PeerSpaces: Data-Driven Coordination on P2P Networks (2003)

From Nadia Busi, Cristian Manfredini, Alberto Montresor and Gianluigi Zavattaro of University of Bologna

Presented by Samuele Milani

Summary

- New coordination model (The Peer Spaces Model)
- Exploit the Linda Model for Peer-to-Peer Networks.
- Prototypical implementation (JXTA).
Overview

1) Peer-to-Peer Networks
2) The Linda Model
3) Linda in a Mobile Environment (LIME)
4) Peer Spaces
5) An Implementation with JXTA

1. Peer-to-Peer Networks

Peer-to-Peer Basics

- Various type of Networks:
  - Client-Server: Web servers
  - Hybrid P2P: Napster, eDonkey
  - Pure P2P: Gnutella, Freenet
2. The Linda Model
The Linda Model

- Developed at Yale University in 1985
- **Shared memory** accessible by all processes
- Data elements consist of **tuples**
- Tuple is like a record ex. (`worker`, `0`, `i`)
- Four basic primitives
- Implemented through libraries (Linda for C, Fortran, C++)

The Fantastic 4

- `out('test', 4, 64)` places a data tuple in tuple space
- `eval('test', i, f(i))` creates a live tuple, which start a process that evaluate each argument
- `rd('test', ?i, ?j)` blocking
  - read the values in a tuple in a tuple space
- `in('test', ?i, ?j)` blocking
  - same as `rd` except that it also removes the tuple from the tuple space

Example
3. Lime
(Linda in Mobile Environment)

- Problems with Linda:
  - where I store the tuples (Server)
  - rather statical model
  - not suitable for dynamical changing networks
- Lime designed for mobile applications over wired and ad hoc networks
- Supports physical and logical mobility of the hosts
- Introduces transiently shared data space
- Decoupled and opportunistic style of computation
4. Peer Spaces

- Adapt the Linda/Lime Model to P2P
- Generative communication
- Must support extreme dynamism in structure, content and load
- Large degree of self-configuration and self management is required
- New coordination model PeerSpaces
- Formal rules, abstraction of data and network structure
## Definitions

- **Data** (d)
  - Local data \( d \)
  - Generic data \( d_i \) (Chat)
  - Replicable data \( d_i \) (FileSharing)
- **Peer** \( p \) \([P, DS]\)
- **PS** set of peers
- **MD** Misplaced data
- **Network topology** \( \triangleright \triangleleft \)

## A Peer Space Example

**Peer** \( p \):

\[
p(\text{write}(job, Gen) \cdot \text{read}(response, 8) \cdot \text{write}(cash, Gen), P, \{\}) \]

**Peer** \( q \):

\[
q(\text{read}(job, 8) \cdot \text{write}(response, Rep) \cdot \text{read}(cash, 8) \cdot \\
\text{write}(pay, Gen), Q, \{dog\})
\]

**Peer** \( r \):

\[
r(\text{read}(pay, 8) \cdot \text{write}(receipt, Gen), R, \{credit\})
\]

**Network:**
- fully connected
- **MD** = \( \{\} \)
A Peer Space Example

Peer p:
p(read(response, 8). write(cash, Gen). P, {joby})

Peer q:
q(read(job, 8). write(response, Rep). read(cash, 8).
   write(pay, Gen). Q, {dog})

Peer r:
r(read(pay, 8). write(receipt, Gen). R, {credit})

Network:
- fully connected
MD = {}
A Peer Space Example

Peer p:
\( p(\text{read} (\text{response}, \text{B}), \text{write} (\text{cash}, \text{Gen}), \text{P}, \{\}) ) \)

Peer q:
\( q(\text{read} (\text{cash}, \text{B}). \text{write} (\text{pay}, \text{Gen}). \text{Q}, \{\text{dox, Job, response}\}) ) \)

Peer r:
\( r(\text{read} (\text{pay}, \text{B}). \text{write} (\text{receipt}, \text{Gen}). \text{R}, \{\text{credit}\}) ) \)

Network:
- fully connected
MD = \{ \}

A Peer Space Example

Peer p:
\( p(\text{write} (\text{cash}, \text{Gen}). \text{P}, \{\text{response}_{p}\}) ) \)

Peer q:
\( q(\text{read} (\text{cash}, \text{B}). \text{write} (\text{pay}, \text{Gen}). \text{Q}, \{\text{dox, Job, response}\}) ) \)

Peer r:
\( r(\text{read} (\text{pay}, \text{B}). \text{write} (\text{receipt}, \text{Gen}). \text{R}, \{\text{credit}\}) ) \)

Network:
- fully connected
MD = \{ \}

A Peer Space Example

Peer p:
\( p(P, \{\text{response, cash}\}) \)

Peer q:
\( q(\text{read} (\text{cash}, \text{B}). \text{write} (\text{pay}, \text{Gen}). \text{Q}, \{\text{dox, Job, response}\}) ) \)

Peer r:
\( r(\text{read} (\text{pay}, \text{B}). \text{write} (\text{receipt}, \text{Gen}). \text{R}, \{\text{credit}\}) ) \)

Network:
- fully connected
MD = \{ \}
A Peer Space Example

Peer p:
p(P, \{\text{response}\})

Peer q:
q(\text{write(pay, Gen), Q, (dog, job, response, cash)})

Peer r:
r(\text{read(pay, R). write(receipt, Gen), R, \{credit\}})

Network:
- fully connected

\[ MD = \{ \} \]
### Data Production

1. \( \text{write}(\text{Fun}, P, DS \oplus P, w, MD) \rightarrow (P, DS \oplus w, MD) \)
2. \( \text{write}(\text{Fun}, P, DS \oplus P, w, MD) \rightarrow (P, DS \oplus w, MD) \)
3. \( \text{write}(\text{Fun}, P, DS \oplus P, w, MD) \rightarrow (P, DS \oplus w, MD) \)
4. \( \text{write}(\text{Fun}, P, DS \oplus P, w, MD) \rightarrow (P, DS \oplus w, MD) \)
5. \( \text{read}(\text{Fun}, P, DS \oplus w, MD) \rightarrow \text{Read}(p, p') \)

### Data Retrieval

1. \( \text{read}(\text{Fun}, P, DS \oplus w, MD) \rightarrow \)
2. \( \text{read}(\text{Fun}, P, DS \oplus w, MD) \rightarrow \)
3. \( \text{read}(\text{Fun}, P, DS \oplus w, MD) \rightarrow \)
4. \( \text{read}(\text{Fun}, P, DS \oplus w, MD) \rightarrow \)
5. \( \text{read}(\text{Fun}, P, DS \oplus w, MD) \rightarrow \)

### Constant and Context Rules

1. \( (P, w, MD) \rightarrow (P, w', MD) \)
2. \( (P, w', MD) \rightarrow (P, w', MD) \)
3. \( (P, w', MD) \rightarrow (P, w', MD) \)
4. \( (P, w', MD) \rightarrow (P, w', MD) \)
5. \( (P, w', MD) \rightarrow (P, w', MD) \)

4 Open issues?

- Implementation of:
  - Route \((p, p')\)
  - Hot \((p, h)\)
  - LoadBal \((p, p'; d)\)

5. A JXTA Prototypical Implementation

What’s JXTA?

- JXTA is an open-source project by Sun to provide a set of basic facilities for P2P applications
- Based on a set of XML protocols
- JXTA middleware has 3 layers
  - Core: low-level functions (routing, communication, ...)
  - Services: indexing, searching, file sharing
  - Applications: high-level application (chat, auction, ...)
Conclusion

- We can not yet draw conclusions until concrete implementations come out!!!
- But the model should theoretically work

To be continued...