Systems Applied

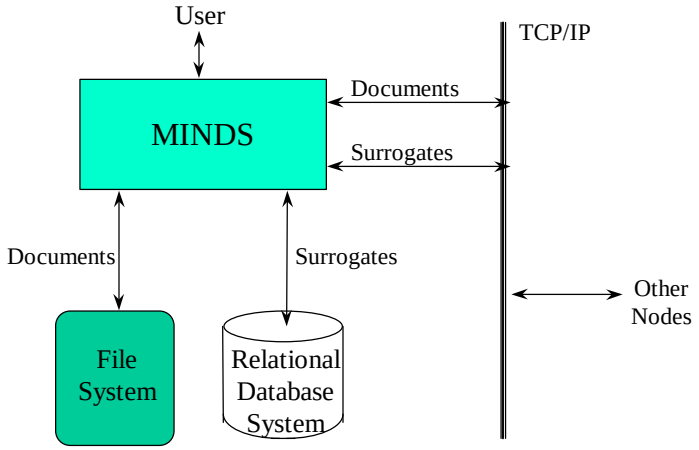
Explosive growth: crucial for

- social interaction
- e-communities
- e-commerce: potentially a way to
  - increase security by presenting a group's views
  - make it harder for a scam artist to repeatedly exploit trusting people

Recommender Techniques

- *Collaborative-filtering*: find things liked by people similar to you (e.g., Lira, Firefly)
- *Matchmaker systems*: cluster the agents with similar interests (e.g., Yenta)
- *Expertise location*: find someone with the needed expertise (e.g., ReferralWeb)

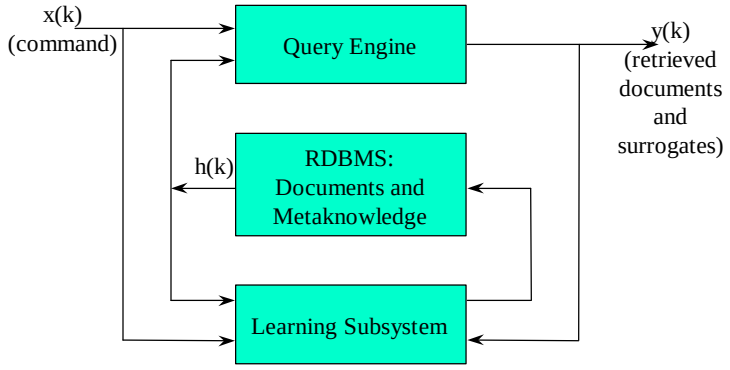
MINDS: Agent-Based Management of Flat-File Documents



Key Features of MINDS

- Uses
  - metaknowledge (about other agents and how information is used)
  - knowledge about the contents and locations of documents-these are acquired dynamically
  - certainty factors that indicate the likelihood that one user will supply relevant information to another to control query processing
- Maintains
  - a system view at each node
  - user models at each node
- Is self-initializing
- Processes queries best-first and in parallel
- Improves its performance over time

System Dynamics of MINDS



Output function: y(k) = q(x(k), h(k))
State function: h(k+1) = a(x(k), h(k))

Updating Metaknowledge in MINDS

```
Heuristic 1: IF a document is deleted,
  THEN no metaknowledge is changed
Heuristic 2: IF a document is created by user1,
  THEN metaknowledge of user1 about user1
  regarding each keyword of the document is
  increased to 1.0 (maximum relevance)
Heuristic 3: IF a retrieve predicated on keyword1
  is issued by user1,
  AND at least one user2 surrogate contains
  keyword1,
  THEN (a) user1 metaknowledge about user2
  regarding keyword1 is increased (weight 0.1),
  (b) user2 metaknowledge about user1 regarding
  keyword1 is increased (weight 0.1)
```

Agent Amplified Expertise Location

Each person is assigned a user agent, which keeps the user's expertise/interest profile:

- scans (all) private email and files
- indexes keywords and phrases
- creates a list of email contacts, indexed by context
- matches requests against the profile using information retrieval techniques

Agent Amplified Process

- Hand query to user agent
- Agent computes set of relevant contacts
- Agent sends query to contacts' agents
- If there is a good match, receiving agent passes the request on to its owner
- Otherwise, receiving agent sends back referrals (if any)

Advantages

- Largely passive on part of humans
  - users are shielded from irrelevant messages
  - messages are tagged by referral chain
- Messages are focused, not broadcast
  - can access selected members of large community

Information Access

Direct connection to online data: DBs, WWW

- Slow, frustrating
- Where to look?
- Much valuable information *deliberately* not online, because of two reasons:
  - *Economic*: Value of info is partly determined by how hard it is to find
  - *Social*: People may be reluctant to state sensitive information publicly

Information Access

Person to person communication: email, newsgroups

- Who to ask?
  - *individuals*: spamming not welcomed
  - *mailing lists*: popularity decrease effectiveness
  - *newsgroups*: no idea of who is listening

Referral Systems

Why Referral Systems?

- Most recommender systems hide the identity of the sources of the recommendations
  - E-communities: fictitious identities
  - Matchmaker systems: deliberately hide true identities
  - Collaborative filtering: aggregate results—no one to trust (or blame)

Why Referral Systems?

Result: anonymous opinions

- People can more easily trust opinions of those whom they personally know
  - Useful for simple decisions, such as buying a CD or watching a movie
  - Essential for serious decisions

Trusted Recommendations

- For serious decisions, you want the opinion of a trusted expert
- If an expert is not personally known, then you want a reference to an expert via a chain of friends and colleagues

Referrals

- Referral chains provide:
  - Way to judge quality of expert's advice
  - Reason for the expert to respond in a trustworthy manner
- Finding good referral-chains is
  - slow and time-consuming
  - but vital

Social Network

Set of all possible referral chains in which an individual may participate

- As the chains get longer,
  - the trustability of a recommendation decreases
  - the effort to find experts increases
- Therefore, shorter chains are better

Small World Phenomenon

Milgram (1967): almost any two individuals in the U.S.A. are linked by a chain of 6 or fewer first-name acquaintances (empirical observation)

- Six degrees of separation
- Erdös numbers

Social Networks via Referrals

Typically people are directly aware of only a small part of their social network

- Referrals can help discover the wider social network or even expand it
- Referrals help create a larger community, exposing connections to more people who would otherwise be hidden over the horizon.

MARS: A Multiagent Referral System

A system for creating and managing social networks in

- a company
- an e-community
- the WWW

- Simulation results described; usable prototype under construction

MARS Features

- Integrates IR techniques with adaptive user modeling to refine the social network
- Allows allows agents to unintrusively exchange explicit profile information to add coverage regarding a user's expertise.

Experimental Setup

- Each agent has
  - some interests
  - some expertise
  - a set of neighbors, each with interests and expertise, which are modeled by the given agent

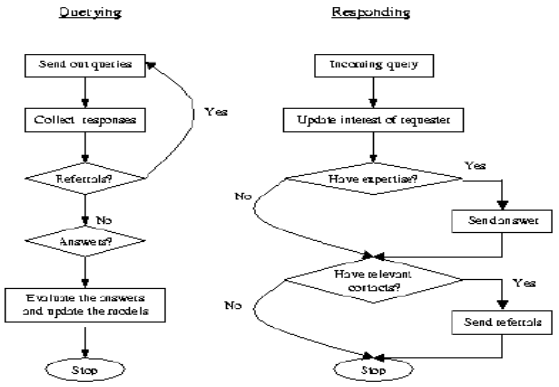
Queries

- Queries are generated by users and forwarded by their agents
- In the simulation, an agent's queries are generated based on his interest

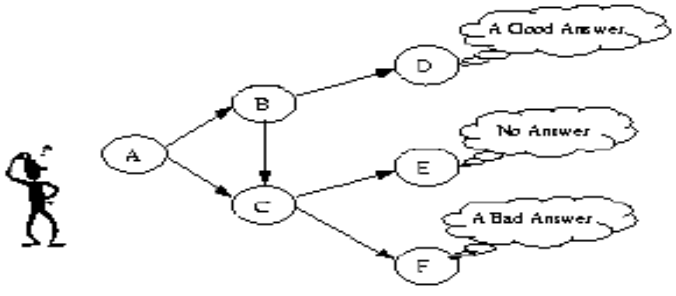
Responses

- Responses are of two kinds
  - answers based on the responder's expertise
  - referrals to other agents known to the responder
- Responses are produced by either
  - a user
  - an agent, who tries to handle as many as queries as possible to minimize interruptions for the user.

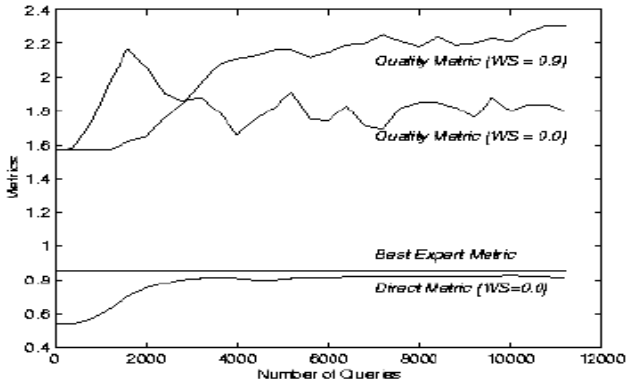
Querying and Responding



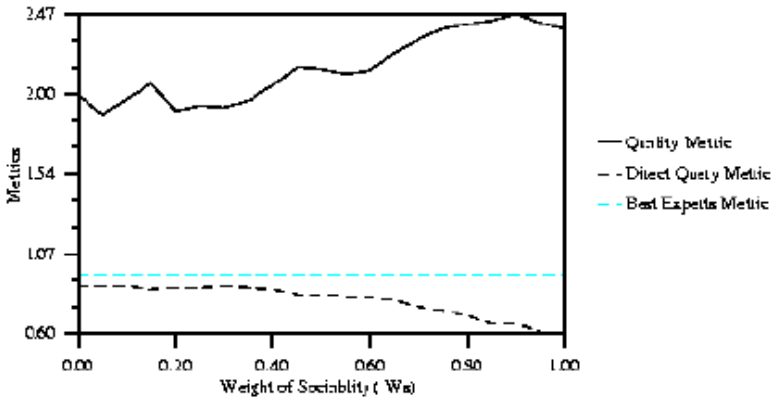
Example Referral Graph



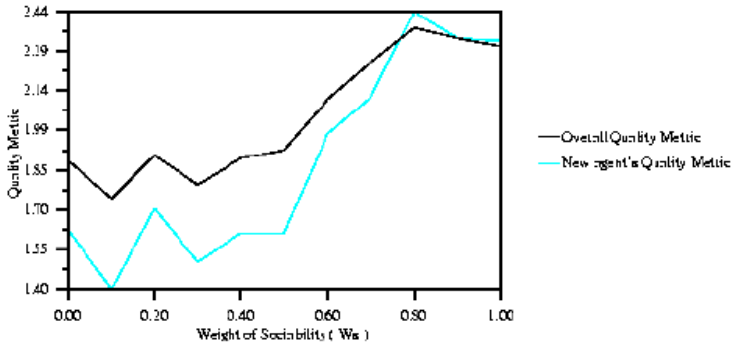
Quality after Increasing Interactions



Referrals at Different Sociability



Improvement of Quality for a New Member



Small-World Network

A highly structured (clustered) network with just a few random edges

- Yields high clustering and short paths
- Random edges correspond to shortcuts
  - direct relationships between people who primarily participate in different sub-communities
  - shortcuts: weak-ties

Weak-ties in a Social Network

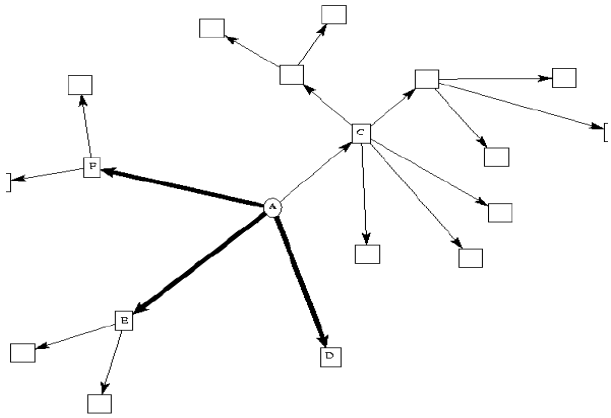
Acquaintances who form shortcuts to other communities

- offer direct relationships between people who primarily participate in different sub-communities
- improve the effectiveness of referrals by bringing in knowledge of other communities.

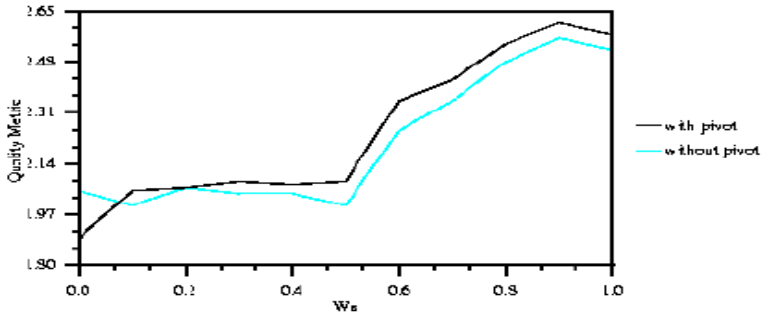
Pivots and Weak-ties

- Pivots are people with high connectivity who are important to the network
- The best weak ties are to pivots, who then link one up with lots of other communities.

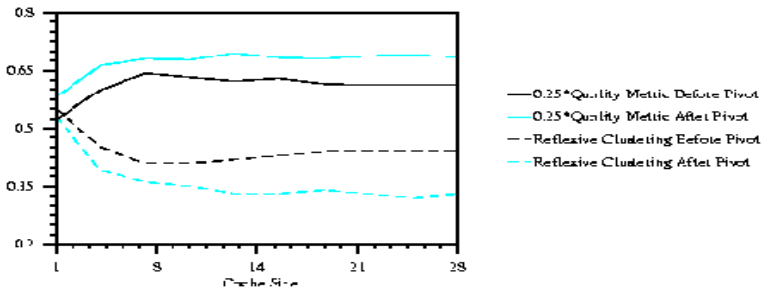
Pivots in Social Networks



Improvement Due to a Pivot



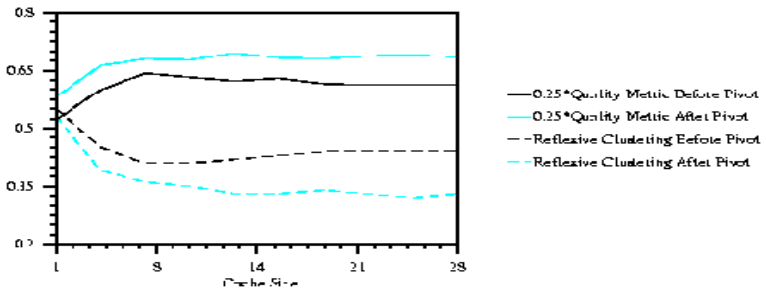
Catalysis at Different Caches



Weak-ties versus Clustering

- Conventional approaches give recommendations based on the preferences of similar users (form clusters)
- For finding referrals, it is best to ask dissimilar people who bring a novel perspective
  - define a form of controlled scattering

Catalysis at Different Caches



Future Work

Usable prototype (in progress)

- Scale-up to handle more users
- Learning
  - user profiles and helpfulness
  - user needs
- Trust and Reputation
  - trust and authority
  - reliability

General Agent and MAS Technologies

Control

Importance of Control

How to maintain global coherence without explicit global control? Important aspects include how to

- determine shared goals
- determine common tasks
- avoid unnecessary conflicts
- pool knowledge and evidence

Task Decomposition

Divide-and-conquer to reduce complexity: smaller subtasks require less capable agents and fewer resources

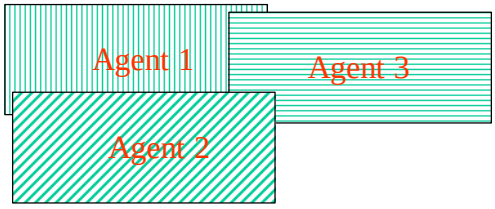
- Task decomposition must consider the resources and capabilities of the agents, and potential conflicts among tasks and agents
  - The designer decides what operators to decompose over
  - The system decides among alternative decompositions, if available

Task Decomposition Methods

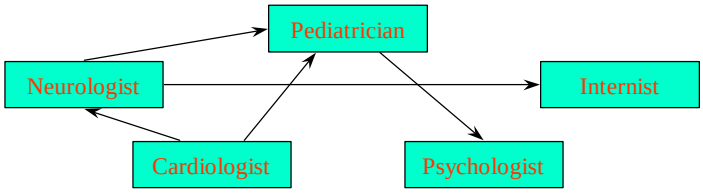
- Inherent (free!): the representation of the problem contains its decomposition, as in an AND-OR graph
- System designer (human does it): decomposition is programmed during implementation. (There are few principles for automatically decomposing tasks)
- Hierarchical planning (agents do it): decomposition again depends heavily on task and operator representation

Task Decomposition Examples

- Spatial decomposition by information source or decision point:



- Functional decomposition by expertise:



Task Distribution Criteria

- Avoid overloading critical resources
- Assign tasks to agents with matching capabilities
- Make an agent with a wide view assign tasks to other agents
- Assign overlapping responsibilities to agents to achieve coherence
- Assign highly interdependent tasks to agents in spatial or semantic proximity. This minimizes communication and synchronization costs
- Reassign tasks if necessary for completing urgent tasks

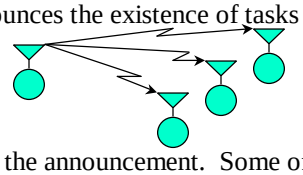
Task Distribution Mechanisms

- Market mechanisms: tasks are matched to agents by generalized agreement or mutual selection (analogous to pricing commodities)
- Contract net: announce, bid, and award cycles
- Multiagent planning: planning agents have the responsibility for task assignment
- Organizational structure: agents have fixed responsibilities for particular tasks
- Recursive allocation: responsible agent may further decompose task and allocate the resultant subtasks

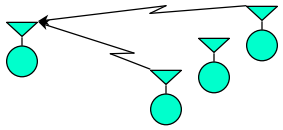
The Contract Net Protocol

An important generic protocol

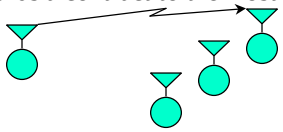
- A manager announces the existence of tasks via a (possibly selective) multicast



- Agents evaluate the announcement. Some of these agents submit bids



- The manager awards a contract to the most appropriate agent



- The manager and contractor communicate privately as necessary

Task Announcement Message

- Eligibility specification: criteria that a node must meet to be eligible to submit a bid
- Task abstraction: a brief description of the task to be executed
- Bid specification: a description of the expected format of the bid
- Expiration time: a statement of the time interval during which the task announcement is valid

Bid and Award Messages

- A bid consists of a *node abstraction*—a brief specification of the agent's capabilities that are relevant to the task
- An award consists of a *task specification*—the complete specification of the task

Applicability of Contract Net

The Contract Net is

- a high-level communication protocol
- a way of distributing tasks
- a means of self-organization for a group of agents

Best used when

- the application has a well-defined hierarchy of tasks
- the problem has a coarse-grained decomposition
- the subtasks minimally interact with each other, but cooperate when they do

Interaction and Communication

Communication

- The primary reason for communication among agents is to *coordinate* activities
- Agents may coordinate *without* communication provided they have models of the others' behavior
- Communication involves the dimensions of *who, what, when, how* (resources and protocol), *why*
- To cooperate, agents often need to communicate their *intentions, goals, results,* and *state*

Communication Versus Computation

- Communication is generally more expensive and less reliable:
  - Recomputing is often faster than requesting remote information
  - Communication can lead to prolonged reasoning and negotiation
- Communication is qualitatively superior:
  - Information cannot always be reconstructed locally
  - Communication can be avoided only when the agents are set up to
    - share all necessary knowledge
    - make observations directly from their shared environment

Interaction and Communication

- Interactions occur when agents exist and act in close proximity:
  - resource contention, e.g., bumping into each other
- Communications are the interactions that preserve autonomy of all participants, but maybe realized through physical actions that don't
- Communications can be realized in several ways, e.g.,
  - through shared memory
  - because of shared conventions
  - by messaging passing

MAS Communication Protocols

- A MAS protocol is specified by the following:
  - sender
  - receiver(s)
  - language in the protocol
  - actions to be taken by the participants at various stages
- A MAS protocol is defined above the transport layer
  - not about bit patterns
  - not about retransmissions or routing
- A MAS protocol is defined at the knowledge level
  - involves high-level concepts, such as
    - commitments, beliefs, intentions
    - permissions, requests

A Classification of Message Classifications

- Syntactic
  - distinguish messages based on grammatical forms in natural language
- Semantic
  - distinguish messages based on a notion of intrinsic meaning

    prohibitive is different from *directive*, despite syntactic similarity
- Use-based
  - distinguish messages based on their roles in specific classes of protocols

    assertion is different from *acknowledgment*

Speech Act Theory

- Speech act theory, developed for natural language, views communication as action.
- It considers three aspects of a message:
  - *Locution*, or how it is phrased, e.g., "It is hot here" or "Turn on the cooler"
  - *Illocution*, or how it is meant by the sender or understood by the receiver, e.g., a request to turn on the cooler or an assertion about the temperature
  - *Perlocution*, or how it influences the recipient, e.g., turns on the cooler, opens the window, ignores the speaker

Illocution is the core aspect.

Speech Act Theory and MAS

- Classifications of illocutions motivate message types in MAS, but are typically designed for natural language
  - rely on NL syntax, e.g., they conflate directives and prohibitives
- Most research in speech act theory is about determining the agents' beliefs and intentions, e.g., how locutions map to illocutions.
- In MAS,
  - determining the message type is trivial , because it is explicitly encoded
  - determining the agents' beliefs and intentions is impossible, because the internal details of the agents are not known.

Informing

How can one agent tell another agent something?

- Send the information in a message (message passing)
- Write the information in a location where the other agent is likely to look (shared memory)
- Show or demonstrate to the other agent (teaching)
- Insert or program the information directly into the other agent (master --> slave; controller --> controllee; "brain surgery")

Querying

How can one agent get information from another agent?

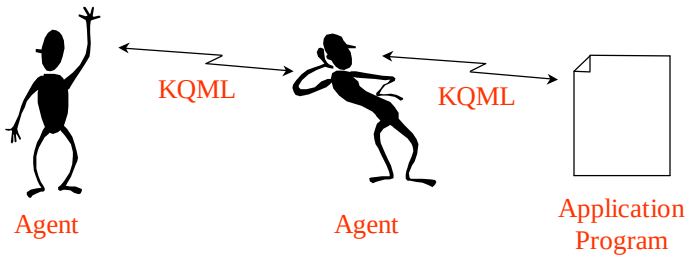
- Ask the other agent a question (message passing)
- Read a location where the other agent is likely to write something (shared memory)
- Observe the other agent (learning)
- Access the information directly from the other agent ("brain surgery")

Syntax, Semantics, Pragmatics

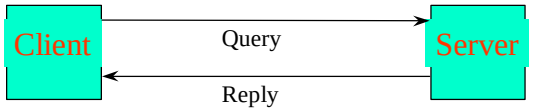
For message passing

- *Syntax*: requires a common language to represent information and queries, or languages that are intertranslatable
- *Semantics*: requires a structured vocabulary and a shared framework of knowledge-a shared ontology
- *Pragmatics*:
  - knowing whom to communicate with and how to find them
  - knowing how to initiate and maintain an exchange
  - knowing the effect of the communication on the recipient

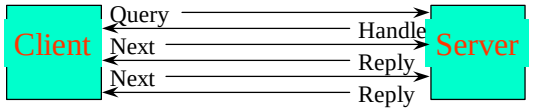
KQML: Knowledge Query and Manipulation Language



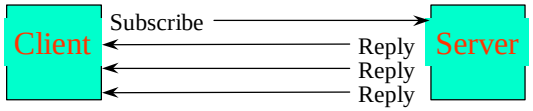
KQML Protocols



Synchronous: a blocking query waits for an expected reply

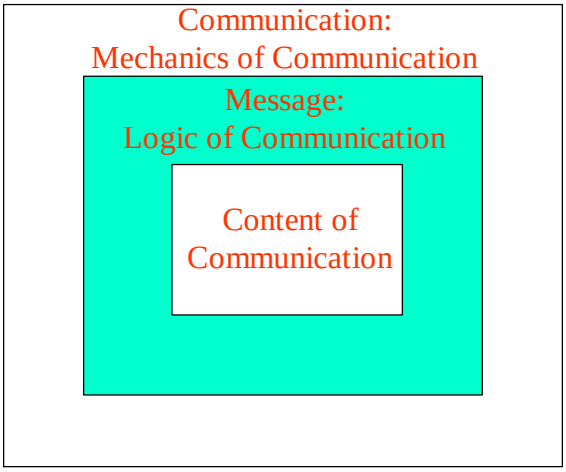


Server maintains state; replies sent individually when requested



Asynchronous: a nonblocking subscribe results in replies

KQML Is a Layered Language



Communication Assumptions

- Agents are connected by unidirectional links that carry discrete messages
- Links have nonzero transport delay
- Agent knows link of received message
- Agent controls link for sending
- Messages to a single destination arrive in the order they were sent
- Message delivery is reliable

KQML Semantics

- Each agent manages a virtual knowledge base (VKB)
- Statements in a VKB can be classified into *beliefs* and *goals*
- Beliefs encode information an agent has about itself and its environment
- Goals encode states of an agent's environment that it will act to achieve
- Agents use KQML to communicate about the contents of their own and others' VKBs

Reserved Performative Types

1. Query performatives:
   - evaluate, ask-if, ask-one, ask-all
2. Multiresponse performatives:
   - stream-in, stream-all
3. Response performatives:
   - reply, sorry
4. Generic informational performatives:
   - tell, achieve, cancel, untell, unachieve
5. Generator performatives:
   - standby, ready, next, rest, discard
6. Capability-definition performatives:
   - advertise, subscribe, monitor, import, export
7. Networking performatives:
   - register, unregister, forward, broadcast, route, recommend

Informatives

tell
:content <expression>
:language <word>
:ontology <word>
:in-reply-to <expression>
:force <word>
:sender <word>
:receiver <word>

untell
:content <expression>
:language <word>
:ontology <word>
:in-reply-to <expression>
:force <word>
:sender <word>
:receiver <word>

deny
:content <performative>
:language KQML
:ontology <word>
:in-reply-to <expression>
:sender <word>
:receiver <word>

Database Informatives

insert
:content <expression>
:language <word>
:ontology <word>
:reply-with <expression>
:in-reply-to <expression>
:force <word>
:sender <word>
:receiver <word>

delete
:content <performative>
:language KQML
:ontology <word>
:reply-with <expression>
:in-reply-to <expression>
:sender <word>
:receiver <word>

Query Performatives

evaluate
:content <expression>
:language <word>
:ontology <word>
:reply-with <expression>
:sender <word>
:receiver <word>

reply
:content <expression>
:language KQML
:ontology <word>
:in-reply-to <expression>
:force <word>
:sender <word>
:receiver <word>

ask-one
:content <expression>
:aspect <expression>
:language <word>
:ontology <word>
:reply-with <expression>
:sender <word>
:receiver <word>

Common Ontologies

- A shared representation is essential to successful communication and coordination
  - For humans: physical, biological, and social world
  - For computational agents: common ontology (terms used in communication)
- Current efforts are
  - Cyc
  - DARPA ontology sharing project
  - Ontology Base (ISI)
  - WordNet (Princeton)

Legal Abstractions

Legal Abstractions

- Contracts
- Directed obligations
- Hohfeldian concepts
- Compliance

Legal Concepts

- Because law involves the interactions of citizens with one another and with the government, the legal abstractions have been rich in multiagent concepts
- Traditional formalisms for legal reasoning, however, are often single-agent in orientation, e.g., deontic logic

Contracts

- Much of the law is about the creation and manipulation of contracts among legal entities
  - people
  - corporations
  - governmental agencies

The law is the study of how to break contracts!

Motivation

The legal abstractions provide a basis for agents to enter into contracts, e.g., service agreement, with each other

- Contracts
  - are about behavior
  - important in open environments

Directed Obligations

- Contracts lead naturally to one party being obliged to another party
  - more precise notion of obligation than in traditional deontic logic
  - two-party concept has a more multiagent flavor

Rights

The rights or claims a party has, as opposed to the right (ethical) thing to do

- The claims of one party are the duties of another: claim is a *correlate* of duty

Hohfeldian Concepts: 1

Hohfeld discovered that "right" is used ambiguously and proposed a uniform terminology to distinguish the various situations. Sixteen concepts result:

- Four main concepts
- Their correlates
- Their negations
- Their negations' correlates

Hohfeldian Concepts: 2

- Claim-duty: as above
- Privilege-exposure: freedom from the claims of another agent
- Power-liability: when an agent can change the claim-duty relationship of another agent
- Immunity-disability: freedom from the power of another agent

Compliance

- The legal concepts are meaningful only if satisfaction with them can be verified or falsified.

Social Commitments

Social commitments in our approach are a legal abstraction, because they subsume directed obligations as well as the Hohfeldian concepts. Importantly, they are

- public
- can be used as the basis for compliance (discussed under protocols)

Metacommitments

Agents often wish to, or must, violate their commitments. Metacommitments

- constrain the manipulation of commitments
- may involve a higher authority, the context, to adjudicate
- can be used to specify acceptable behavior

Protocols and Compliance

Commitment Protocols

- Interaction protocols can be viewed in two main lights:
  - *coordination*: ordering and occurrence of actions by the agents, e.g., FSM, Petri Nets, temporal logic (TL)
  - *commitment*: the creation and modification of the agents' commitments to one another, also formalized using TL

Compliance with Protocols

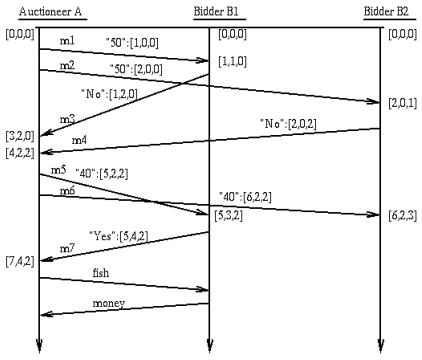
In open multiagent systems, agents are contributed by different vendors and serve different interests

- How can we check if the agents *comply* with the specified protocols?
  - Coordination aspects: traditional techniques
  - Commitment aspects: representations of the agents' commitments in TL

Verifying Compliance With Commitment Protocols

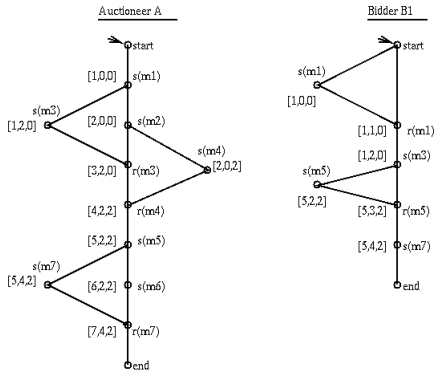
- Specification
  - models based on *potential causality*
  - commitments based on branching-time TL
- Run-time Verification
  - respects design autonomy
  - uses TL model-checking
  - local verification based on observed messages

Example: Fish-market: 1



Execution

Example: Fish-market: 2



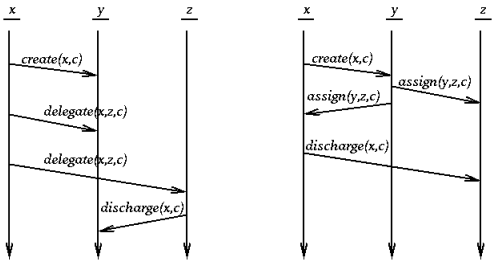
Local Models

Specification of Commitment Protocols

- Main roles and SoCom
  - metacommitments
  - protocol tokens and their meaning in terms of commitments
- Domain-specific propositions and actions
- Skeletons of roles essential for coordination

Message Patterns for Operations on Commitment

Message patterns on commitment operations ensure that the information flows to the right parties, so decisions can be made locally.



Run-time Compliance Checking

- An agent can keep track of
  - its pending commitments
  - commitments made by others that are not satisfied
- It uses this local model to see if a commitment has been violated
- An agent who benefits from a commitment can always determine if it was violated

Economic Abstractions

Motivation

The economic abstractions have a lot of appeal as an existing approach to capture complex systems of autonomous agents

- By themselves they are incomplete
- Can provide a basis for achieving some of the contractual behaviors, especially in
  - helping an agent decide what to do
  - helping agents negotiate

Market-oriented Programming

- An approach to distributed computation based on market price mechanisms
- Effective for coordinating the activities of many agents with minimal communication
- Goal: build computational economies to solve problems of distributed resource allocation

Benefits

- The state of the world is described completely by current prices
- Agents do not need to consider the preferences or abilities of others
- Communications are offers to exchange goods at various prices
- Under certain conditions, a simultaneous equilibrium of supply and demand across all of the goods is guaranteed to exist, to be reachable via distributed bidding, and to be Pareto optimal

Market Behavior

- Agents interact by offering to buy or sell quantities of commodities at fixed unit prices
- At equilibrium, the market has computed the allocation of resources and dictates the activities and consumptions of the agents

Agent Behavior

- Consumer agents: exchange goods
- Producer agents: transform some goods into other goods
- Assume individual impact on market is negligible
- Both types of agents bid so as to maximize profits (or utility)

Negotiation

Negotiation is central to adaptive, cooperative behavior

- Negotiation involves a small set of agents
- Actions are propose, counterpropose, support, accept, reject, dismiss, retract
- Negotiation requires a common language and common framework (an abstraction of the problem and its solution)

Negotiation

- A deal is a joint plan between two agents that would satisfy their goals
- The utility of a deal for an agent is the amount he is willing to pay minus the cost to him of the deal
- The negotiation set is the set of all deals that have a positive utility for every agent. The possible situations for interaction are
  - *conflict*: the negotiation set is empty
  - *compromise*: agents prefer to be alone, but will agree to a negotiated deal
  - *cooperative*: all deals in the negotiation set are preferred by both agents over achieving their goals alone

Negotiation Mechanism

The agents follow a Unified Negotiation Protocol, which applies to any situation. In this protocol,

- the agents negotiate on mixed-joint plans, i.e., plans that bring the world to a new state that is better for both agents
- if there is a conflict, they "flip a coin" to decide which agent gets to satisfy his goal

Negotiation Mechanism Attributes

- Efficiency
- Stability
- Simplicity
- Distribution
- Symmetry

e.g., sharing book purchases, with cost decided by coin flip

Third-party Negotiation

- Resolves conflicts among *antagonistic* agents directly or through a mediator
- Handles multiagent, multiple-issue, multiple-encounter interactions using case-based reasoning and multiattribute utility theory
- Agents exchange messages that contain
  - the proposed compromise
  - persuasive arguments
  - agreement (or not) with the compromise or argument
  - requests for additional information
  - reasons for disagreement
  - utilities / preferences for the disagreed-upon issues

[Sycara]

Negotiation in *RAD*

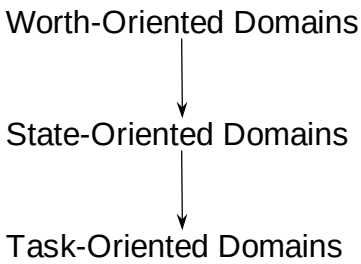
- Resolves conflicts among agents during problem solving
- To negotiate, agents exchange
  - justifications, which are maintained by a DTMS
  - class information, which is maintained by a frame system
- Maintains global consistency, *but only where necessary for problem solving*

Negotiation Among Utility-based Agents

Problem: How to design the rules of an environment so that agents interact productively and fairly, e.g.,

- Vickrey's Mechanism: lowest bidder wins, but gets paid second lowest bid (this motivates telling the truth?? and is best for the consumer??)

Problem Domain Hierarchy



Task-Oriented Domains

- A TOD is a tuple <T, A, c>, where T is the set of tasks, A is the set of agents, and c(X) is a monotonic function for the cost of executing the set of tasks X
- Examples
  - delivery domain: c(X) is length of minimal path that visits X
  - postmen domain: c(X) is length of minimal path plus return
  - database queries: c(X) is minimal number of needed DB ops

TODs

- A deal is a redistribution of tasks
- Utility of deal d for agent k is $U_k(d) = c(T_k) - c(d_k)$
- The conflict deal, D, is no deal
- A deal d is individual rational if $d \geq D$
- Deal d dominates d' if d is better for at least one agent and not worse for the rest
- Deal d is Pareto optimal if there is no d'>d
- The set of all deals that are individual rational and Pareto optimal is the negotiation set, NS

Monotonic Concession Protocol

- Each agent proposes a deal
- If one agent matches or exceeds what the other demands, the negotiation ends
- Else, the agents propose the same or more (concede)
- If no agent concedes, the negotiation ends with the conflict deal

  This protocol is simple, symmetric, distributed, and guaranteed to end in a finite number of steps in any TOD.
  What strategy should an agent adopt?

Zeuthen Strategy

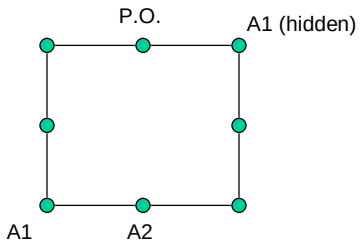
Offer deal that is best among all deals in NS

- Calculate risks of self and opponent

  R1=(utility A1 loses by accepting A2's offer) / (utility A1 loses by causing a conflict)
- If risk is smaller than opponent, offer minimal sufficient concession (a sufficient concession makes opponent's risk less than yours); else offer original deal
- If both use this strategy, they will agree on deal that maximizes the product of their utilities (Pareto optimal)
- The strategy is not stable (when both should concede on last step, but it's sufficient for only one to concede, then one can benefit by dropping strategy)

Deception-Free Protocols

- Zeuthen strategy requires full knowledge of
  - tasks
  - protocol
  - strategies
  - commitments
- Hidden tasks
- Phantom tasks
- Decoy tasks



Problem

- How to achieve coordination in a decentralized multiagent system?
- What does coordination result from?
  - Main concepts
  - Trade-offs

Underlying Tools And Infrastructure

Distributed Object Infrastructure

CORBA

- Persistence
- Externalization and Streams
- Naming and Trading
- Events
- Transactions

Persistent Object Service

- The POS
  - standardizes how an object's persistent state is stored and retrieved via operations such as *store*, *checkpoint*, and *restore* (aka *revert*)
  - provides a persistence service with an IDL interface
  - is superfluous if an OODB is available
- Variations the POS must handle
  - whether control of the storage is automatic or by the client
  - whether control is over individual objects or sets (or graphs) of them
  - what underlying systems are available
  - what is the granularity of the underlying operations

Naming Services

Essence:

- bind names to objects (their references)
- find objects given their names

Key issues:

- Representing names
- Making NSs federate, i.e., share names so that objects in different domains can be found-key to interoperability

Names

- Lots of naming conventions exist, e.g., Unix and DOS
- OMG defines a name as a sequence of name components, each component being an identifier and a type field
  - the last component binds to an object reference
  - all preceding components define successive naming contexts
  - main operations: *bind, resolve, unbind*
- It is claimed this can map to any naming convention
  - (IMHO, should work for any (tree-based) hierarchical naming scheme)

Trader

- Name service = white pages
- Trader = yellow pages + mail-order catalog
  - YP to discover objects that meet some criteria
  - Mail-order to dynamically invoke operations
- Services are given by domain-specific properties:
    - what
    - how
    - how much

Traders can federate with other traders

Event Service

Generalizes of techniques for maintaining referential integrity

- One way to maintain constraints is to notify other objects of changes in a given object
- ES separates notification from object's program logic

ORB Communication

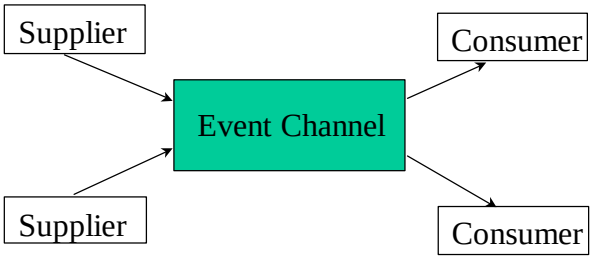
ORB communications are of 3 kinds:

- *synchronous*: sender blocks until receiver responds
- *asynchronous*: (one-way) sender doesn't wait for receiver
- *deferred synchronous*: sender proceeds independently of the receiver, but only up to a point

Execution is best effort, *at most once*

- With idempotent operations, more than once would be OK, but with nonidempotent operations it wouldn't

ES Architecture



- Event channels decouple communication
- Each event from a supplier is sent to every consumer
- Amount of storage for notifications is a QoS issue, left to implementers

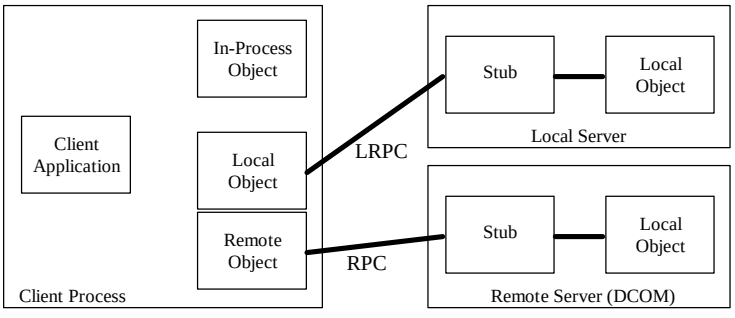
Transaction Services

OTS supports OnLine Transaction Processing

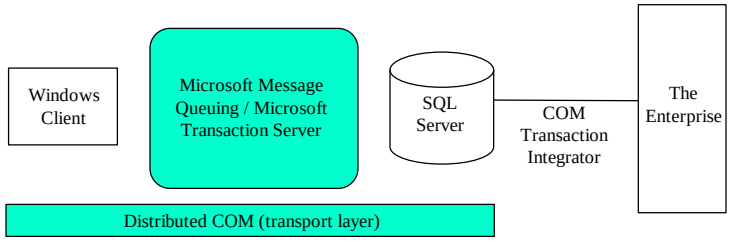
- ACID transactions: flat or nested
- Wrapping existing systems
- Interoperability of various shades, e.g.,
  - single transaction over ORB and nonORB apps
  - access to nonobject programs and resources
  - access to objects from existing programs
  - coordination over the above
- Network interoperability: >=1 OTS over >=1 ORB (4 cases)
- Flexible transaction control:
  - client determines if op is part of transaction
  - client can invoke trans and nontrans objects
  - objects can specify transaction behavior of interfaces
- TP monitors:
  - concurrency and recovery across processes

Distributed Component Object Model (DCOM)

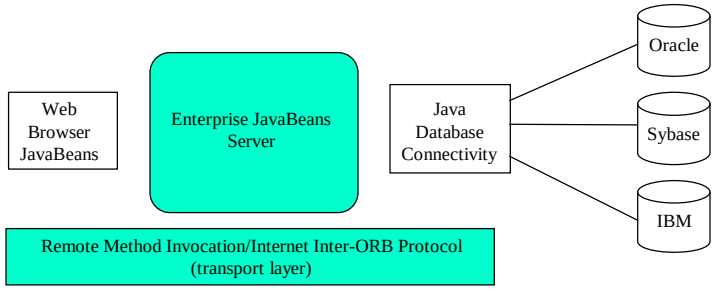
COM is an ORB; it provides an object standard and a common method of inter-ORB communication using OLE. DCOM distributes this across platforms



Microsoft Middleware Approach



Sun Middleware Approach

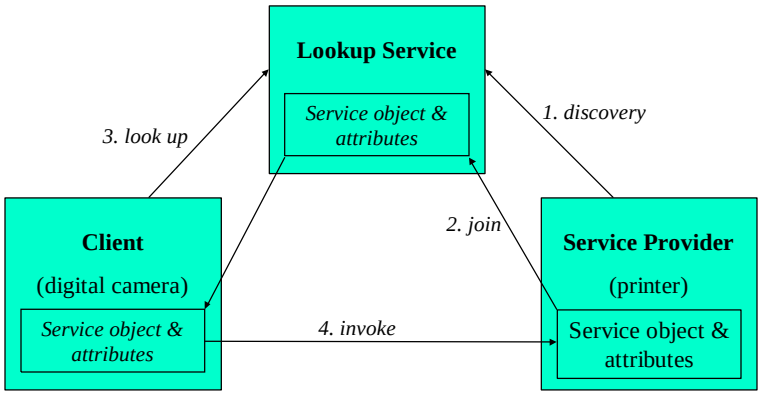


Jini Architecture

- Extends Java from one machine to a network of machines
- Uses Remote Method Invocation (RMI) to move code around a network
- Provides mechanisms for devices, services, and users to join and detach from a network

	Infrastructure	*Programming Model*	*Services*
Java	Java VM RMI Security	Java APIs JavaBeans	JNDI Enterprise Beans JTS
Java + Jini	Discovery/Join Lookup Distributed Security	Leasing Transactions Events	Printing Transaction manager JavaSpaces

Jini Services and Protocols



Jini As an Agent Infrastructure

- + Jini provides two-phase commit for transactions
- + Clients have leases on services for specific durations
- + Lookup services can be arranged hierarchically
- – Lookup service requires exact match on name of Java class (or its subclass)
- – Agents (clients & servers) interact procedurally

Mobile Agents

Mobile Agents

A computation that can change its location of execution (given a suitable underlying execution environment), both

- code
- program state

- Motivations:
  - Efficiency: move the code instead of the data
  - Upgradability: dynamically adding functionality at a remote server by sending it an agent
  - Survivability

Mobile Agent Applications

- Remote data processing
- Disconnected operation, especially for PDAs
- Testing distributed network hardware (a multihop application)

Mobile Agent Toolkit from IBM

- Aglets are mobile Java agents that can roam the Internet. They are developed using the Aglets Workbench
- Aglets are in use in Japan http://www.tabican.ne.jp/index.html will help you to find a package tour or flight that matches your requirements

Mobile Agent Technology at Mitsubishi

- Concordia - a framework for development and management of network-efficient mobile agent applications for accessing information anytime, anywhere, and on any device supporting Java. With Concordia, applications:
  - Process data at the data source
  - Process data even if the user is disconnected from the network
  - Access and deliver information across multiple networks (LANs, Intranets and Internet), using wire-line or wireless communication
  - Support multiple client devices, such as Desktop Computers, PDAs, Notebook Computers, and Smart Phones

Mobile Agent Frameworks

- Odyssey from General Magic (Java-based)
- ARA "Agents for Remote Action" from University of Kaiserslautern (Tcl, C++, Java)
- MOA "Mobile Objects and Agents" from The OpenGroup
  - uses OS process migration technology
- Concordia from Mitsubishi Electric IT Center (Java-based)
- Aglets from IBM (Java-based)
- TKQML from UMBC (Tcl and KQML)

Most allow agents to be started, stopped, moved, and monitored

Telescript

Now defunct, but interesting nevertheless

Telescript is

- object-oriented
- persistent
- interpreted
- with special primitives for communication

The telescript environment consists of

- places organized into regions with homogeneous control
- agents can travel over places
- agents must prove their credentials before entering a new region

Telescript

- Agents move from one Place to another Place to perform their assigned tasks
- Agents and Places are processes
- Agents and Places can request operations and data from each other
- Places are grouped into Clouds, each of which has a local directory (Finder database)

Challenges for Mobile Agents

Programming languages are needed that

- can express useful remote computations
- are understood at remote sites and are portable (standards)
- do not violate security of the sender or receiver
- are extensible

Understand the semantics of the different information resources being accessed

Challenges for Mobile Agents

Techniques to manage distributed computations that

- disseminate extensions to the programming language interpreter
- authenticate senders
- improve interfaces so that advanced users are not penalized while older systems are supported
- prevent deadlock
- prevent livelock
- control lifetimes
- prevent flooding of communication or storage resources

Analysis

Anything that can be done with mobile agents can be done with conventional software technology

- Mobility raises concerns of security
- Much safer to move just the code than the program state
  - No fancier than dynamic installation of functionality
  - Better to have declarative interfaces than procedural interfaces
- Mobile agents are still waiting for a killer application

Agent Tools And Projects

Agent Toolkit at IBM

- ABE "Agent Building Environment" from IBM
  - written in C++ and Java, agents have rule-based reasoning and interfaces to the web (http), news groups (nntp), and email (smtp)

Agent Toolkit at IBM

- JKQML from IBM: a framework and API for constructing Java-based, KQML-speaking software agents that communicate over the Internet. JKQML includes
  - KTP (KQML transfer protocol): a socket-based transport protocol for a KQML message represented in ASCII.
  - ATP (agent transfer protocol): a protocol for KQML messages transferred by a mobile agent that is implemented by Aglets.
  - OTP (object transfer protocol): a transfer protocol for Java objects that are contained in a KQML message

Agent Toolkit at Stanford

- The "Java Agent Template" JATLite
  - enables simple Java agents to communicate over a LAN via KQML
  - provides an agent nameserver
  - provides a message router that implements a store-and-forward capability (useful for disconnected agents)
  - enables communication via TCP/IP or email

Agent Toolkit at CMU

- RETSINA: an architecture for multiple agents that team up on demand to access, filter, and integrate information in support of user tasks

Agent Toolkit from Sandia

- JESS "Java Expert System Shell"
  - (CLIPS in Java) enables solitary, rule-based reasoning agents to be constructed
  - agents can reason about Java objects

Tools from Reticular Systems

- AgentBuilder is an integrated tool suite for constructing intelligent software agents. It consists of two major components
  - The AgentBuilder Toolkit includes tools for managing the agent-based software development process, analyzing the domain of agent operations, designing and developing networks of communicating agents, defining behaviors of individual agents, and debugging and testing agent software
  - The Run-Time System includes an agent engine that provides an environment for execution of agent software. Agents constructed using AgentBuilder communicate using KQML

Agent Toolkit from ObjectSpace

- AgentSpace is a Java-based framework for mobile agent systems developed on top of the ObjectSpace Voyager system. It provides
  - a Java multithreaded server process in which agents can be executed
  - a set of Java client applets that support the management and monitoring of agents and related resources
  - a package of Java interfaces and classes that defines the rules to build agents

Agents and Tools from AgentSoft Inc.

- AgentSoft's Web macros can be used to automate Internet and intranet jobs
- For example, LiveAgent Pro makes it easy to create scripts that gather information from all over the Web

Agent Projects at MIT

- Amalthea - ecosystem of evolving information-filtering and discovery agents that cooperate and compete in markets
- Butterfly - an agent that samples 1000s of groups and recommends ones of interest
- Expert Finder - agents who help find experts that can assist people with problems
- Friend of a Friend Finder - a network of agents that enables using social networks to get answers to personal questions
- Kasbah - a multiagent system that helps people transact goods
- Letizia - a user interface agent that helps a user browse the web by learning the user's interests and scouting ahead
- Mobile Agents for Routing Discovery - mobile agents that map dynamic network topologies

Agent Projects at MIT

- PDA@Shop - mobile agents on handheld computers for point-of-sale comparison shopping
- Remembrance Agents - proactive just-in-time memory aids that use a person's current environment to recommend information
- Straum - representing a person's Internet presence by creating an ecology of distributed agents
- Tete-a-Tete - agent-mediated integrative negotiation techniques for online merchants to differentiate their wares
- Trafficopter - a decentralized self-organizing network of agents to collect and communicate traffic information
- Yenta - an agent-based system that finds clusters of people with common interests

Other Agent Projects

- CWRU Autonomous Agents Research Group
- UMASS Distributed Artificial Intelligence Laboratory
- UNH Cooperative Distributed Problem Solving Research Group
- USC Soar Project
- Agent Projects at HCRL (Chicago)
- Stanford Nobotics Group
- Michigan Distributed Intelligent Agent Group - agents for digital libraries
- Intelligent Web Agents / Houston (IWAH) (Research Institute for Computing and Information Systems - University of Houston)
- Intelligent Autonomous Software Agents for Virtual Environments and other areas of telematics at the Austrian Research Institute for Artificial Intelligence

Other Agent Projects

- MAS Research at Vrije Universiteit Brussels (VUB)
- Distributed Artificial Intelligence at the Dept of Information Engineering of PARMA University
- Knowledgeable Community Project (Nishida Lab.)
- DAI Research Unit at QMW Electronic Engineering Department specializes in building real-world multiagent systems
- Multi-Agent Systems Research Group at Université de Laval
- CALVIN : Communicating Agents Living Vicariously In Networks - KSL (NRC - CNR)
- The Multi-Agent Systems Group of the University of Maastricht
- DAI at Geneva University Hospital
- HUJI DAI group

Conclusions

Research Trends in Agents

- Social and organizational behavior
- Multiagent learning
- Formal Methods
- Negotiation
- Interaction-Oriented Programming

Research Trends in EC

- Virtual enterprises
- Supply chain management
- Auction theory
- Mechanism design
- Personalization

Recurring Challenges

- Design rules for systems with autonomous, heterogeneous components
- Security and trust concerns in open environments
- Scalability
- User interfaces

Elements of Trust

Ultimately, what we would like is to trust our agents. Trust involves

- having the right capabilities
- following legal contracts where specified
- supporting one's organization or society
- being ethical
- failing all else, being rational

Lessons: 1

- Advanced abstractions can help a lot in multiagent systems by helping
  - understand how MAS will be used
  - specify MAS in high-level terms
  - design MAS in a principled manner
  - validate MAS with respect to user needs
- But the abstractions must
  - be conceptually simple and well-grounded in theory
  - reflect true status of the system, not just a nice-sounding buzzword

Lessons: 2

- Most real-world problems are mundane
  - integrate advanced technology into existing systems
  - user interfaces and database systems deserve much attention
  - adaptive approaches are better, because the specification is usually not available at the outset
- Don’t mess with autonomy and heterogeneity
- Do it locally
- Be problem-driven, not solution-driven

To Probe Further

- *Readings in Agents* (Huhns & Singh, eds.), Morgan Kaufmann, 1998
  http://www.mkp.com/books_catalog/1-55860-495-2.asp
- IEEE Internet Computing, *http://computer.org/internet*
- DAI-List-Request@ece.sc.edu
- International Conference on Multiagent Systems (ICMAS)
- International Joint Conference on Artificial Intelligence
- ACM Conference on Electronic Commerce (EC)
- International Workshop on Agent Theories, Architectures, and Languages (ATAL)