

Secure Service Provision in Ad Hoc Networks

Radu Handorean

5 December 2003

Mobile Computing Laboratory

Department of Computer Science and Engineering

 Washington University in St. Louis

Service Oriented Computing

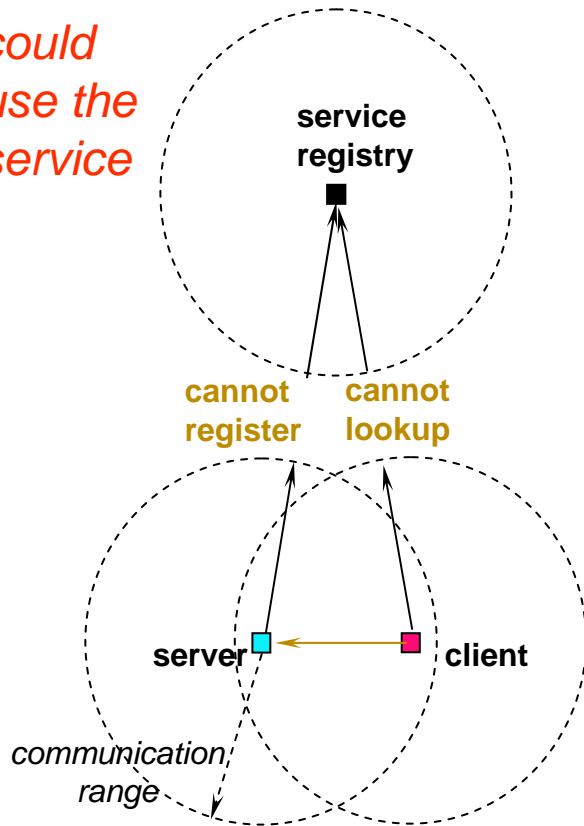
- End user and business process requirements drive the service provision dynamics
- Expectations focus on high levels of availability, rapid growth, and predictable performance
- The network is a service support infrastructure
- Hosts can advertise, discover, and use services
- Service registry function extends into the semantic domain
- Services can discover and use other services

Ad Hoc Networks

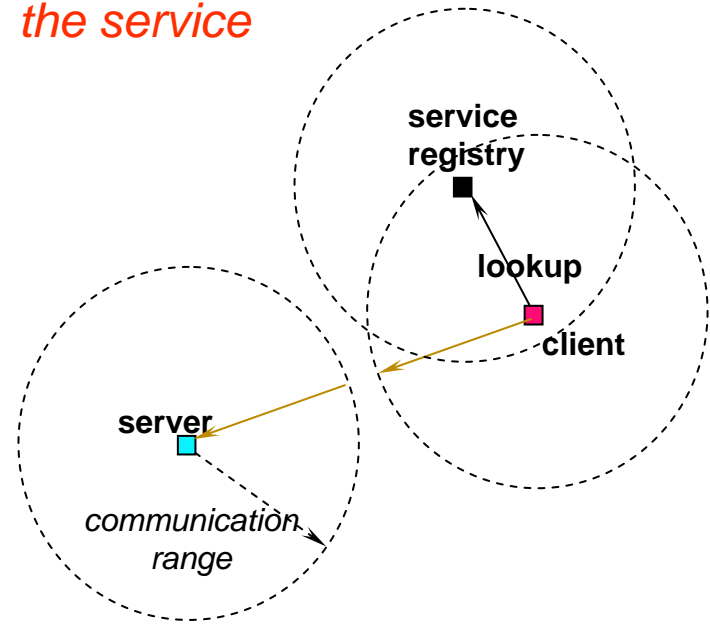
- Wireless communication
- Lack of fixed infrastructure
- Frequent disconnections
- Limited guarantees
- Resource-poor participants
- Opportunistic resource usage
- Open environment

Impact of Disconnection

could use the service



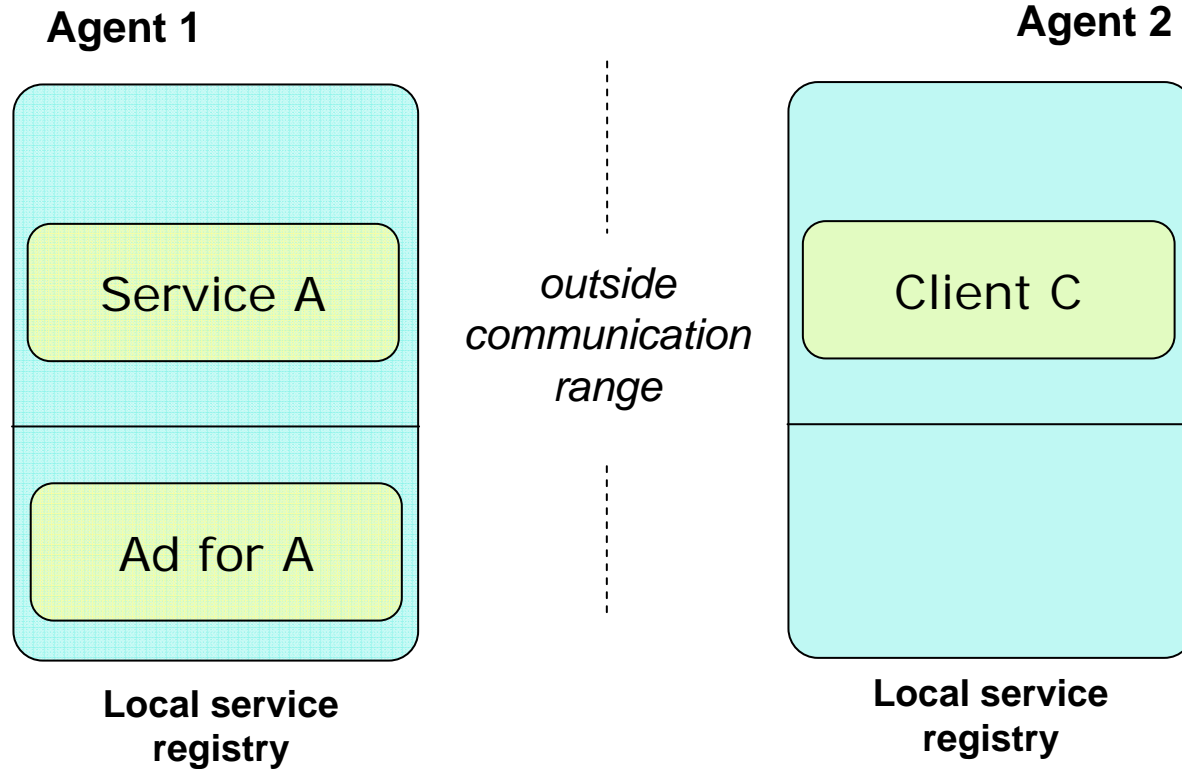
cannot use the service



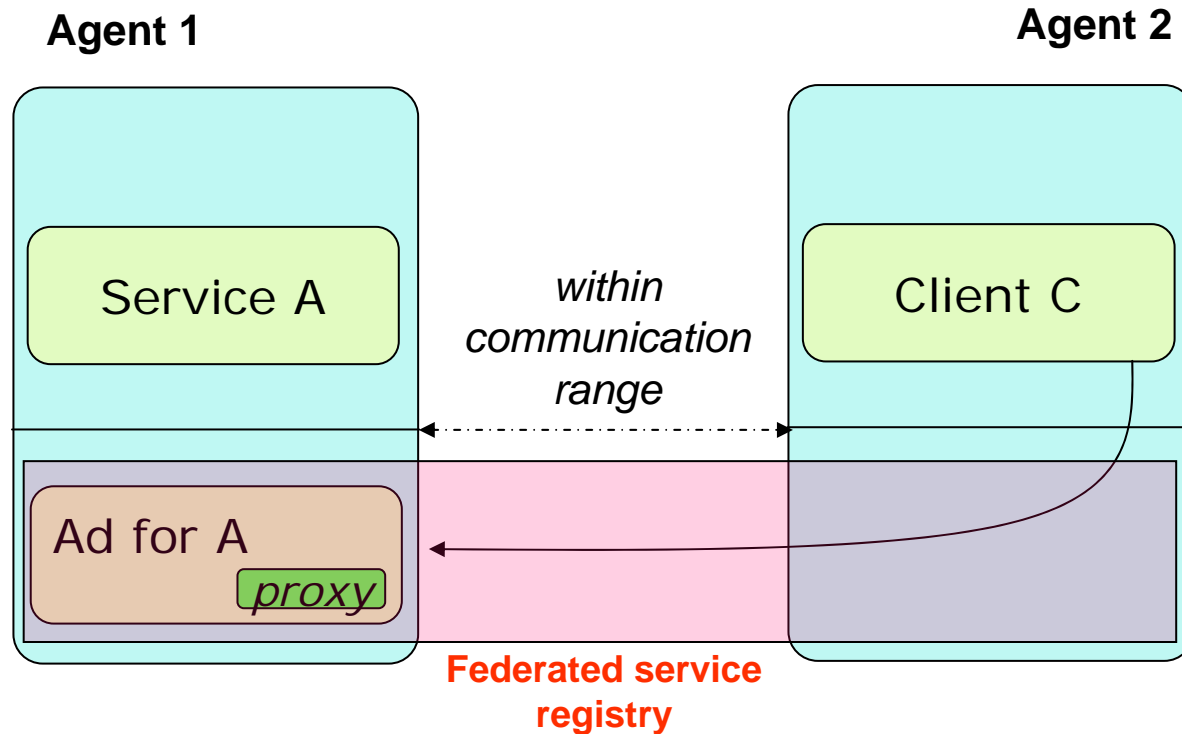
First Entry into the Ad Hoc Setting

- Eliminate the need for centralized support
 - ❑ Distributed approach to advertising and discovery
- Create the illusion of local interaction
 - ❑ Locally owned service registries
 - ❑ Proxies designed to hide the communication mechanics
- Manage mobility and disconnection
 - ❑ Atomic updating of service availability
 - ❑ Continued service in the presence of mobility
- Secure the service discovery process
- Secure the communication

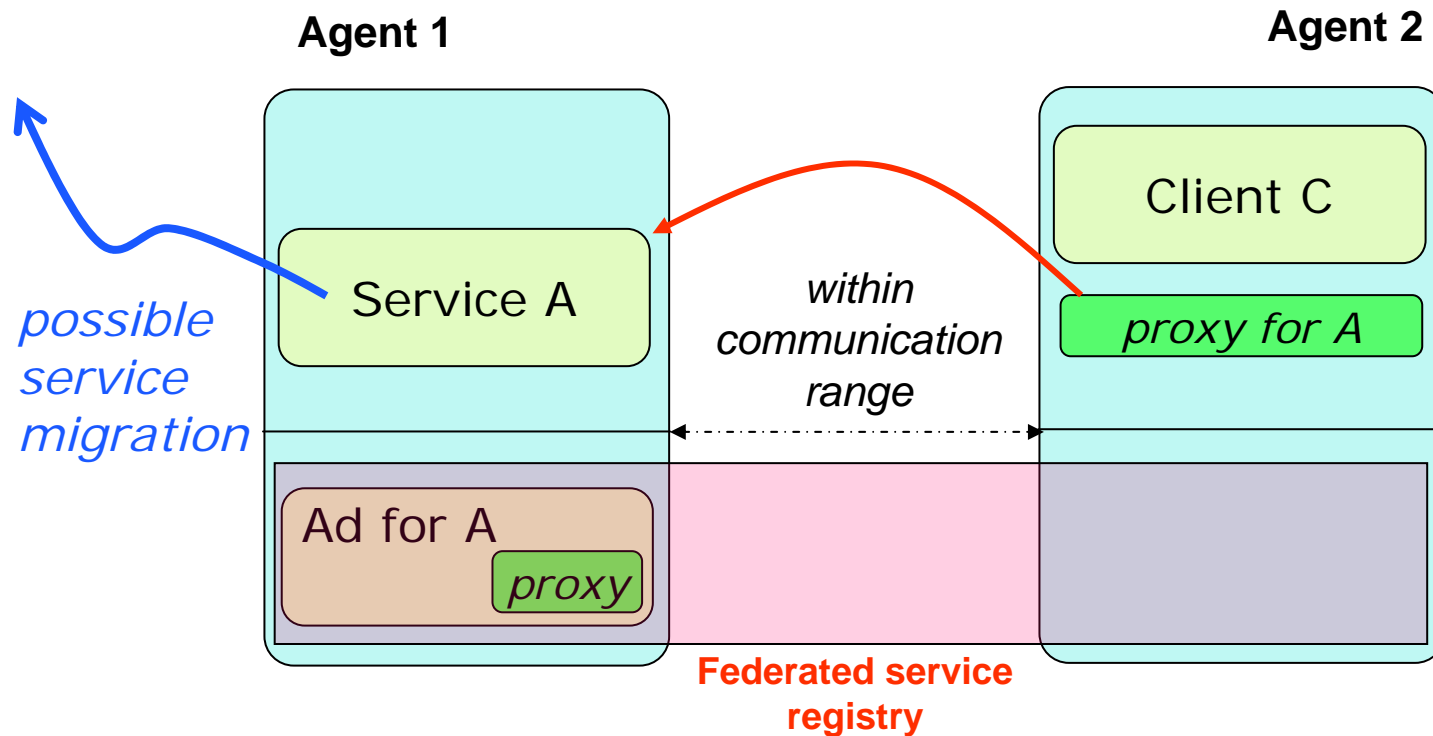
Server/Client Duality



Registry Access and Update

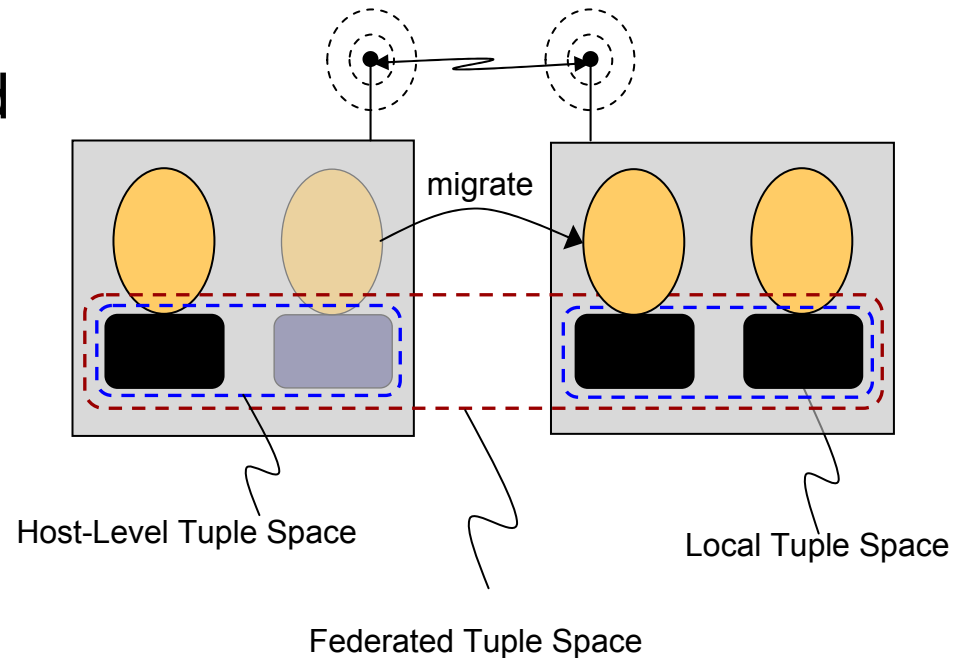


Service Transparency

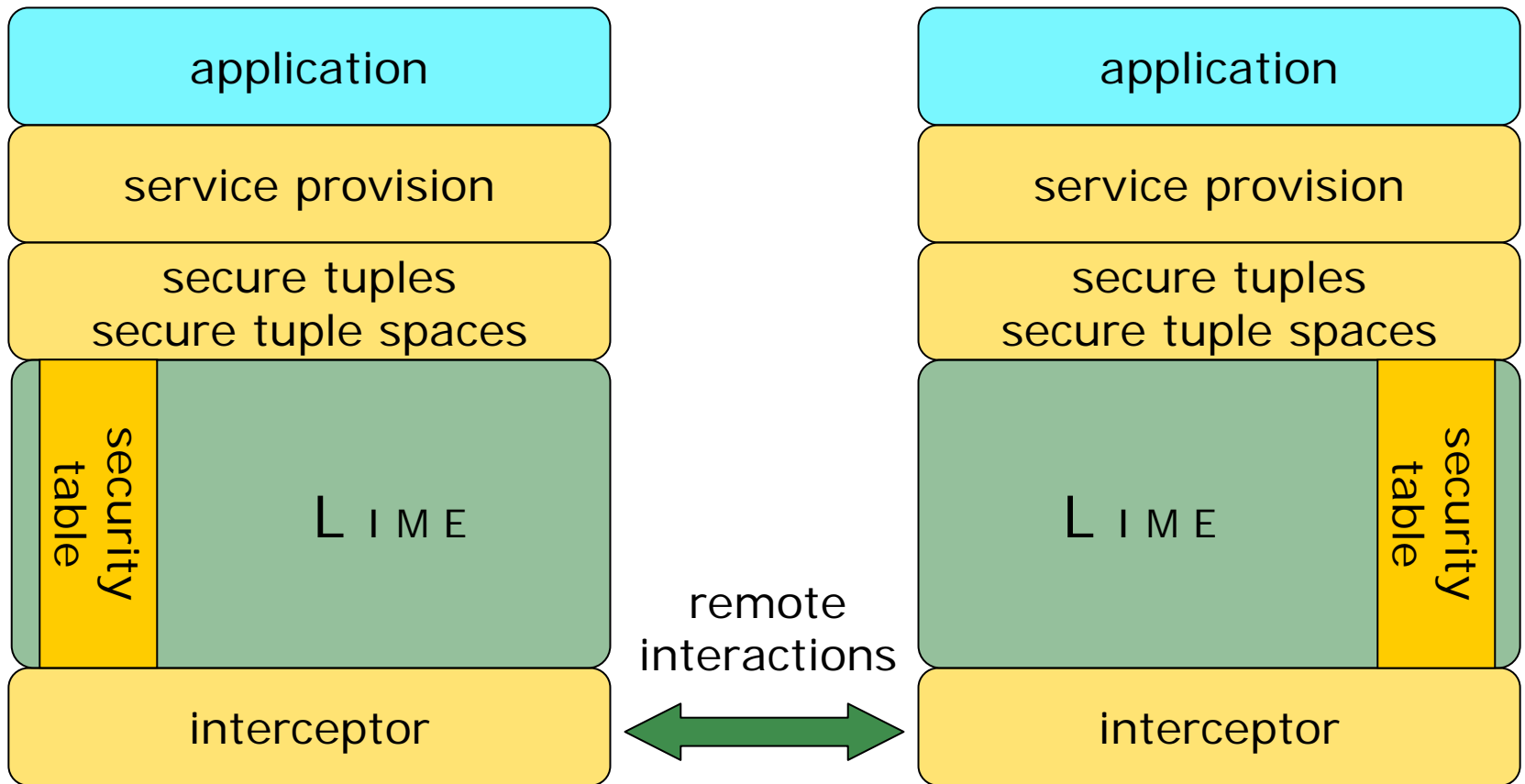


Implementation Base — LIME

- A coordination model supporting physical and logical mobility
- Linda-like tuple space coordination
- Transient tuple space sharing
- Java implementation



Design Overview



Service Representation

- Service profile
 - ❑ capabilities
 - ❑ attributes
 - ❑ proxy
- Service advertisement—tuple representation
[InkJet("Yes"), PgPerMin(25), RemoteHandle(proxy)]
- Service discovery—template specification
[InkJet("Yes"), PgPerMin.class, PrinterInterface.class]

Vulnerability

- LIME System Tuple Space (LSTS) supports explicit context-awareness by exposing
 - hosts
 - agents
 - tuple spaces
- Identically-named tuple spaces are shared
- All tuple space names appear in LSTS
- The name offers access to the entire federated tuple space

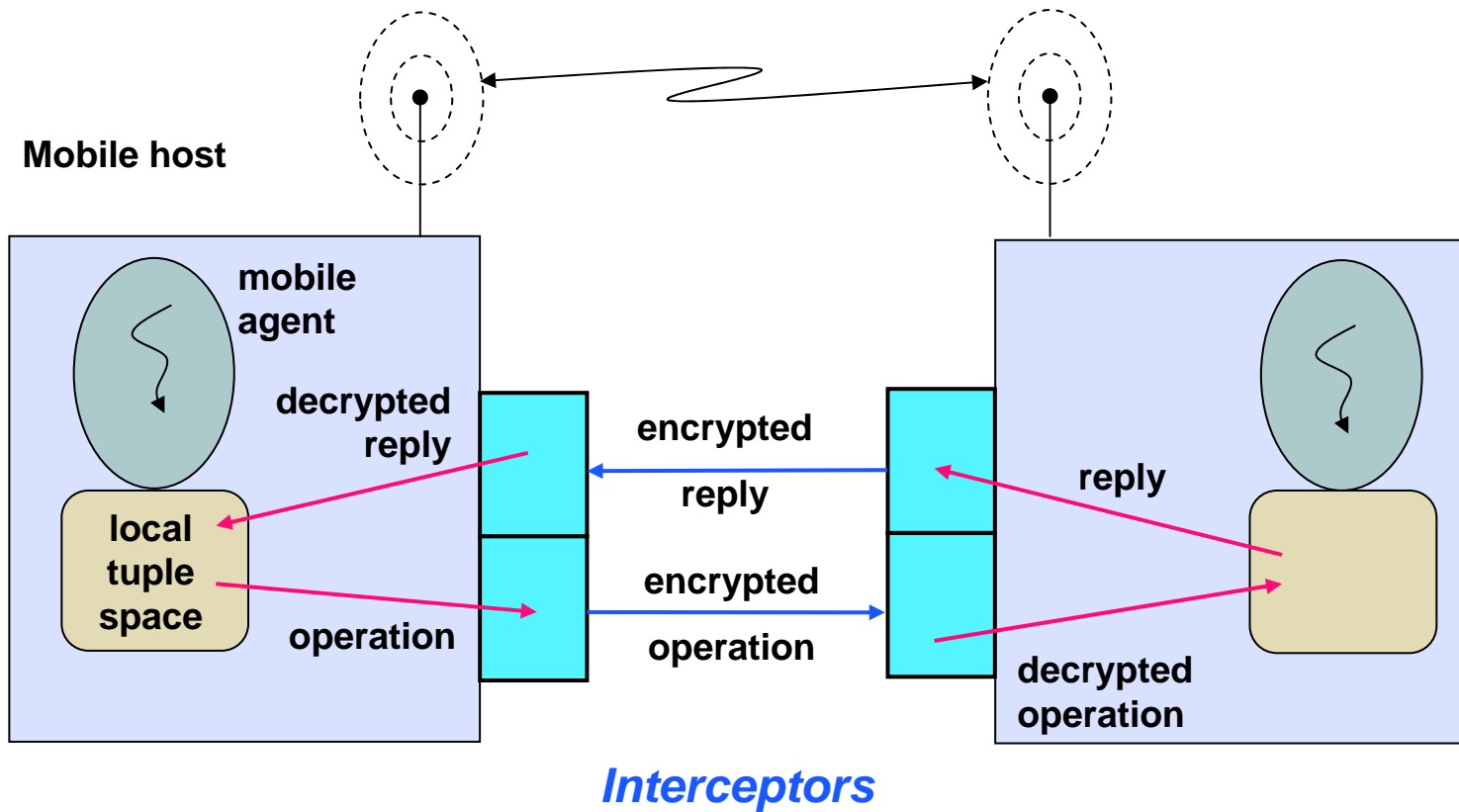
Tuple Space Protection

- Provide password protection
 - $RED \neq RED + PWD$
- Render the LSTS information useless
 - Use internally added prefixes to tuple space names
 - Public: RED becomes U_RED
 - Private: $RED + PWD$ becomes $S_Xo*3)@r\#2$
- Tuple space handles are not transferable
- The password is needed only once

Secure Communication

- Vulnerability:
 - ❑ Transmission consists of Java serialized objects
- Design opportunity:
 - ❑ Tuple space names appear in all messages
- Solution path:
 - ❑ Use symmetric encryption to secure communication channels
 - ❑ Use the password to derive the key
 - ❑ Use interceptors at both ends

Interceptor Pattern Application



Tuple-Level Protection

- Vulnerability:
 - ❑ Service profiles need to be widely accessible
 - ❑ Agents can access the entire content of the tuple space
 - ❑ Polymorphic matching offers convenient access
 - ❑ Tuples cannot be encrypted
- Solution strategy:
 - ❑ Offer tuple-level remove/read passwords
 - ❑ Control tuple access by extending existing capabilities

[InkJet("Yes"), PgPerMin(25), RemoteHandle(proxy), rd_pwd, rm_pwd]

Conclusions

- We solved the service repository consistency problem
 - Connectivity-bound service discovery
- We support distributed peer-to-peer service discovery
 - We eliminated the need for any third party support
- We include multiple degrees of protection
 - Tuple space, tuple, host-to-host communication
- We can support public key distribution
 - Advertise public keys in read-only tuples
 - Limited authentication capabilities

Future Work

- Define requirements for service provision in the new setting
- Evaluate spatiotemporal aspects of service provision in ad hoc settings
- Exploit motion profiles and explore ways to acquire them
- Develop support for context sensitive binding