## Z25 Adaptive and Mobile Systems
## Dr. Cecilia Mascolo

# XMIDDLE: A Data-Sharing Middleware for Mobile Computing

C. Mascolo, L. Capra, S. Zachariadis and W. Emmerich

University College London

(presented by Mirco Musolesi)

## Motivation of the Paper

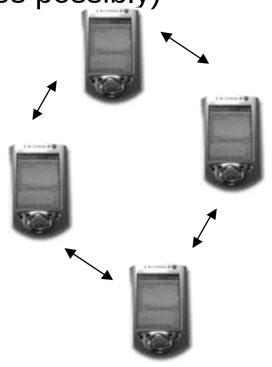
- Middleware to help data sharing and reconciliation in mobile ad hoc networks
- People can work offline on copies of data
- When online again they can synchronize their data
- No central server, all decentralized

## Mobile scenario

- Challenging problems:
  - weak connectivity
    - limited bandwidth/high error rate
    - possible and unexpected disconnections
  - scarce resources
    - computational capabilities
    - limited memory resources
    - limited battery
    - constrained user interface
  - Completely decentralized setting (possibly no central server)

## Scenario

- Mobile ad-hoc networks (different interfaces possibly)
- WLAN 802.11
- Bluetooth (WPAN 802.15)
- Additional issues
- Lack of infrastructure
- Client/server model not suitable



## Middleware

- Traditional middleware systems used to hide the complexity related to heterogeneity and distribution of resources and software components
- Traditional systems designed for fixed networks:
  - permanent connectivity
  - disconnections are considered exceptions
- Are they suitable for a mobile context?

## Mobile Middleware

- Possible approaches:
  - adaptation of the existing traditional middleware systems (Mobiware, Rover Toolkit, ALICE)
  - middleware systems expressly designed for mobile environment (Bayou, Odissey, Lime)

## XMIDDLE

- Specifically designed for mobile ad-hoc networks scenario
- It provides a framework to develop applications that need data-sharing
- It supports disconnected operations by means of data replication.

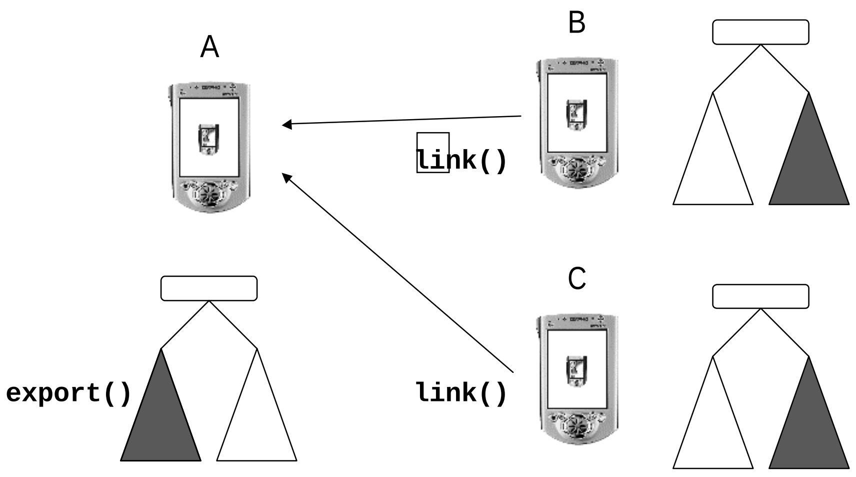
## XMIDDLE

- It provides a support for automatic data reconciliation after changes performed during disconnections.
- Managed data structures are XML documents that can be semantically associated with trees.

## XMIDDLE architecture

| | |
|---|---|
| **Mobile applications** | Application layer |
| **XMIDDLE** | Presentation layer |
| | Session layer |
| **UDP** | Transport layer |
| **IP** | Network layer |
| **IEEE802.11 data link layer** | Data link layer |
| **IEEE802.11 physical layer** | Physical layer |

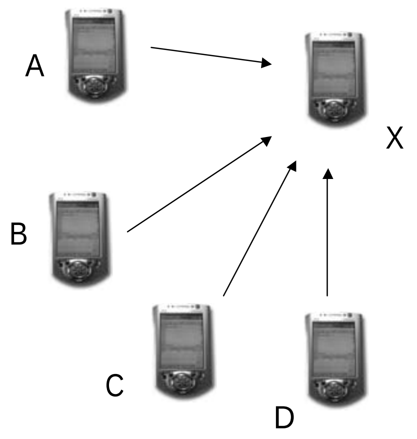
## Data replication

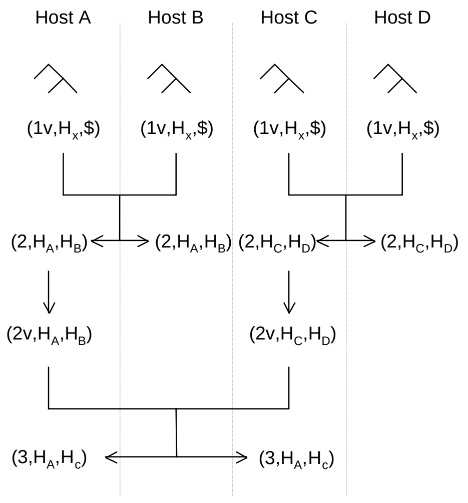


## Versions and editions

- In order to provide data replication and reconciliation XMIDDLE uses a versioning system.
- The exported document has both versions and editions:
- Versions contain changes that are performed locally (without communicating them to other hosts, for example performed during a disconnection).
- Editions are “stable versions”: they are released after a reconciliation process.

## Versions and editions





## Versions and editions

- The reconciliation process of divergent data replicas is based on this versioning system.
- The latest common edition is used to detect the changes performed by the two hosts on their local copy after the last reconciliation.

## Reconciliation protocol

- Executed by hosts that share the same part of the tree:
- after changes performed on the shared data structure (if the hosts are connected and in reach)
- after a reconnection (explicit or implicit)
- Point-to-point protocol

## Reconciliation algorithm

- Use of *XML tree diff and merge techniques* to find differences between XML documents
- The remote divergent replica is reconstructed locally by using these techniques.

## Semantic problems in conflict resolution

- The challenging problem is not finding the conflicts, but solving them
- If conflict detection is a *syntactic problem* (i.e. finding differences between two XML documents)…
- …conflict resolution is typically a *semantics problem*.

## A general design principle

- It is possible to identify a general principle related to the design of middleware platforms:

  *Middleware has to provide mechanisms without implementing any particular policies.*
- How is it possible to introduce semantic information in middleware systems design?

## Semantic problems in conflict resolution

- We use metadata in order to distinguish the two aspects of middleware:
  - the mechanisms that it implements
  - the policies according which it works
- We exploit this approach using XML technologies: XML, DOM and XML Schema to specify:
  - tree topology
  - *application-dependent* conflict resolution

## A simple example

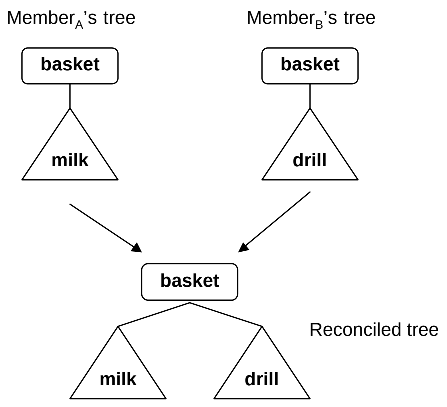
```
<?xml version=”1.0”?>
<basket order=”no”>
     <item>
           <name>milk</name>
           <quantity resolutor=“add”>1</quantity>
           <price>1.2</price>
     </item>
</basket>

<?xml version=”1.0”?>
<basket order=”no”>
     <item>
           <name>drill</name>
           <quantity resolutor=“add”>1</quantity>
           <price>60</price>
     </item>
</basket>
```

## A simple example

```
<?xml version=”1.0”?>
<basket order=”no”>
     <item>
            <name>milk</name>
            <quantity resolutor=“add”>1</quantity>
            <price>1.2</price>
     </item>
     <item>
            <name>drill</name>
            <quantity resolutor=“add”>1</quantity>
            <price>60</price>
     </item>
</basket>
```

## A simple example

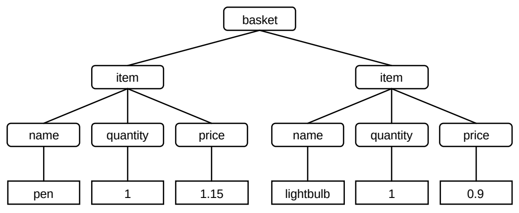


## A simple example

```
 <xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
 <xsd:element name="basket">
  <xsd:complexType>
   <xsd:element name="item" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
    <xsd:complexType>
     <xsd:element name="name" type="xsd:string"/>
     <xsd:element name="price" type="xsd:decimal"/>
     <xsd:element name="quantity">
      <xsd:complexType base="decimal">
       <xsd:attribute name="resolutor" use="default" value="greater"/>
      </xsd:complexType>
     </xsd:element>
    </xsd:complexType>
   </xsd:element>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

## Trees

- XML documents can be semantically associated with trees.



```
<?xml version="1.0"?>
<basket>
     <item>
       <name>pen</name>
       <quantity>1</quantity>
       <price>1.15</price>
     </item>
     <item>
       <name>lightbulb</name>
       <quantity>1</quantity>
       <price>0.9</price>
     </item>
</basket>
```

## Unordered tree: an example

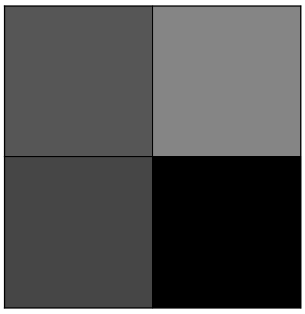
```
<?xml version="1.0"?>
<basket order="no">
  <item>
          <name>pen</name>
          <quantity>1</quantity>
          <price>1.15</price>
   </item>
   <item>
          <name>rubber</name>
          <quantity>1</quantity>
          <price>0.5</price>
   </item>
</basket>
```

$Doc_A$

```
<?xml version="1.0"?>
<basket order="no">
      <item>
          <name>rubber</name>
          <quantity>1</quantity>
          <price>0.5</price>
      </item>
      <item>
          <name>pen</name>
          <quantity>1</quantity>
          <price>1.15</price>
      </item>
</basket>
```

$Doc_B$

## Ordered tree: an example



```
<?xml version="1.0"?>
<image order="yes">
 <pixel>
   <R>255</R>
   <G>0</G>
   <B>0</B>
 </pixel>
 <pixel>
   <R>51</R>
   <G>51</G>
   <B>204</B>
 </pixel>
 <pixel>
   <R>255</R>
   <G>207</G>
   <B>1</B>
 </pixel>
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>0</B>
 </pixel>
</image>
```

## Types of data conflicts

- There are essentially two possible types of data conflicts:
  - value conflicts
  - structure conflicts
- *Value conflicts* are related to the values of text nodes
- *Structure conflicts* are related to the topology of the document

## Example of value conflicts

```
<?xml version=”1.0”?>
<basket order=”no”>
  <item>
    <name>pen</name>
    <quantity>
     1
    </quantity>
    <price>1.15</price>
  </item>
</basket>
```

Doc$_A$

```
<?xml version=”1.0”?>
<basket order=”no”>
  <item>
    <name>pen</name>
    <quantity>
     2
    </quantity>
    <price>1.15</price>
  </item>
</basket>
```

Doc$_B$

How is it possible to deal with this *value conflict*?

## Example of structure conflicts

```
<?xml version=”1.0”?>
<basket order=“no”>
 <item>
  <name>pen</name>
  <quantity>1</quantity>
  <price>1.15</price>
 </item>
 <item>
  <name>rubber</name>
  <quantity>1</quantity>
  <price>0.5</price>
 </item>
</basket>
```

```
<?xml version=”1.0”?>
<basket order=“no”>
 <item>
  <name>pen</name>
  <quantity>1</quantity>
  <price>1.15</price>
 </item>
</basket>
```

How is it possible to deal with this *structure conflict*?

## Putting all together…

```
<?xml version="1.0"?>
<basket>
 <item>
  <name>pen</name>
  <quantity>1</quantity>
  <price>1.15</price>
 </item>
 <item>
  <name>rubber</name>
  <quantity>1</quantity>
  <price>0.5</price>
 </item>
</basket>
```

$\text{Doc}_A$

```
<?xml version="1.0"?>
<basket>
 <item>
  <name>rubber</name>
  <quantity>1</quantity>
  <price>0.5</price>
 </item>
</basket>
```

$\text{Doc}_B$

How is it possible to deal with this situation?

## Conflict resolution using the latest common edition

```
<?xml version="1.0"?>
<image order="yes">
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>0</B>
 </pixel>
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>255</B>
 </pixel>
 <pixel>
   <R>255</R>
   <G>255</G>
   <B>255</B>
 </pixel>
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>0</B>
 </pixel>
</image>
```

$\text{Doc}_A$

```
<?xml version="1.0"?>
<image order="yes">
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>0</B>
 </pixel>
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>255</B>
 </pixel>
 <pixel>
   <R>255</R>
   <G>255</G>
   <B>255</B>
 </pixel>
 <pixel>
   <R>255</R>
   <G>255</G>
   <B>255</B>
 </pixel>
</image>
```

$\text{Doc}_B$

```
<?xml version="1.0"?>
<image order="yes">
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>0</B>
 </pixel>
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>255</B>
 </pixel>
 <pixel>
   <R>255</R>
   <G>255</G>
   <B>255</B>
 </pixel>
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>0</B>
 </pixel>
</image>
```

$\text{Doc}_{old}$

```
<?xml version="1.0"?>
<image order="yes">
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>0</B>
 </pixel>
 <pixel>
   <R>0</R>
   <G>0</G>
   <B>255</B>
 </pixel>
 <pixel>
   <R>255</R>
   <G>255</G>
   <B>255</B>
 </pixel>
 <pixel>
   <R>255</R>
   <G>255</G>
   <B>255</B>
 </pixel>
</image>
```

$\text{Doc}_{rec}$

## Conflict resolution using the latest common edition

```
<?xml version="1.0"?>
<basket order="no">
 <item>
  <name>pen</name>
  <quantity>1</quantity>
  <price>1.15</price>
 </item>
</basket>
```

$Doc_A$

```
<?xml version="1.0"?>
<basket order="no">
 <item>
  <name>rubber</name>
  <quantity>1</quantity>
  <price>0.5</price>
 </item>
</basket>
```

$Doc_B$

```
<?xml version="1.0"?>
<basket order="no">
</basket>
```

Latest Common Edition

## Conflict resolution using the latest common edition

```
<?xml version="1.0"?>
 <basket order="no">
   <item>
    <name>pen</name>
    <quantity>1</quantity>
    <price>1.15</price>
   </item>
   <item>
    <name>rubber</name>
    <quantity>1</quantity>
    <price>0.5</price>
   </item>
  </basket>
```

Reconciled document

→The latest common edition is an empty document.

→After the last reconciliation process user A and user B bought respectively the pen and the rubber items.

→In this case the reconciled document must be composed of two branches corresponding to rubber and pen items.

## Conflict resolution using the latest common edition

```
<?xml version="1.0"?>
<basket order="no">
 <item>
  <name>pen</name>
  <quantity>1</quantity>
  <price>1.15</price>
 </item>
 <item>
   <name>rubber</name>
   <quantity>1</quantity>
   <price>0.5</price>
  </item>
</basket>
```

Doc$_A$

```
<?xml version="1.0"?>
<basket order="no">
 <item>
  <name>pen</name>
  <quantity>1</quantity>
  <price>1.15</price>
 </item>
</basket>
```

Doc$_B$

```
<?xml version="1.0"?>
<basket order="no">
 <item>
  <name>pen</name>
  <quantity>1</quantity>
  <price>1.15</price>
 </item>
 <item>
  <name>rubber</name>
  <quantity>1</quantity>
  <price>0.5</price>
 </item>
</basket>
```

Latest Common Edition

## Conflict resolution using the latest common edition

```
<?xml version="1.0"?>
 <basket order="no">
  <item>
    <name>pen</name>
     <quantity>1</quantity>
     <price>1.15</price>
    </item>
 </basket>
```

Reconciled document

→In the latest common edition there are the `pen` and `rubber` items.

→After the execution of the last reconciliation process, user B has modified his document, deleting the `rubber` item.

→Using the latest common edition we can detect this change.

→In the reconciled document the `rubber` item must not be present.

## Application-specific conflict resolutors

- XMIDDLE also supports the definition of application-specific policies to solve value conflicts.
- Use of the `resolutor` attribute

```
<?xml version=”1.0”?>
 <basket order=”no”>
  <item>
   <name>pen</name>
   <quantity resolutor=“greater”>1</quantity>
   <price>1.15</price>
  </item>
 </basket>
```

## Application-specific conflict resolutors

```
<?xml version=”1.0”?>
<basket order=”no”>
  <item>
    <name>pen</name>
    <quantity
resolutor=“greater”>
     1
    </quantity>
    <price>1.15</price>
  </item>
</basket>
```

Doc$_A$

```
<?xml version=”1.0”?>
<basket order=”no”>
  <item>
    <name>pen</name>
    <quantity
resolutor=“greater”>
     2
    </quantity>
    <price>1.15</price>
  </item>
</basket>
```

Doc$_B$

```
<?xml version=”1.0”?>
<basket order=”no”>
  <item>
    <name>pen</name>
    <quantity
resolutor=“greater”>
     2
    </quantity>
    <price>1.15</price>
  </item>
</basket>
```

Reconciled document

→In this case we suppose that the latest common edition is the empty document.

## Application-specific conflict resolutors

- Possible application-specific conflict resolutors:
- arithmetic resolutors (add, lesser, greater)
- string resolutors
- …
- user-defined resolutors

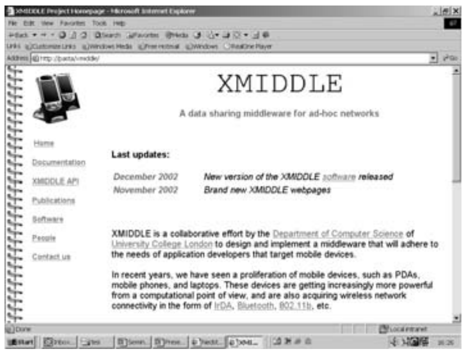
## Discussion

- Group reconciliation protocol (introduction of group policies)
- Definition of *host profiles* (adaptation?)
- Security issues:
  - data encryption
  - permissions (as in UNIX)
- Identity
- Performance

## XMIDDLE Website (sourceforge project)

XMIDDLE website (open source release):

http://xmiddle.sourceforge.net/


XMIDDLE Project Homepage - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://parta/xmiddle/
Home
Documentation
XMIDDLE API
Publications
Software
People
Contact us
Done
Local intranet

XMIDDLE

A data sharing middleware for ad-hoc networks

**Last updates:**

| | |
|---|---|
| *December 2002* | ***New version of the XMIDDLE software released*** |
| *November 2002* | ***Brand new XMIDDLE webpages*** |

XMIDDLE is a collaborative effort by the Department of Computer Science of University College London to design and implement a middleware that will adhere to the needs of application developers that target mobile devices.

In recent years, we have seen a proliferation of mobile devices, such as PDAs, mobile phones, and laptops. These devices are getting increasingly more powerful from a computational point of view, and are also acquiring wireless network connectivity in the form of IrDA, Bluetooth, 802.11b, etc.