

Ambient References

Addressing Objects in Mobile Networks

Tom Van Cutsem



Programming Technology Laboratory
Vrije Universiteit Brussel



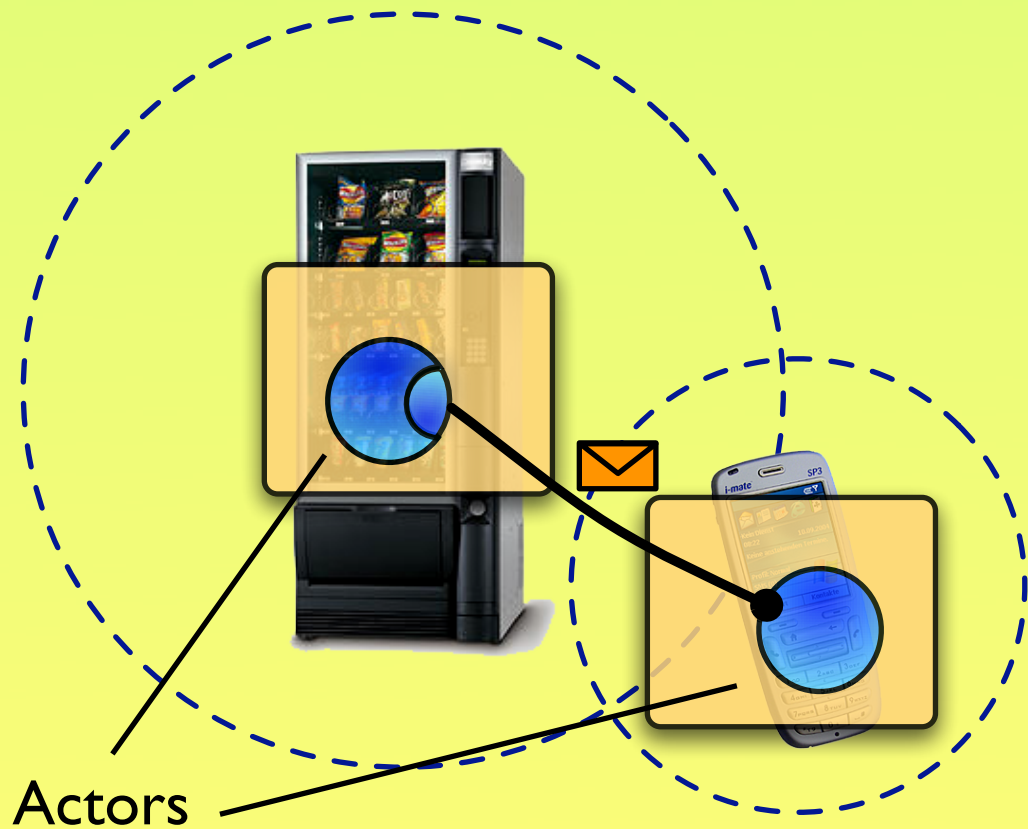
Mobile Networks

Pervasive/Ubiquitous Computing



Mobile Networks

Pervasive/Ubiquitous Computing

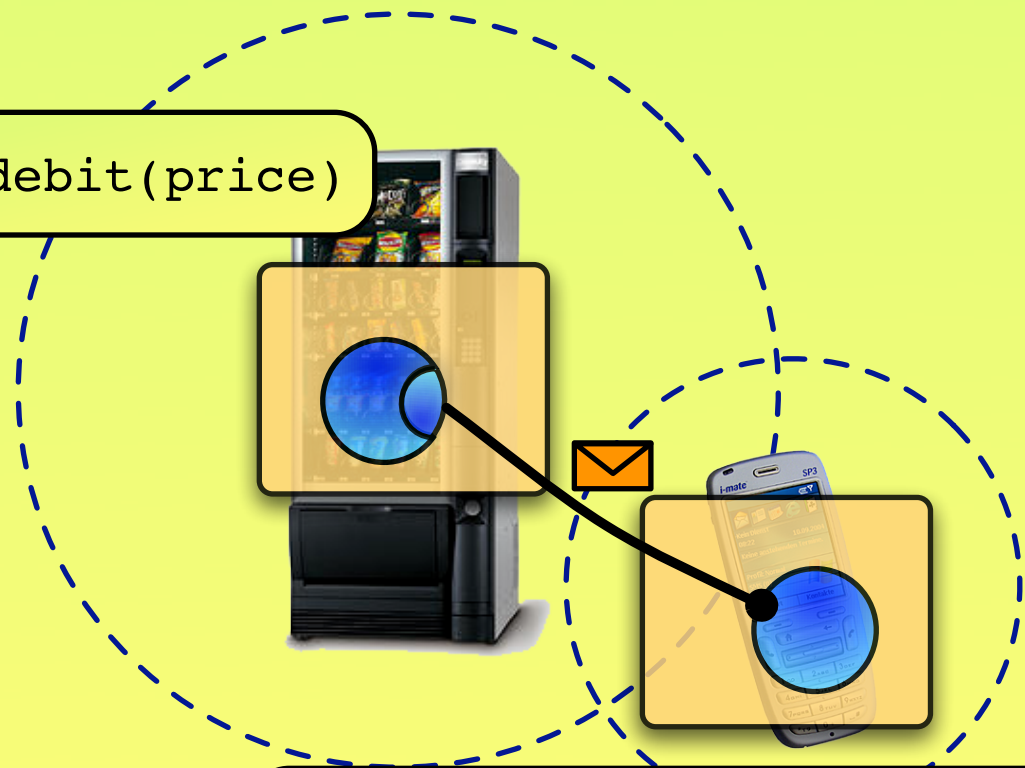


Mobile Networks

Pervasive/Ubiquitous Computing



```
cellPhone<-debit(price)
```



```
vendingMachine<-getProduct(id)
```

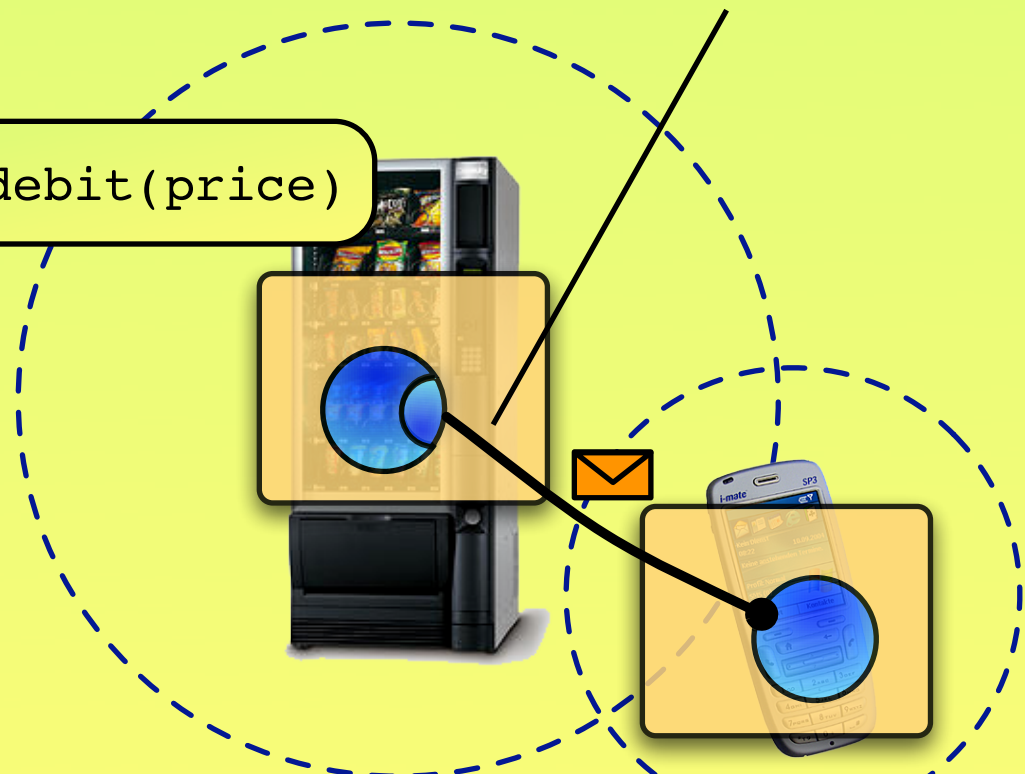
Mobile Networks

Pervasive/Ubiquitous Computing

Ambient References “remote actor references”



```
cellPhone<-debit(price)
```



```
vendingMachine<-getProduct(id)
```

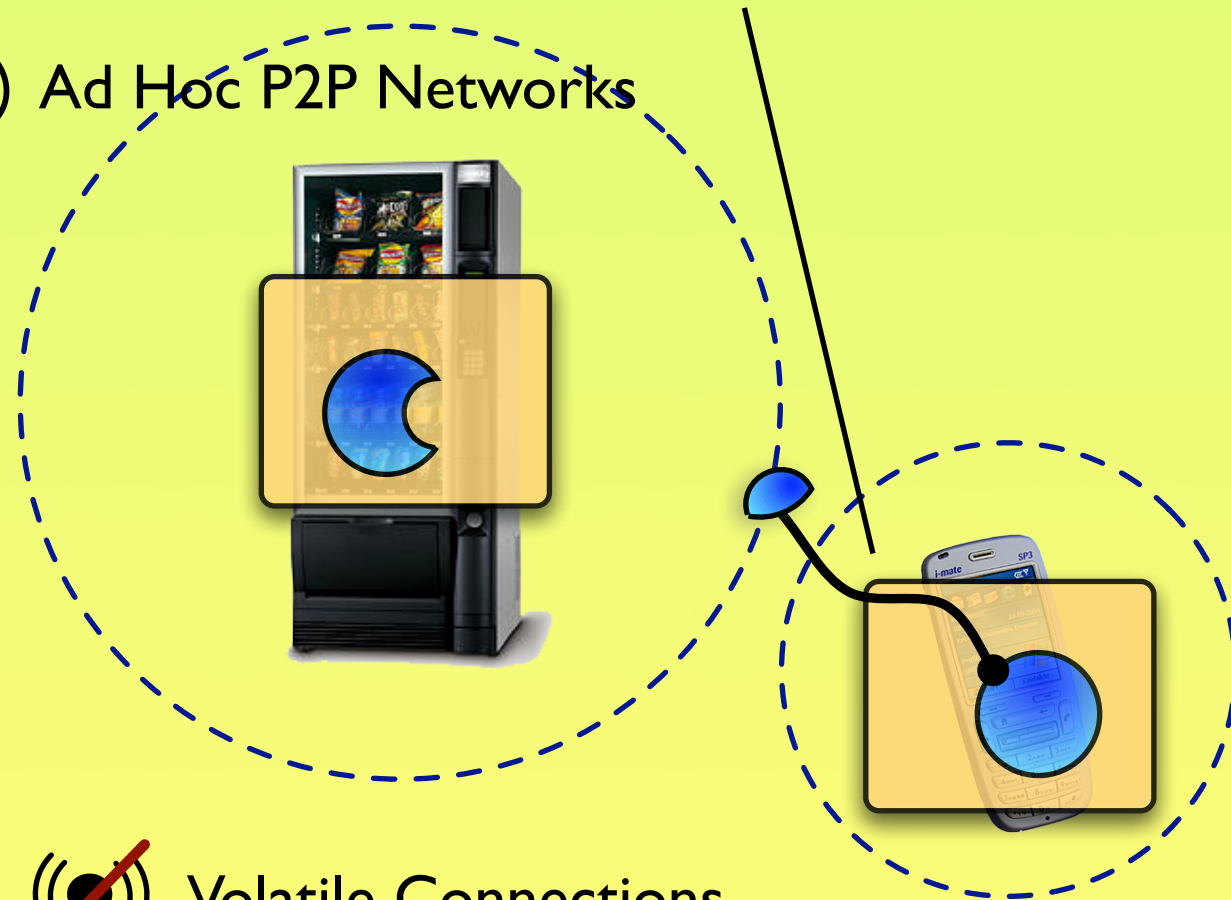
Mobile Networks

Pervasive/Ubiquitous Computing



Ambient References “remote actor references”

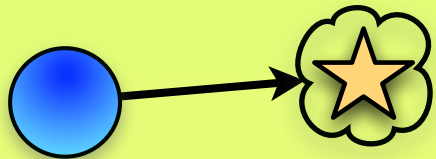
●)) Ad Hoc P2P Networks



~~(())~~ Volatile Connections

Provisionality

Service description



?

remote reference



Provisionality

Resilience

Trans.Addressing

Group Comm.

p2p discovery?

- concurrency control
- callbacks partition code
- managing disconnections

```
...  
Discovery.search(serviceDescription,  
    new DiscoveryListener() {  
        void foundService(Service s) {  
            // use the service  
        }  
        void lostService(Service s) {  
            // manage disconnection  
        }  
    }  
});
```

Resilience

- Temporary disconnections
 - should not break a remote reference
 - should not immediately raise exceptions
- Communication should resume upon reconnection

Provisionality

Resilience

Trans.Addressing

Group Comm.



Resilience

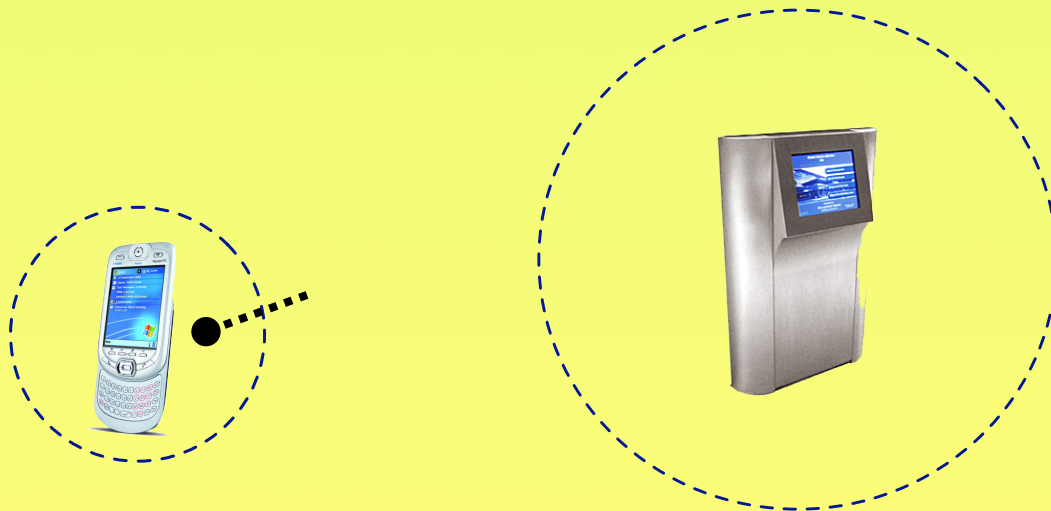
- Temporary disconnections
 - should not break a remote reference
 - should not immediately raise exceptions
- Communication should resume upon reconnection

Provisionality

Resilience

Trans.Addressing

Group Comm.



Resilience

- Temporary disconnections
 - should not break a remote reference
 - should not immediately raise exceptions
- Communication should resume upon reconnection

Provisionality

Resilience

Trans.Addressing

Group Comm.



Transitory Addressing

- Remote references: UID-based, often device-dependent
- Too inflexible: cannot rebind

Provisionality

Resilience

Trans.Addressing

Group Comm.



Transitory Addressing

- Remote references: UID-based, often device-dependent
- Too inflexible: cannot rebind

Provisionality

Resilience

Trans.Addressing

Group Comm.



Transitory Addressing

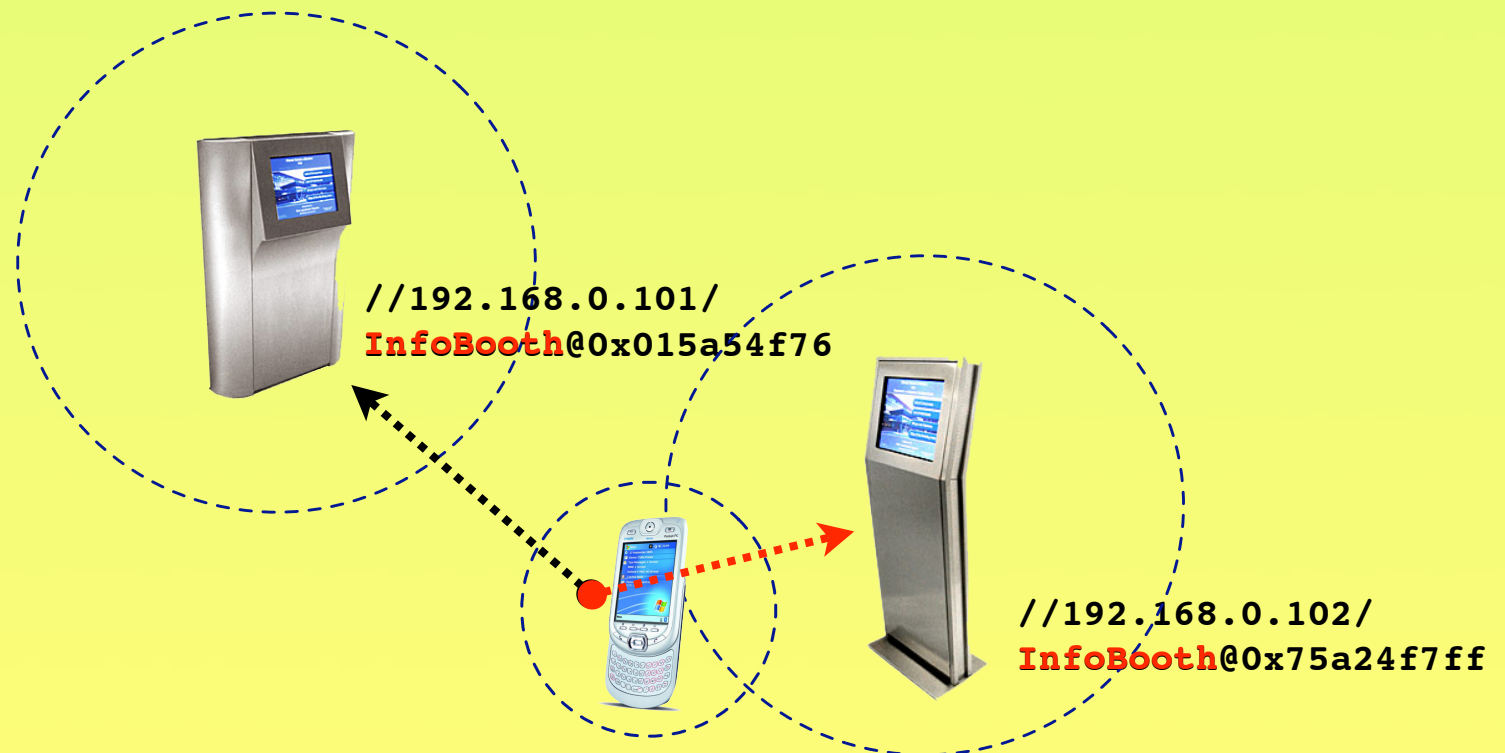
- Remote references: UID-based, often device-dependent
- Too inflexible: cannot rebind

Provisionality

Resilience

Trans.Addressing

Group Comm.



Group Communication

- Abstract from multitude of devices
- Ad hoc 'proximate' groups

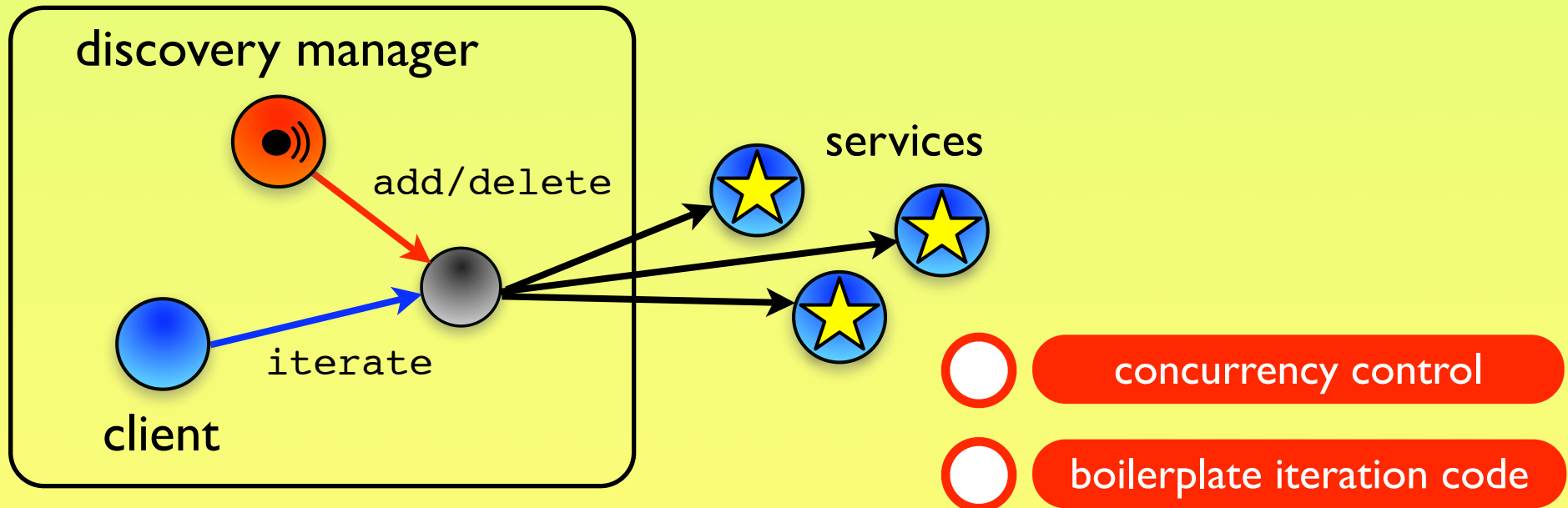
Provisionality

Resilience

Trans.Addressing

Group Comm.

collections?



Problem Statement

Provisionality

Resilience

Transitory
Addressing

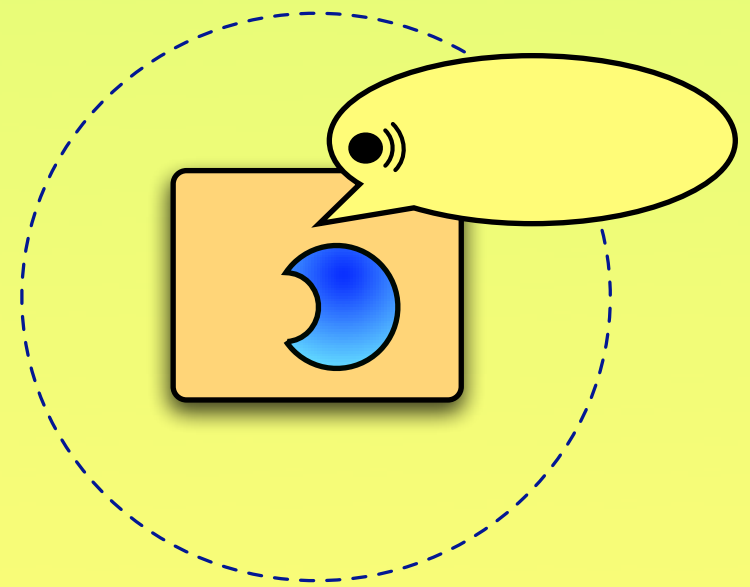
Group
Communication

- Standard remote object references fail to meet these requirements
- Need for **dedicated** referencing abstractions for mobile networks

Computational Model

- Services are ‘public’ **actors** advertising themselves via *service types*

```
deftype InstantMessenger;  
export: object as: InstantMessenger;
```

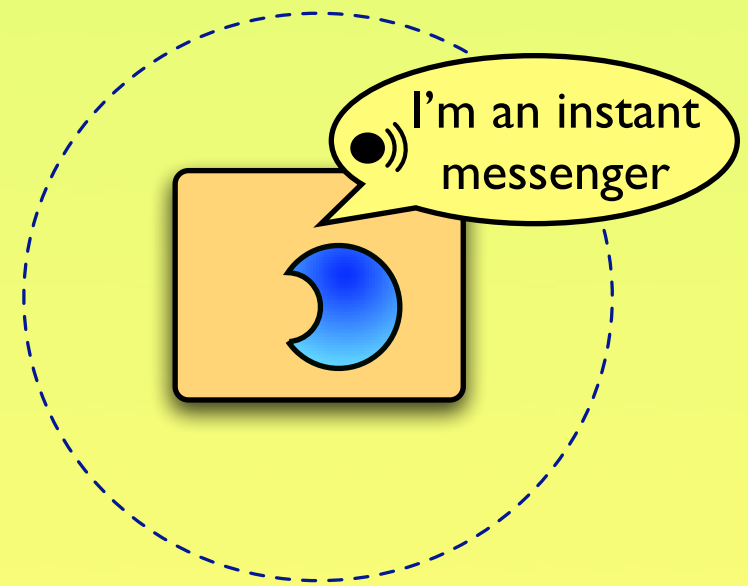


Computational Model

- Services are ‘public’ **actors** advertising themselves via *service types*

```
deftype InstantMessenger;
```

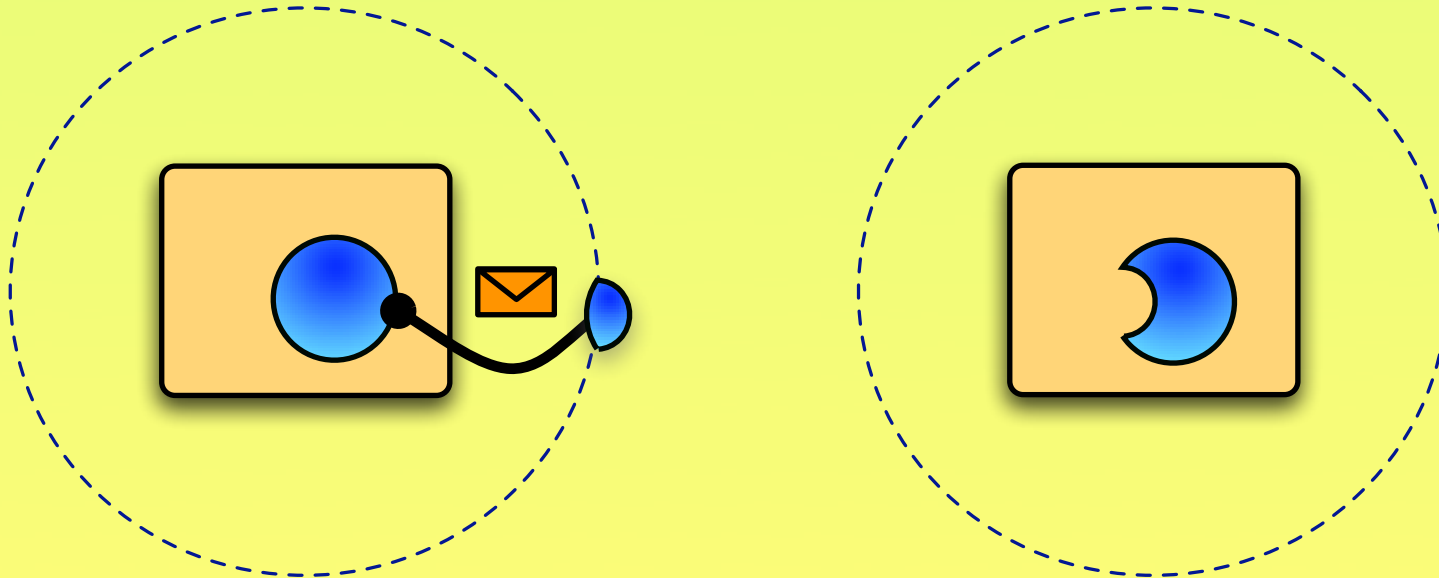
```
export: object as: InstantMessenger;
```



Ambient References

- Two states: **bound** or **unbound**
- Binds to proximate matching services

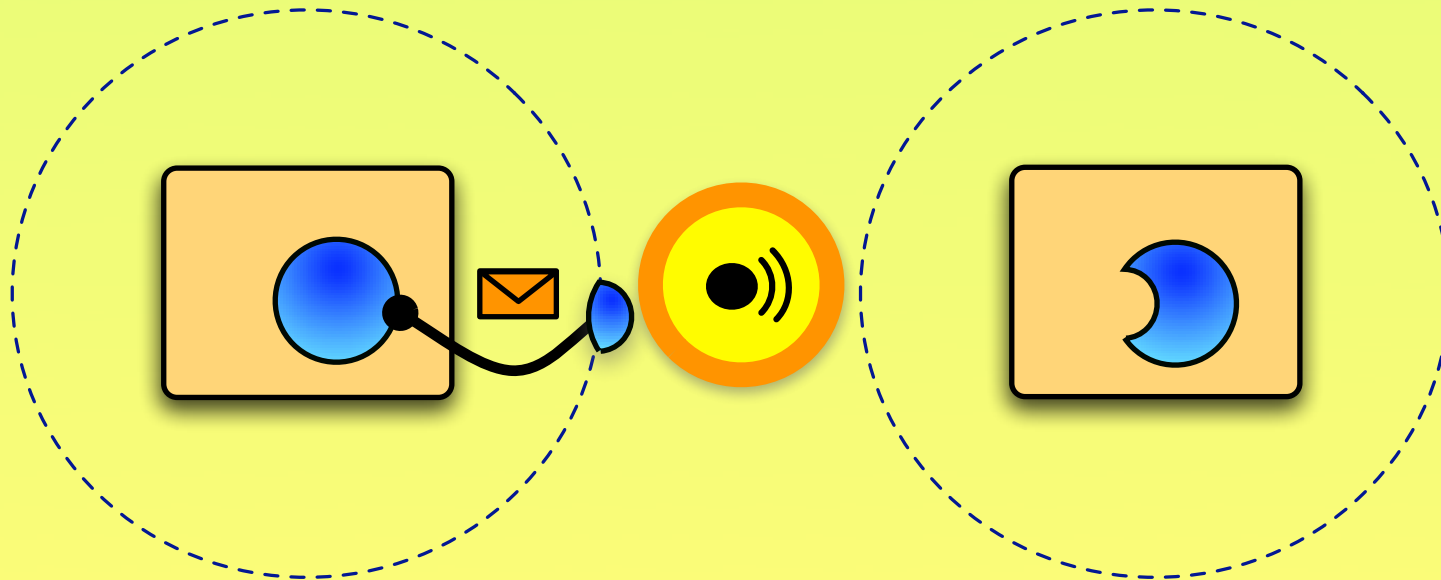
```
def aMessenger := ambient: InstantMessenger;  
aMessenger<-talk("Hello")
```



Ambient References

- Two states: **bound** or **unbound**
- Binds to proximate matching services

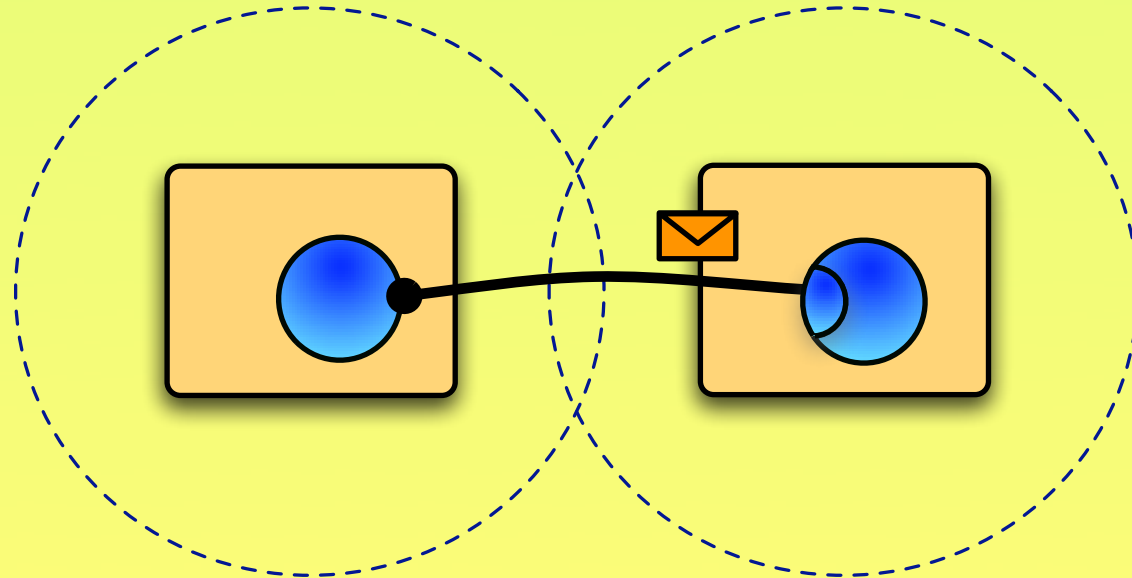
```
def aMessenger := ambient: InstantMessenger;  
aMessenger<-talk("Hello")
```



Ambient References

- Two states: **bound** or **unbound**
- Binds to proximate matching services

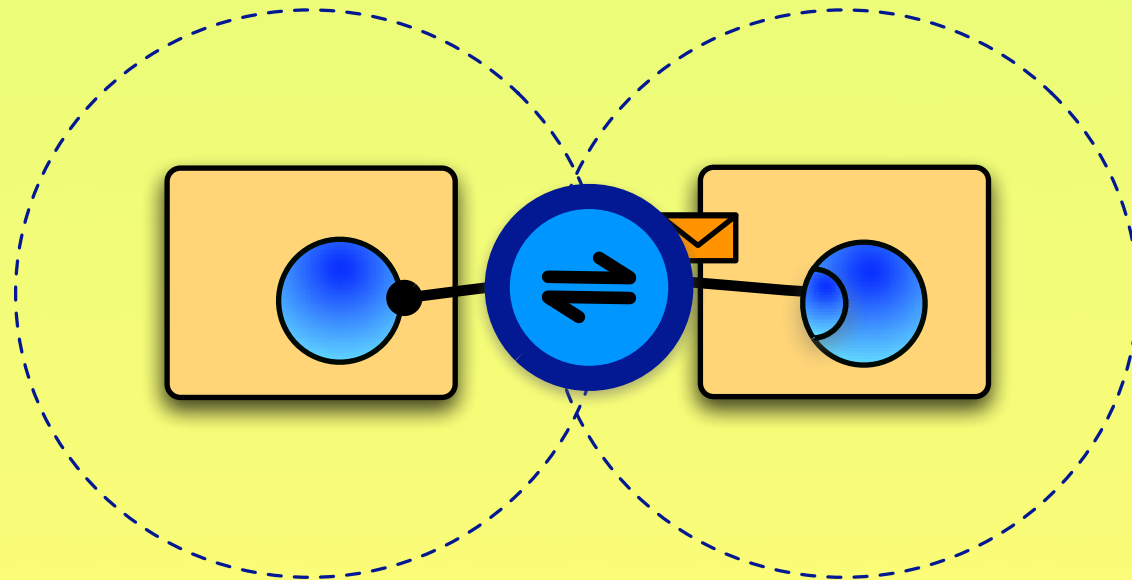
```
def aMessenger := ambient: InstantMessenger;  
aMessenger<-talk("Hello")
```



Ambient References

- Two states: **bound** or **unbound**
- Binds to proximate matching services

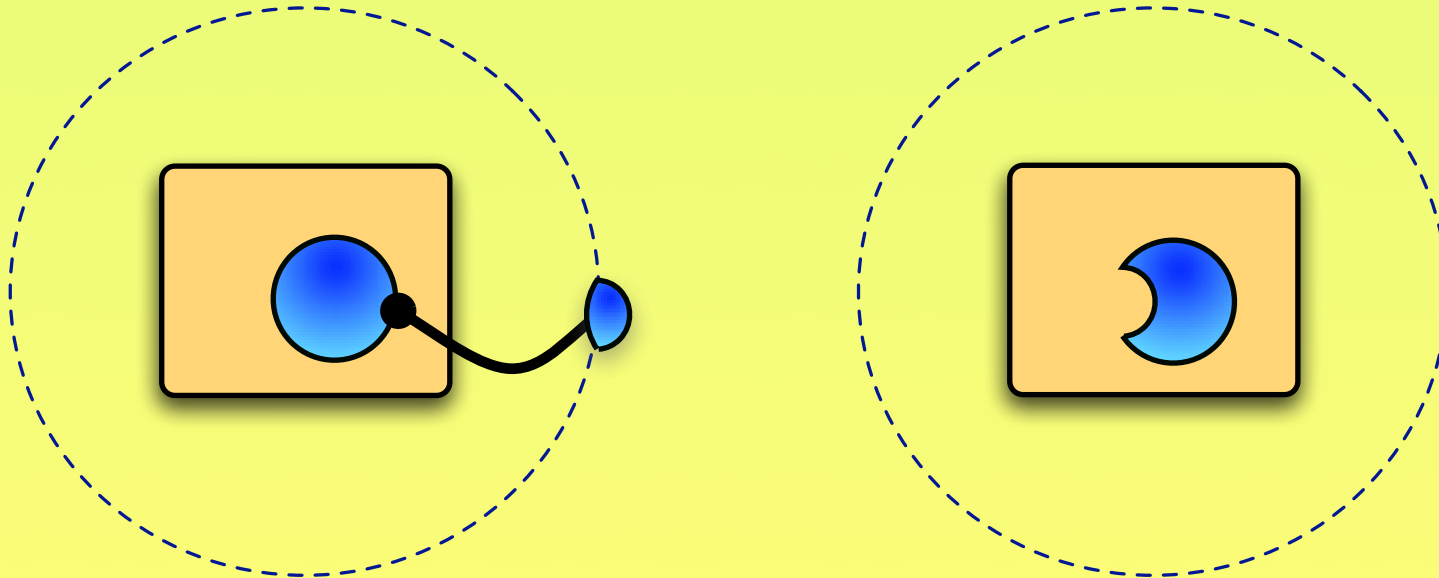
```
def aMessenger := ambient: InstantMessenger;  
aMessenger<-talk("Hello")
```



Ambient References

- Two states: **bound** or **unbound**
- Binds to proximate matching services

```
def aMessenger := ambient: InstantMessenger;  
aMessenger<-talk("Hello")
```



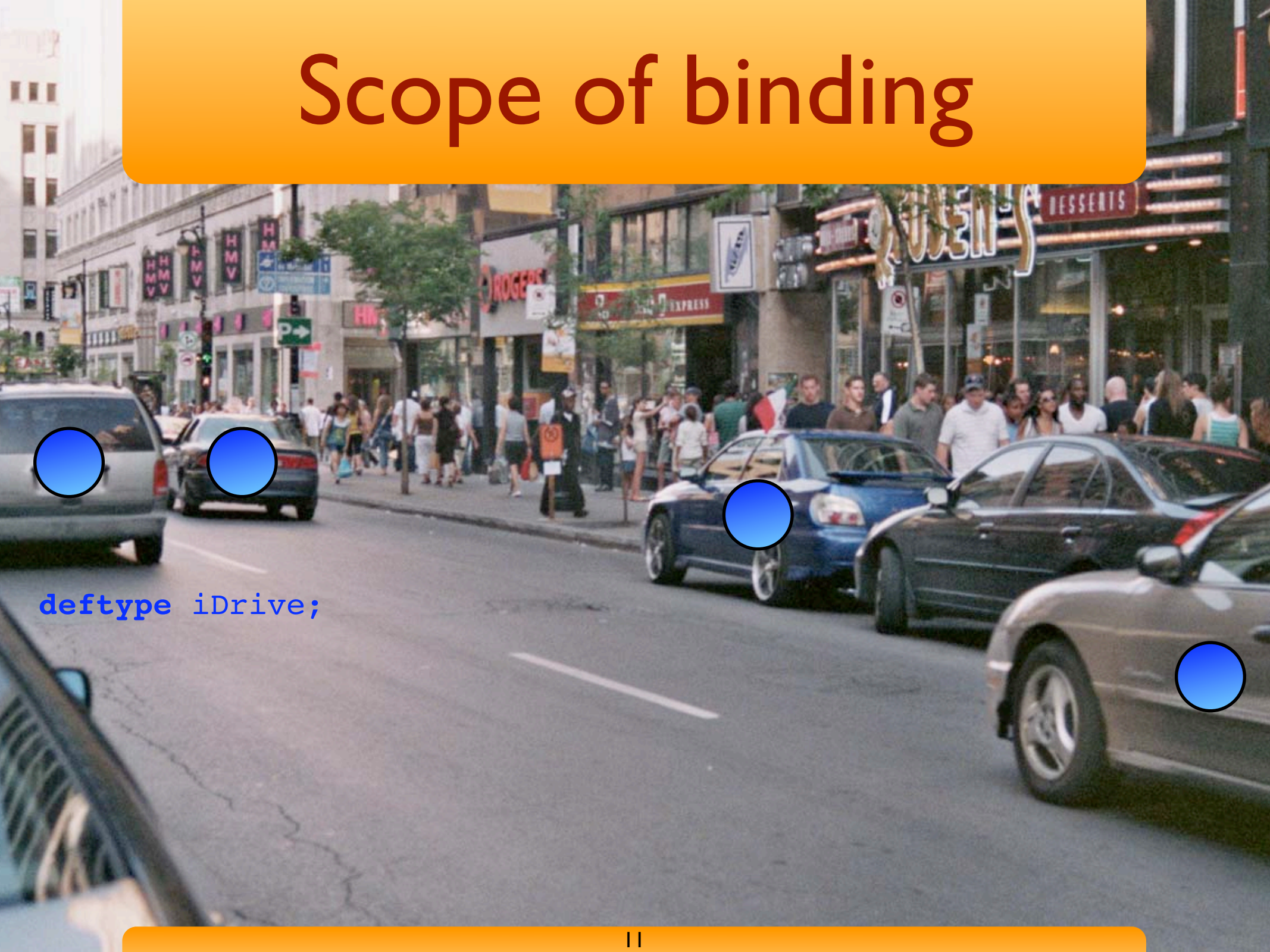
Design Dimensions

- Design **family** of remote references
- each suitable for different kind of collaboration
- Three properties:
 - Scope of binding
 - Elasticity
 - Cardinality

Scope of binding



Scope of binding



```
deftype iDrive;
```

Scope of binding

```
deftype iSign;
```

```
deftype iDrive;
```

Scope of binding

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```

Scope of binding

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```

```
deftype iAm;
```

Scope of binding

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```

```
deftype iAm;
```

```
aCar = ambient iDrive;
```

Scope of binding

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```


```
deftype iAm;
```

```
aShop:= ambient: iSell;
```

Scope of binding

```
deftype iSell;
```

```
deftype iSign;
```



```
name = ...;  
forSale = ...;  
discount = ...;
```

```
deftype iDrive;
```


```
deftype iAm;
```

```
aShop:= ambient: iSell;
```

Scope of binding

```
deftype iSell;
```

```
deftype iSign;
```



```
name = ...;  
forSale = ...;  
discount = ...;
```

```
deftype iDrive;
```

```
deftype iAm;
```

```
aShop:= ambient: iSell where: { |s|  
    s.forSale.includes("gizmo") }
```


Elasticity

`deftype iSell;`

`deftype iSign;`

`deftype iAm;`

`deftype iDrive;`

Fragile

Elastic

Sturdy

Elasticity

`deftype iSell;`

`deftype iSign;`

`deftype iDrive;`

`deftype iAm;`

Fragile

Elastic

Sturdy

Elasticity

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```

```
deftype iAm;
```

```
customer := ambient(1 day) iAm;
```

Fragile

Elastic

Sturdy

Elasticity

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```

```
deftype iAm;
```

```
favoriteShop := ambient! iSell s  
  where: { |s| s.name = "..."
```

Fragile

Elastic

Sturdy

Cardinality

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```

```
deftype iAm;
```

Uni

Multi

Omni

Cardinality

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```

```
deftype iAm;
```

Uni

Multi

Omni

Cardinality

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```

```
deftype iAm;
```

```
nearbyCars := ambient[2] iDrive;
```

Uni

Multi

Omni

Cardinality

```
deftype iSell;
```

```
deftype iSign;
```

```
deftype iDrive;
```

```
deftype iAm;
```

```
nearbyCars := ambient* iDrive;
```

Uni

Multi

Omni

Taxonomy

Scope of binding			
Elasticity x Cardinality	Fragile	Elastic	Sturdy
Uni	<code>ambient S;</code>	<code>ambient(e) S;</code>	<code>ambient! S;</code>
Multi	<code>ambient[n] S;</code>	<code>ambient(e)[n] S;</code>	<code>ambient![n] S;</code>
Omni	<code>ambient* S;</code>	<code>ambient(e)* S;</code>	<code>ambient!* S;</code>

Requirements revisited

Provisionality

Resilience

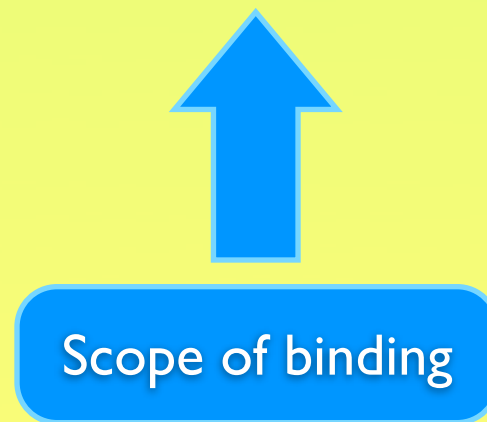
Transitory
Addressing

Group
Communication

Requirements revisited



aService = **ambient** *ServiceType*



Requirements revisited



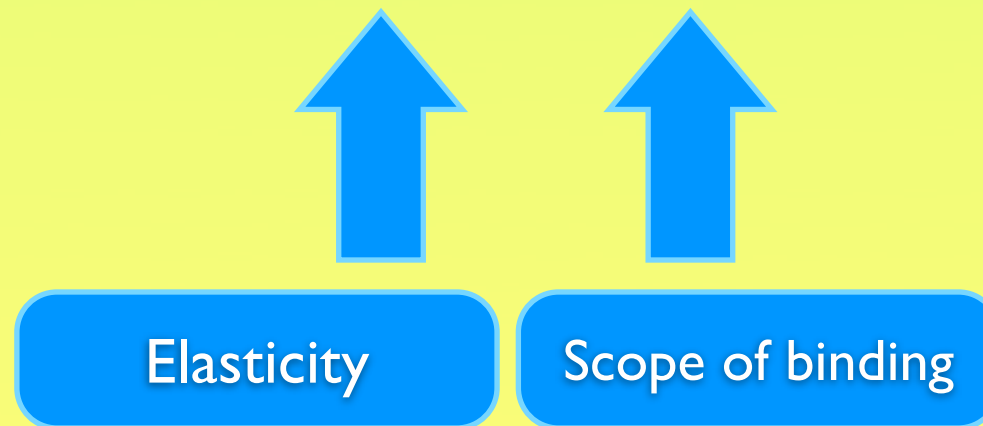
```
aService = ambient! ServiceType
```



Requirements revisited



`aService = ambient(t) ServiceType`



Requirements revisited

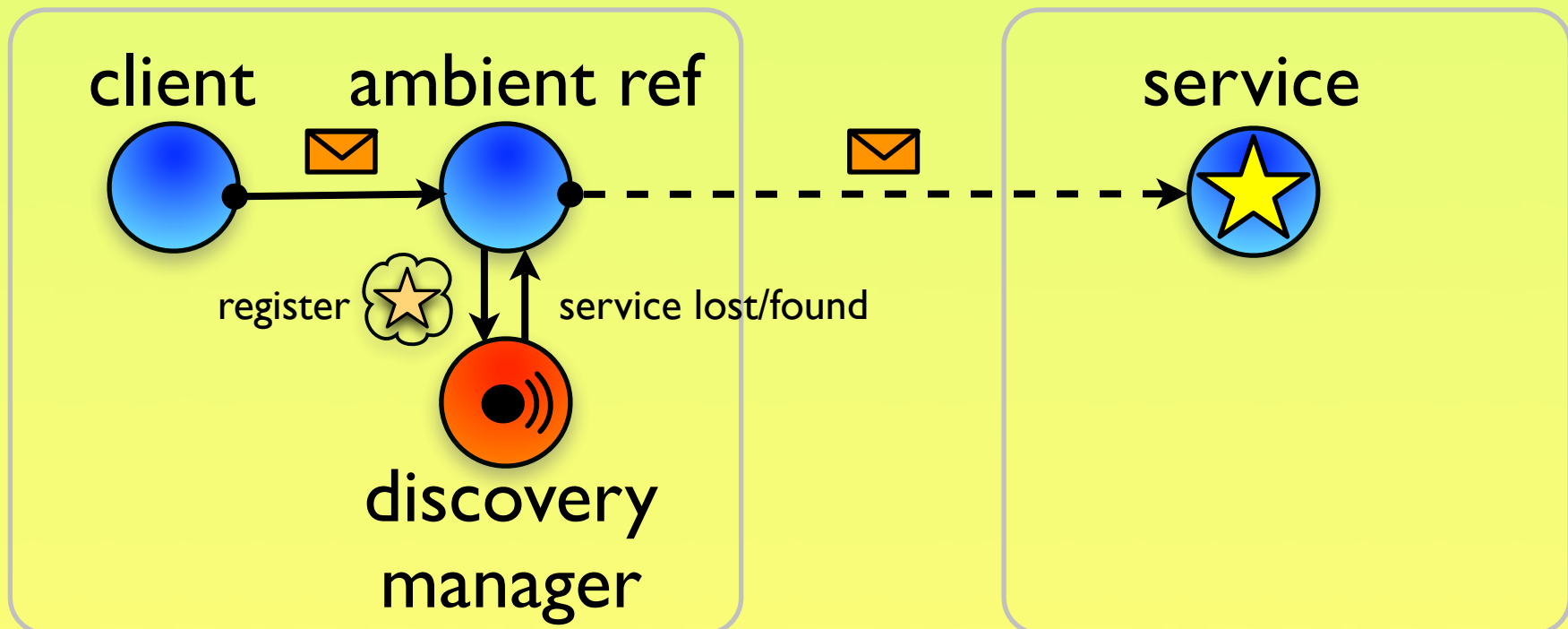


aService = **ambient*** ServiceType



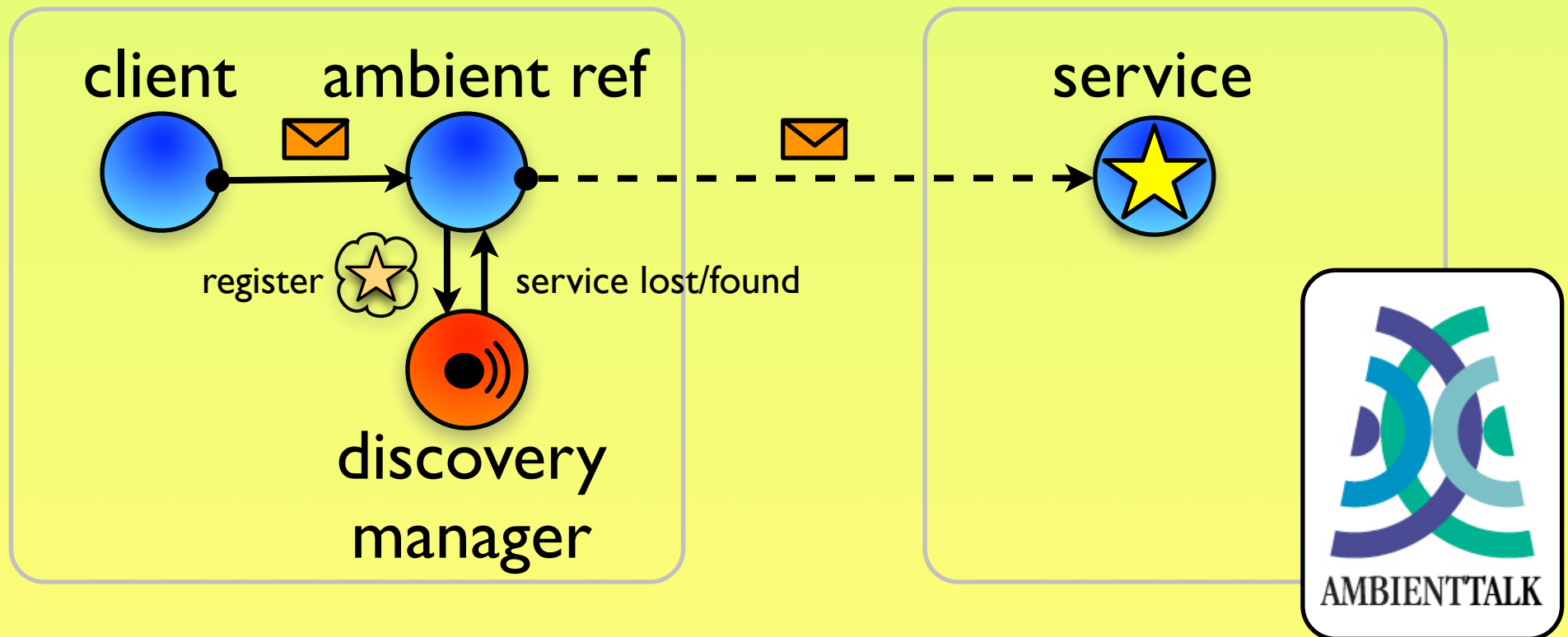
Implementation

- Local **proxy** for remote service
- Performs discovery on behalf of its client



Implementation

- Local **proxy** for remote service
- Performs discovery on behalf of its client



Conclusion

- Pervasive computing requires novel language abstractions!
- Ambient references: **remote object references** for mobile networks

