Middlewares for Mobile Environments

Giovanni Turi

Giovanni.Turi@iit.cnr.it

Istituto di Informatica e Telematica, CNR, Pisa, Italy

Agenda

1. Advisable features of a middleware for mobile environment

2. Fast (and not exhaustive) scan of current proposals from which we can learn

3. Conclusions and our research directions

Advisable features

From a survey on existing MW's for mobile distributed environments [1]:

Advisable features

From a survey on existing MW's for mobile distributed environments [1]:

- *Light-weight support*
  - Ad Hoc nodes could be resource-constrained

Advisable features

From a survey on existing MW's for mobile distributed environments [1]:

- *Light-weight support*
  - Ad Hoc nodes could be resource-constrained
- *Support for asynchronous and decoupled communication*
  - Intermittent connectivity is not an exception: data exchange has to be as flexible as possible

Advisable features

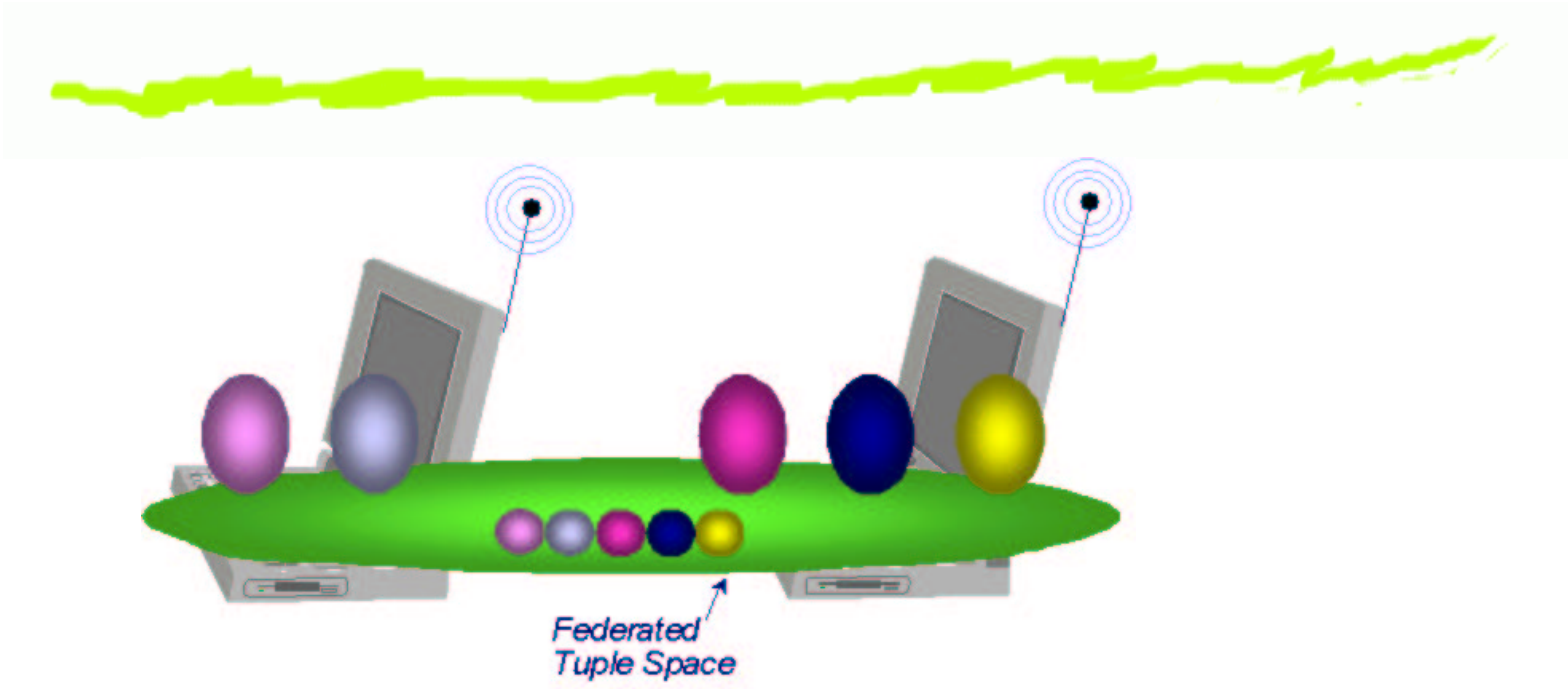
From a survey on existing MW's for mobile distributed environments [1]:

- *Light-weight support*
  - Ad Hoc nodes could be resource-constrained
- *Support for asynchronous and decoupled communication*
  - Intermittent connectivity is not an exception: data exchange has to be as flexible as possible
- *Context Awareness*
  - The MW cannot be programmed to handle context alone, as mobile environments are dynamic and the context is not *visible* at programming time

Lime: Linda in a mobile environment

- Mobile agents coordinate via global virtual data structures called *Federated tuple spaces*
- The content is distributed (without replications) among the mobile units and logically partitioned according to their connectivity
- Primitives are provided for operating on tuple spaces and react to their changes:
  - LINDA primitives enriched with location parameters *out(l,t)*, *in(w,l,p)*, *read(w,l,p)*
  - Reaction statements *react(p,a,w,l)*

Lime



- Model of decoupled communication: transient shared memory
- Reactions provide a mechanism to handle context changes
- See [2] for details

Xmiddle

- Xmiddle [3] allows applications to share XML documents through replication
- When two or more hosts are connected, they can link (i.e locally replicate) XML docs (or parts of them)
- Hosts can then work independently on the replicas until the next encounter, when a merge operation takes place
- Merging has to happen using application defined rules (context awareness)
- Similar to a distributed version controller

JXTA

- JXTA [4] is a project sponsored by Sun Microsystem with enphasis on building protocols for distributed P2P service/application
- JXTA builds a true overlay network on top of different transport protocols
- Provides primitives for grouping, and unicast/multicast communication through UNIX like pipes, together with a set basic services built on top:
  - service and group discovery
  - file searching and indexing
  - PKI...

Conclusions (1)

- Major proposals support data sharing introducing interesting semantics (see Lime), but there are no guidelines and evaluations on how that could be efficiently done for an Ad Hoc network
  - For example, Lime *in(p)* specifies a pattern matching request on the whole Federated Tuple Space: how is this request routed around?

$\Rightarrow$ Relax the semantic and focus on the kernel problem: how to work out *unbounded* data queries on Ad Hoc netwoks...

Conclusions (2)

- Things like JXTA are too heavy, but are inspiring from an architectural point of view
  - $\Rightarrow$ Start from a minimal set of services and primitives: service discovery, primitives for grouping and group communication ...
  - $\Rightarrow$ Insert *context-awareness* and *cooperation incentives* from the design phase

$\Rightarrow$ ***To get more details...***

References

[1] C. Mascolo, L. Capra, and W. Emmerich. Middleware for Mobile Computing (A Survey). *Networking 2002 Tutorial Papers*.

[2] http://lime.sourceforge.net/

[3] http://pizza.cs.ucl.ac.uk/xmiddle/

[4] http://www.jxta.org