A Security Based Model for Mobile Agent Software Systems

Haiping Xu, Ph.D.
Computer and Information Science Department
University of Massachusetts Dartmouth

http://www.cis.umassd.edu/~hxu

Outline

- Part 1: Background and Motivations
- Part 2: Agent-Oriented G-Net Model
- Part 3: Design of Intelligent Mobile Agents
- Part 4: Design of Facilitator Agents
- Part 5: A Case Study: Agent Migration
- Part 6: Conclusions and Future Work.

Part 1: Background and Motivations

- The development of software systems starts with two main activities:
  - Software requirements analysis
  - Software design
- Software requirements analysis: to reduce potential errors caused by incomplete and ambiguous requirements
- Software design: to depict the overall structure of a system by decomposing the system into its logical components.

Formal Methods in Software Engineering

- The purpose of software requirements analysis can be achieved in two ways:
  - Write a specification in natural languages
  - Choose a formal language, e.g., Petri nets
- Ideally, formal methods can be applied in each phase of the software development life cycle, e.g., the design phase
- However, to create a formal model in the design phase and to verify its correctness is rare.

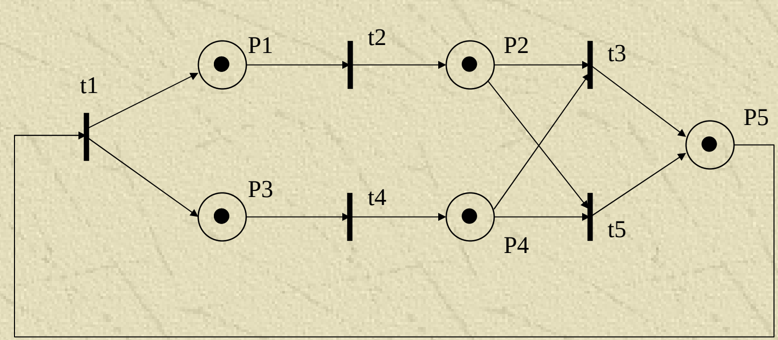
Introduction to Petri Net

- "Three-in-one" capability of Petri net models [Murata 1989]
  - Graphical representation
  - Mathematical description
  - Simulation tool
- Definition:

A Petri net is a 4-tuple, PN = (P, T, F, M_0) where

P = {P1, P2, …, Pm} is a finite set of places;
T = {t1, t2, …, tn} is a finite set of transitions;
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation);
M_0: P --> {0, 1, 2, 3, …} is the initial marking.

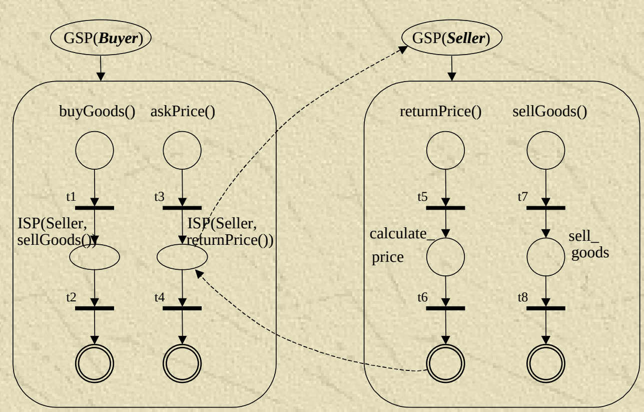
An Example



G-Net: A High Level Petri Net

- Defined to support modeling of systems as a set of independent and loosely-coupled modules [Deng *et al.* 1993]
- Provides support for incremental design and successive modification
- Is not fully object-oriented due to a lack of support for inheritance.

An Example



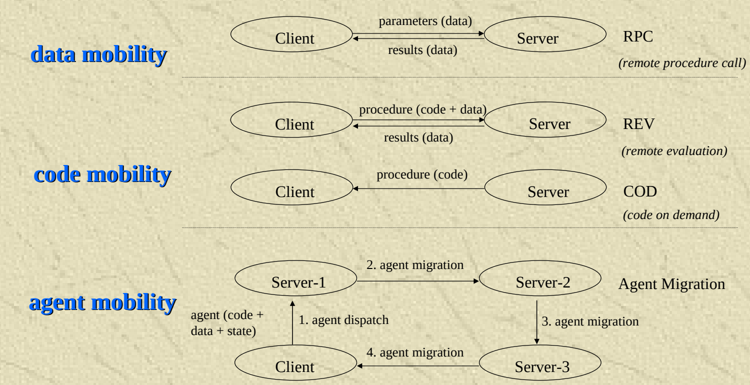
Introduction to Agents

- The term “agent” comes from the Greek word “agein”, which means to drive or to lead
- A software agent is a program that acts on behalf of a (human) user
- A software agent is typically situated in some *environment*, and that is capable of *autonomous action.*

Research Directions

- Multi-agent systems (MAS)
  - Agents act as “active” objects (intelligence)
  - Collaborative or competitive
  - Generally use distributed but static (non-mobile) agents
- Mobile agents (MA)
  - Model agent mobility and agent coordination
  - Generally assume very limited or even no intelligence.

Evolution of the Mobile Agent Paradigm



Why Mobile Agent ?

- Asynchronous Tasks
  - Asynchronous processing of requests
  - Mobile device can be disconnected and reconnected
- Reduction of Communication Costs
  - The number of interactions
  - The amount of data communicated over the network.

Academic Research Work

- Formal models for agent mobility
  - Distributed join-calculus: an extension of π-calculus that introduce the explicit notions of named localities and distributed failure [Fournet *et al.*, 1996]
  - Mobile UNITY: a programming notation that captures the notion of mobility and transient interaction among mobile nodes [Roman *et al.*, 1997]
  - MobiS: an extended version of PoliS, which is a specification language based multiple tuple spaces [Mascolo, 1999]
  - LIME: a middleware based on tuple spaces [Murphy *et al.*, 2001]
- Very few attempts to formally model agent security for mobile agents
  - The use of encrypted functions for mobile agent security, which protects mobile agents from malicious hosts [Lee and Harrison, 2004]
  - Mobile agent security through multi-agent cryptographic protocols [Tate and Xu, 2003].

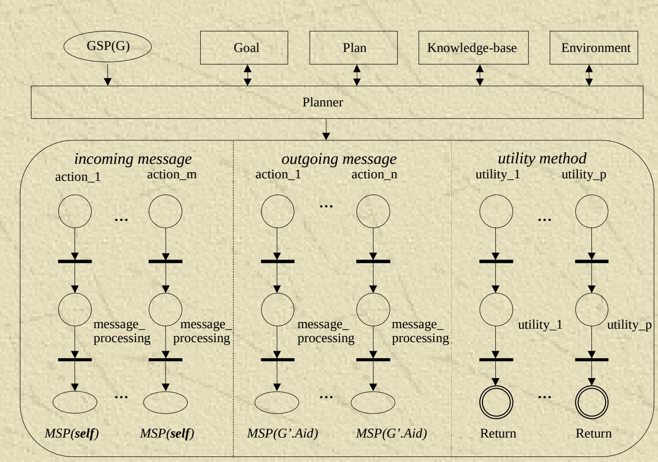
Challenges

- Security issues for mobile agent systems
  - Inter-agent security
  - Agent-host security
  - Inter-host security
- Most of the existing work concentrates on solving one of the above problems
- In contrast, our approach provides a uniform framework to deal with all the above security issues.

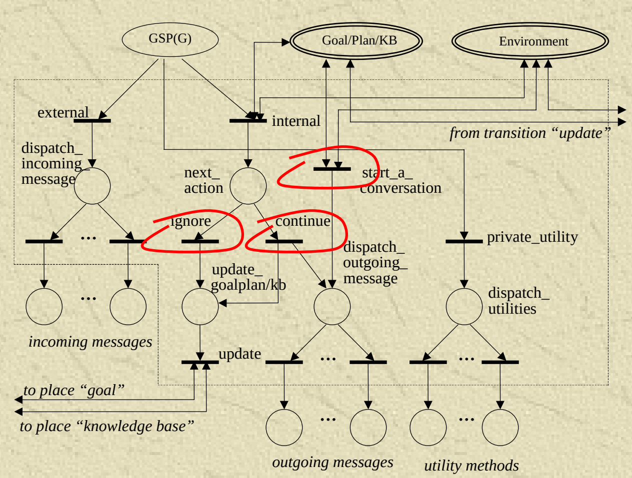
Part 2: Agent-Oriented G-Net Model

- Software agent systems: one of the most important topics in distributed and autonomous decentralized systems
- Key features: autonomous, reactive, proactive and internally-motivated agents
- However, the G-net model is not sufficient for agent modeling because:
  - Does not support a common communication language and common protocols among agents
  - Does not directly support asynchronous message passing
  - Does not support modeling agent's mental state, such as goals, plans and knowledge.

An Agent-Based G-Net Model



A Template of *Planner* Module



Formal Definitions of Agent-Based G-Net Model

Definition 3.1 Agent-Based G-Net

An *agent-based G-net* is a 7-tuple *AG = (GSP, GL, PL, KB, EN, PN, IS)*, where *GSP* is a *Generic Switch Place* providing an abstract for the agent-based G-net, *GL* is a *Goal* module, *PL* is a *Plan* module, *KB* is a *Knowledge-base* module, *EN* is an *Environment* module, *PN* is a *Planner* module, and *IS* is an *internal structure* of *AG*.

Definition 3.2 Planner Module

A *Planner module* of an agent-based G-net *AG* is a colored sub-net defined as a 7-tuple *(IGS, IGO, IPL, IKB, IEN, IIS, DMU)*, where *IGS*, *IGO*, *IPL*, *IKB*, *IEN* and *IIS* are interfaces with *GSP*, *Goal* module, *Plan* module, *Knowledge-base* module, *Environment* module and *internal structure* of *AG*, respectively. *DMU* is a set of decision-making unit, and it contains three abstract transitions: *make_decision*, *sensor* and *update*.

Definition 3.3 Internal Structure (IS)

An *internal structure (IS)* of an agent-based G-net *AG* is a triple *(IM, OM, PU)*, where *IM/OM* is the *incoming/outgoing message* section, which defines a set of *message processing units (MPU)*; and *PU* is the *private utility* section, which defines a set of *methods*.

Definition 3.4 Message Processing Unit (MPU)

A *message processing unit (MPU)* is a triple *(P, T, A)*, where *P* is a set of places consisting of three special places: *entry* place, *ISP* and *MSP*. Each *MPU* has only one *entry* place and one *MSP*, but it may contain multiple *ISP*s. *T* is a set of transitions, and each transition can be associated with a set of guards. *A* is a set of arcs defined as: ((P-{*MSP*}) x T) ∪ ((T x (P-{*entry*}).

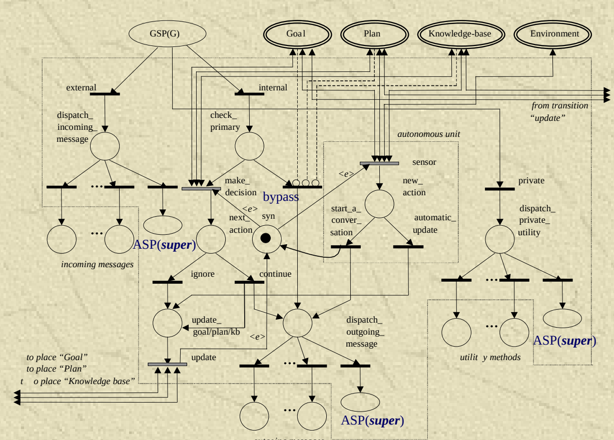
Definition 3.5 U-Method

A *U-Method* or *method* is a triple *(P, T, A)*, where *P* is a set of places with three special places: *entry* place, *ISP* and *return* place. Each method has only one *entry* place and one *return* place, but it may contain multiple *ISP*s. *T* is a set of transitions, and each transition can be associated with a set of guards. *A* is a set of arcs defined as: ((P-{*return*}) x T) ∪ ((T x (P-{*entry*}).

A Framework for Modeling Agent-Oriented Software

- To support inheritance, we revise the *planner* module:
  - *Abstract transition*: represents abstract units of decision-making or mental-state-updating (with synchronization)
  - *Autonomous unit*: makes an agent autonomous and internally-motivated
  - *Asynchronous Superclass switch Place (ASP)*: used to forward a MPU or a method call (token) to a “superclass” model in the case of inherited communication mechanisms.
- Show the useful role of inheritance in agent-oriented software design.

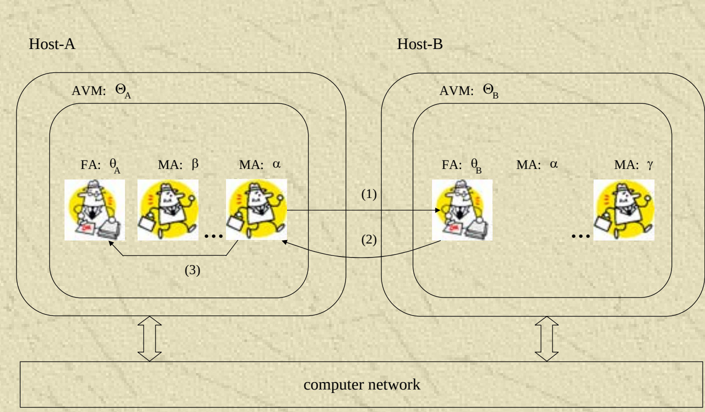
A Template for the *Planner* Module



Part 3: Design of Intelligent Mobile Agents – A Generic Model

- Two schemes for agent development:
  - Weak agent approach
  - Strong agent approach
- Most of the existing work on mobile agents use weak agent approach (not flexible, security issues …)
- In contrast, we propose a generic model for *intelligent* mobile agent.

Agent World Architecture



(1) move-request (2) grant (3) notify (4) move

Formal Definitions of Agent World Architecture

Definition 3.1 *Agent World (AW)*

An *agent world (AW)* is a 3-tuple *(WKHOST, SHOST, HCOM)*, where *WKHOST* is a well-known static host, which is responsible for recording the most recent address and public key of all other hosts and for issuing certificates to the FAs in SHOST. *SHOST* is a set of hosts that can provide the services of an agent virtual machine. *HCOM* is the communication protocol among hosts in *SHOST*; an example of such protocols is TCP/IP.

Definition 3.2 *Agent Virtual Machine (AVM)*

An *agent virtual machine (AVM)* is a 5-tuple *(IFA, SIMA, HOSTIP, ID)*, where *IFA* is a facilitator for *AVM*, which is responsible for recording information of mobile agents running on that *AVM*, and also for providing services for mobile agents running on the AVM. *SMA* is a set of mobile agents. *HOSTIP* is the current IP address of the host that is supporting this *AVM*, and *ID* is a unique identifier for that *AVM*.

Definition 3.3 *Static Host (SH) and Mobile Host (MH)*

A host is a 4-tuple *(SAVM, ACOM, HOMEIP, CURIP)*, where *SAVM* is a set of *agent virtual machines (AVM)*. *ACOM* is the communication protocol among AVMs in *SAVM*, and examples of such protocols are IPC and TCP/IP. *HOMEIP* is the original IP address of the host, and *CURIP* is the current IP address of the host. If at any time, *CURIP = HOMEIP*, we call the host a *static host (SH)*; otherwise, we call it a *mobile host (MH)*.

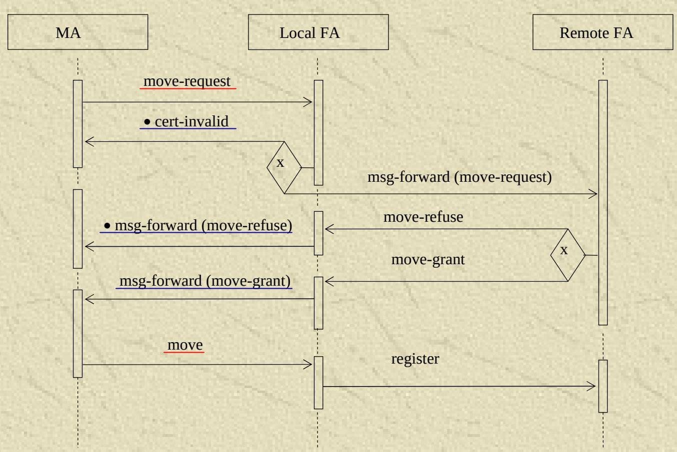
Definition 3.4 *Static Agent (SA) and Mobile Agent (MA)*

An *agent A* is a 3-tuple *(HOMEIP, CURIP, AO)*, where *HOMEIP* is the IP address of the host on which agent *A* is created. *CURIP* is the IP address of the host supporting agent *A*. *AO* is the agent object with the general structure as we described in Section 2. If at any time, *CURIP = HOMEIP*, we refer to agent *A* as a *static agent (SA)*; otherwise, we refer to agent *A* as a *mobile agent (MA)*.

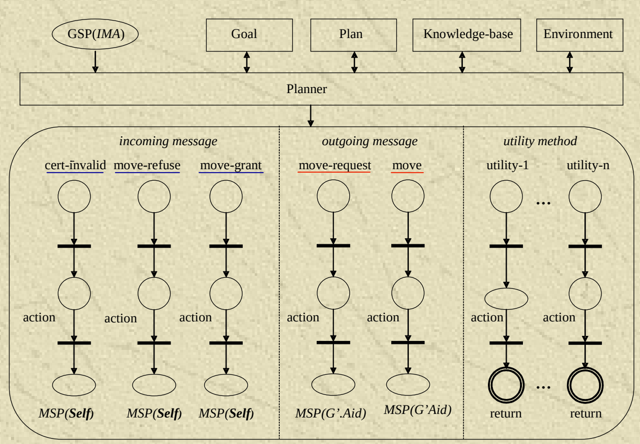
Security Consideration

- If a mobile agent is allowed to communicate with a remote facilitator agent or any other mobile agents directly
  - Both mobile agents and facilitator agents are responsible for security checking all other facilitator agents and mobile agents
- Use the facilitator agents as a middleware for agent communications and agent migration
  - Communications between local mobile agents
  - Communications between a local mobile agent and a remote facilitator agent
- Security checking become more efficient and reliable
  - Mobile agents are only responsible for security checking its local facilitator agent
  - Facilitator agents are only responsible for security checking its local mobile agents and any remote facilitator agents.

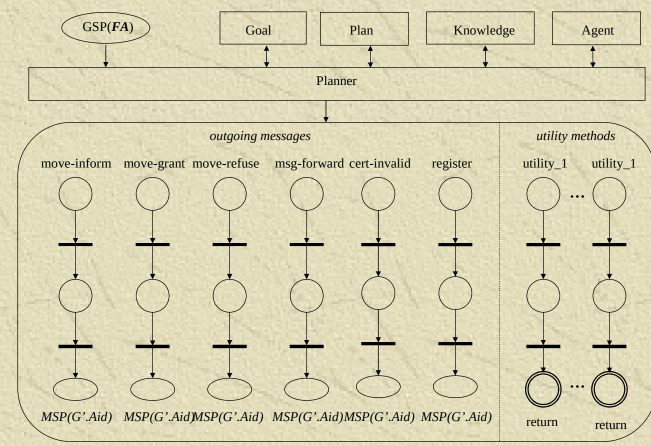
Agent Interaction Protocol for Agent Migration



Intelligent Mobile Agent (IMA)

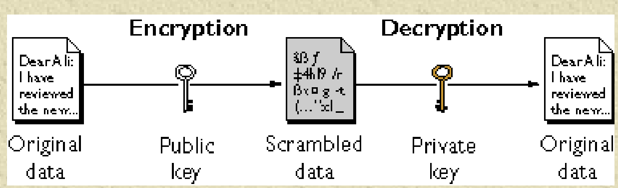


Part 4: The Facilitator Agent Model



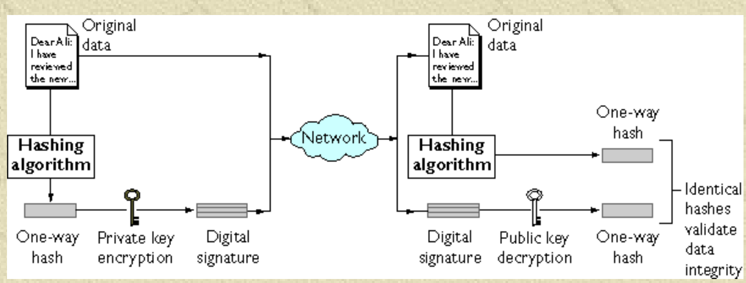
Cryptographic Mechanisms

- Public-key cryptograph is one of the most widely used encryption mechanism on the Internet
- Involves a pair of keys — a public key and a private key



Cryptographic Mechanisms (cont'd)

- Use digital signature to authenticate the message sender
- Use a one-way hash (also called a message digest)
  - The value of the hash is unique for the hashed data
  - The content of the hashed data cannot be deduced from the hash.



04/9/2004 CIS Dept., UMass Dartmouth 29

Cryptographic Mechanisms (cont'd)

- A certificate is an electronic document used to identify an entity and to associate that identity with a public key
- A certificate also includes the name of certificate holder, an expiration date, the issuer's name, a serial number etc.
- Most importantly, a certificate always includes the digital signature of the issuer.

04/9/2004 CIS Dept., UMass Dartmouth 30

Certificate/Passport/Visa Approach

- User assigns a certificate to a mobile agent when it is created
  - Contains info such as issuer's name, public key etc.
  - Is recognizable by the local facilitator agent
  - Is not recognizable by a remote facilitator agent
- Local facilitator agent assigns a passport to the mobile agent to replace the initial certificate
- A mobile agent can use the passport to apply for a visa from a foreign facilitator agent.

Structure of Certificate, Passport and Visa Stamp

```
Struct Certificate {
    int serial_number;      // the serial number of the certificate
    String issuer_name;     // the issuer's name
    String name;            // the name of holder
    Privilege privilege;    // the privilege assigned by the issuer
    String public key;      // the public key of the holder
    Time valid_time;        // the valid time for the certificate
    Signature signature;    // the encrypted value of the above items
                            // encoded by the issuer's private key
}

Struct Passport {
    Certificate passport;   // issued by the local facilitator agent
    Visapage visapages;     // visa pages to hold visa stamps
}

Struct Visapage {
    Certificate visaStamp; // the same structure as a certificate
    Visapage nextVisapage; // visa is defined as linked list
}
```

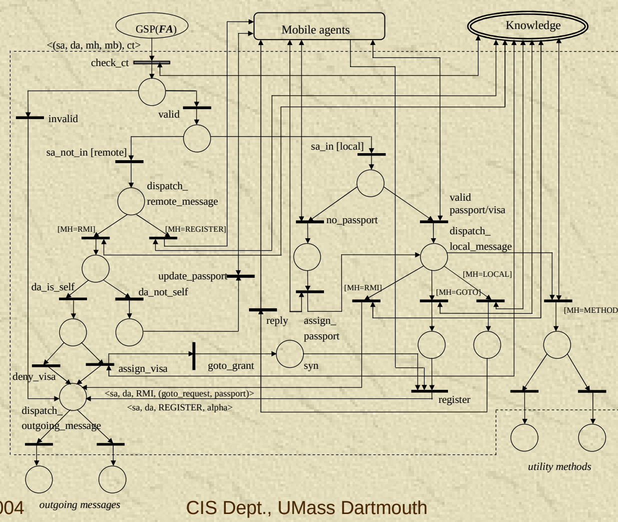
Encrypting Messages

- Each message MSG is first encrypted by the sender's private key: $(MSG)_{Ks^{-1}}$
- Then combined with the sending agent's certificate/passport: $((MSG)_{Ks^{-1}}$, certificate)
- Finally encrypted by the receiver's public key: $((MSG)_{Ks^{-1}}, \text{certificate})_{Kr}$.

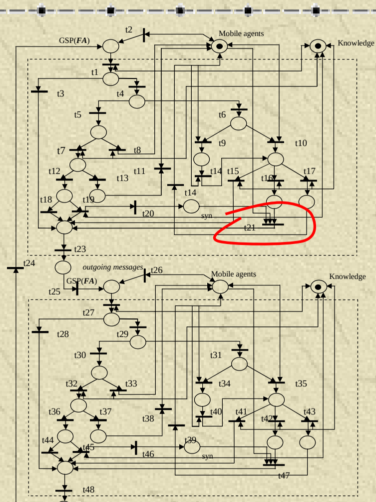
Structure of a Message

```
Struct Message {
   AgentID sa;        // source agent
   AgentID da;        // destination agent
   Head mh;           // message head
   String mb;         // message body
   FileNode fileRef;  // binary attachments
}

enum Head {RMI, GOTO, REGISTER, METHOD, LOCAL};

Struct FileNode {
   File file;
   FileNode nextFile;
}
```

The Planner Module of Facilitator Agent (initial design)



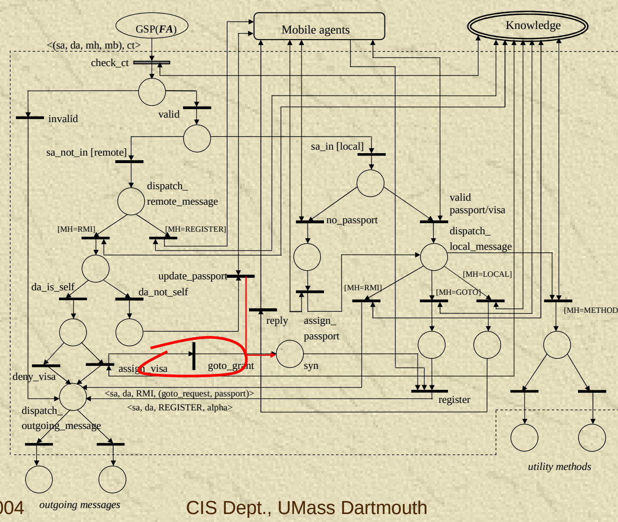
Part 5: A Case Study: Agent Migration



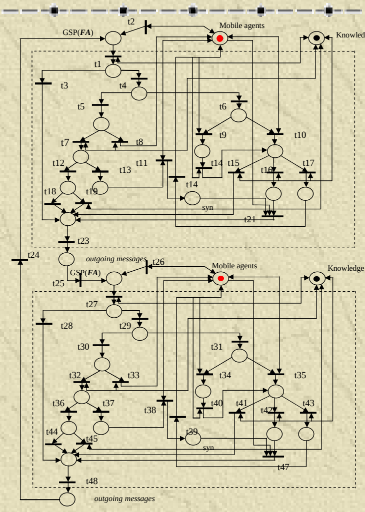
local facilitator agent

remote facilitator agent

Redesign of the *Planner* Module



Example of Agent Migration



Local facilitator agent

Remote facilitator agent

Advantages of Our Approach

- Application-specific mobile agent class can be defined as a subclass of IMA
- Security checking for mobile agents is efficient and reliable due to localization
- The resulting Petri net model can be used as a foundation for formal Petri net analysis and simulation techniques.

Part 6: Concluding Comments

- There is an increasing need to ensure that complex software systems are robust, reliable and fit for purpose (Agent-Oriented SE)
- Petri nets provide a formal and visual model with natural expression for concurrency and coordination
- Adapt Petri net models to define a security-based model for mobile agent software system.

Future Work

- Study various security issues in mobile agent design, especially the efficiency and reliability of different security protocols
- Design and develop a compilation process to automatically build security protocols into our existing agent models
- Develop a model-based mobile agent development environment (M-MADE) for rapid agent design and implementation (i.e., synthesis of the work).

Thanks for your attention!

The slides for this talk may be downloaded from

http://www.cis.umassd.edu/~hxu