Active XML

Serge Abiteboul, Omar Benjelloun,
Ioana Manolescu, Tova Milo,
Bernd Amann, Jerome Baumgarten,
Bogdan Cautis
And many others

Organization

1. **The context: XML and Web services**
2. Introduction
3. Active XML
4. Architecture and implementation
5. Four technical issues in brief
    a) Data exchange
    b) Lazy service calls and query optimization
    c) Distribution and replication
    d) Security and access control
6. Illustration: some applications
7. Conclusion

Information is everywhere

- Data integration
  - Mediation, warehousing or hybrid data integration
  - Web portals, enterprise knowledge, comparative shopping, procurement, business intelligence, …
- Data management for
  - cooperative work
  - ambient computing
  - mobile applications
  - Grid computing
- Digital Libraries
- Electronic something
  - E-commerce, E-government, E-procurement…
  - B2C, B2G, B2B…
- Network management

Information is accessible

Information used to live in islands but it is changing

- Step1: The Web of yesterday
  - HTTP, HTML, browsing and full-text indexing
  - Variety of formats, protocols, languages…
  - Primarily used by humans
- Step2: The Web of today
  - A standard for data with query languages
  - A standard for distribution
  - Used by humans and software applications

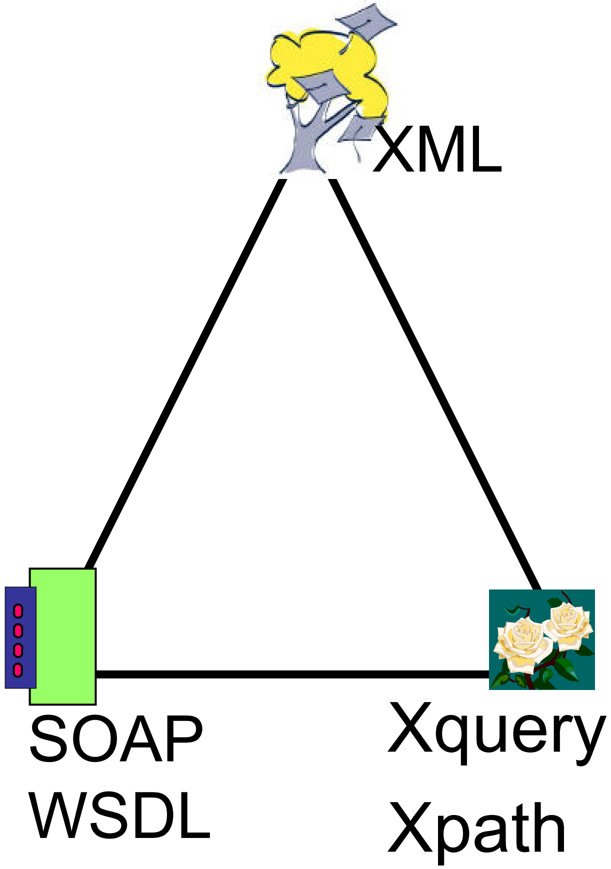
Uniform access to information…

…the dream for distributed data management

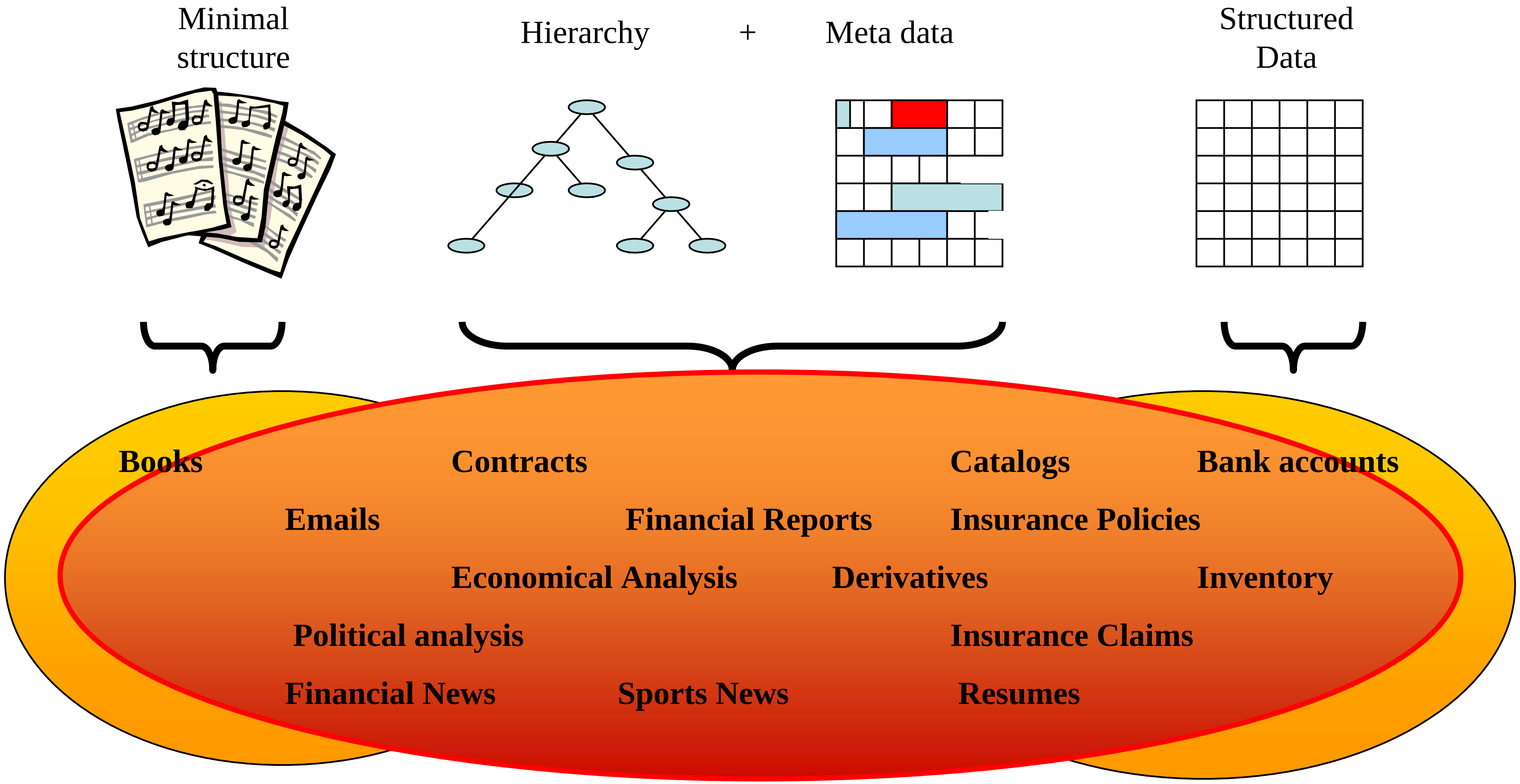
The golden triangle

of distributed information management

- Standard for data exchange
  - **XML, XML Schema…**
  - Extensible Markup Language
  - Labeled ordered trees
- Query languages
  - **XPATH, XQuery…**
- Standards for distributed computing: Web services
  - **SOAP, WSDL, UDDI…**
  - Simple Object Access Protocols



The information spectrum
XML and Semi-structured data



Minimal structure

Hierarchy + Meta data

Structured Data

Books

Contracts

Catalogs

Bank accounts

Emails

Financial Reports

Insurance Policies

Economical Analysis

Derivatives

Inventory

Political analysis

Insurance Claims

Financial News

Sports News

Resumes

What can be captured with XML?

- Very structured information
  - Databases, knowledge bases
  - Most DBMS now export in XML
- Semi-structured information
  - Data exchange formats (ASN.1, SGML), e.g., technical documentation
- Less structured data: documents
  - Structure in them: chapter, section, table of content and index
  - Tagging of elements in it (citation, special words)
  - Links to other documents
- Unstructured data such as images and sound
  - Meta-data: Author, date, status

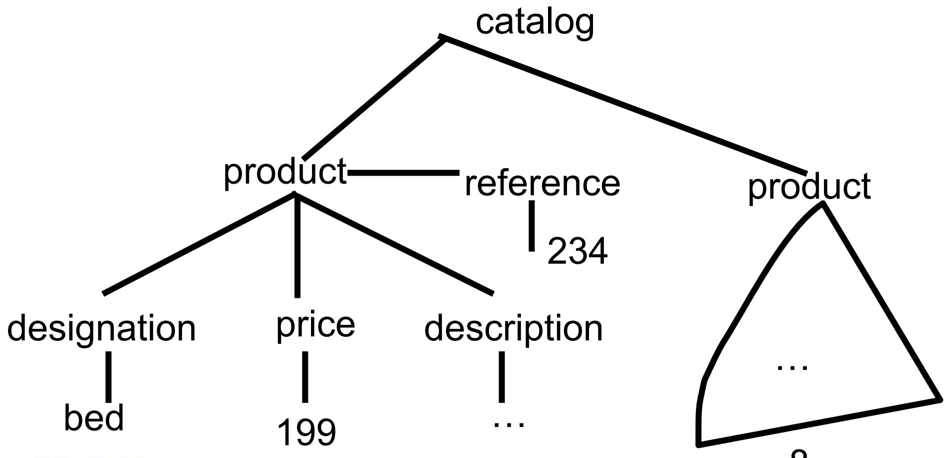
A standard for information: XML

Labeled ordered trees

where leaves are text

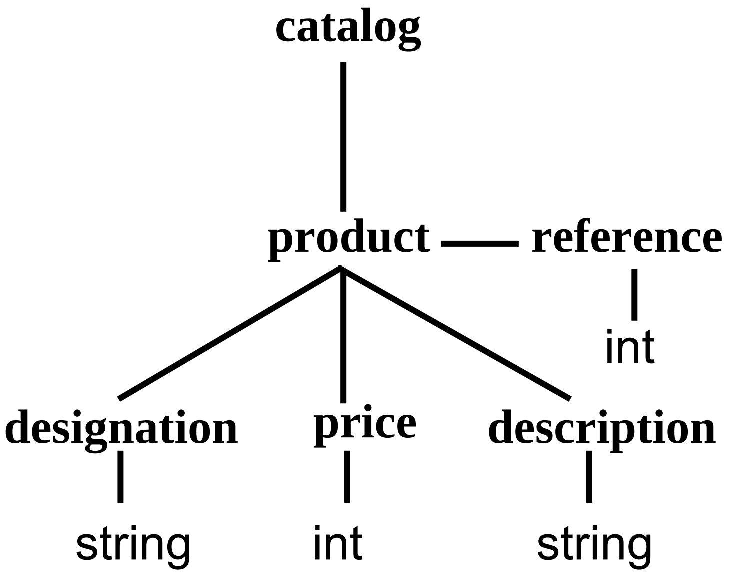
- Marriage of document and database worlds
- Is this the ultimate data model? No
- Purely syntax – more semantics needed
- Is it OK for now? Definitely yes (standard)

```
<catalog>
  <product reference="234">
    <designation>bed</designation>
    <price>199</price>
    <description> … </description>
  </product>
  <product>…</product>
</catalog>
```



The main asset of XML: flexible typing

- Applications need typing
  - XML data can be typed if needed (DTD, XML schema)
- Logical Granularity
  - neither page or document
  - but the piece of information that is needed
- Semantics and structure are in tags and paths
  - catalog, table…
  - catalog/product/price
- **Tree automata**



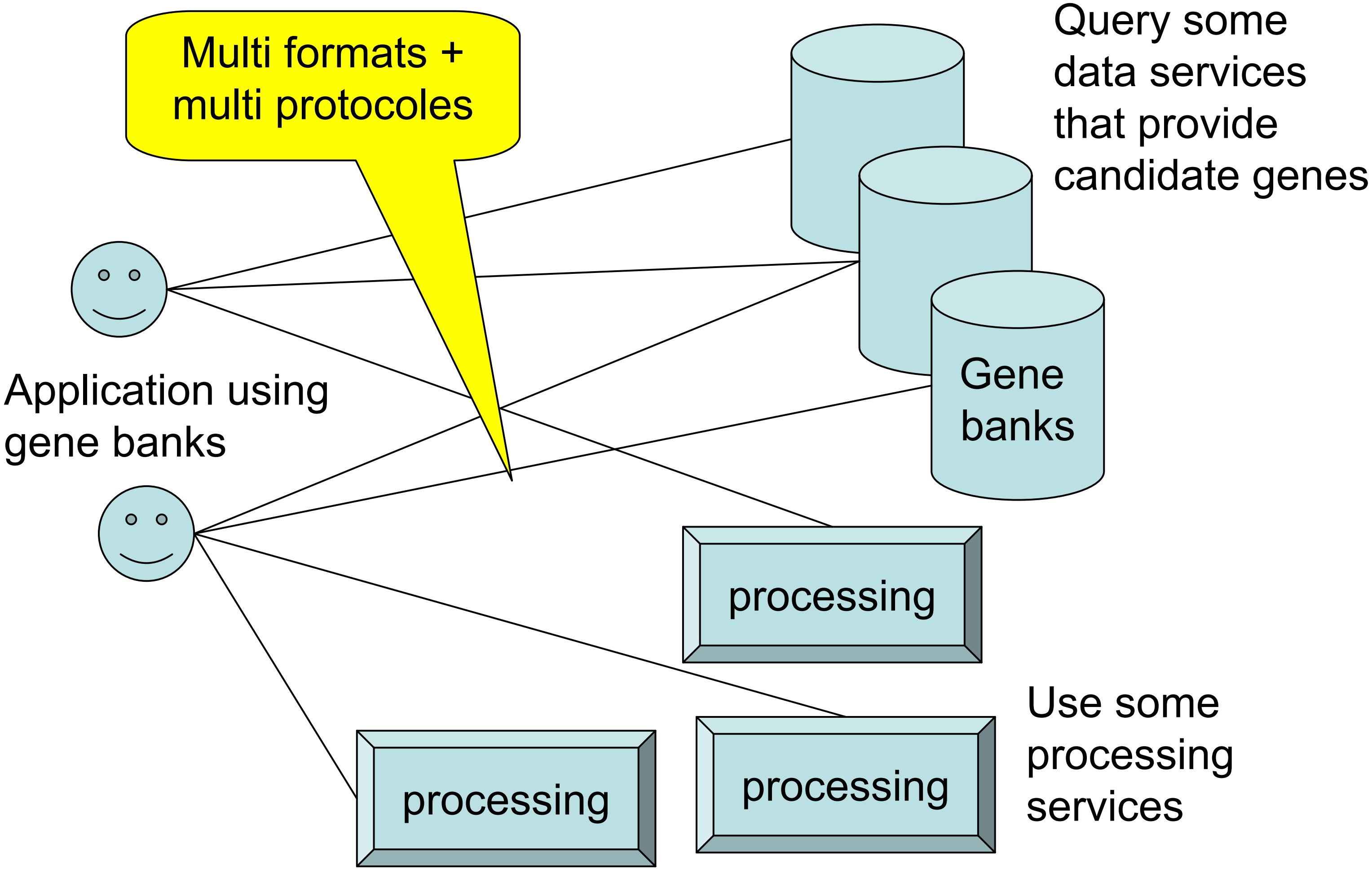
A standard for distributed computing: Web services

- Possibility to activate a method on any Web server
- Exchange information in XML: input/output are in XML

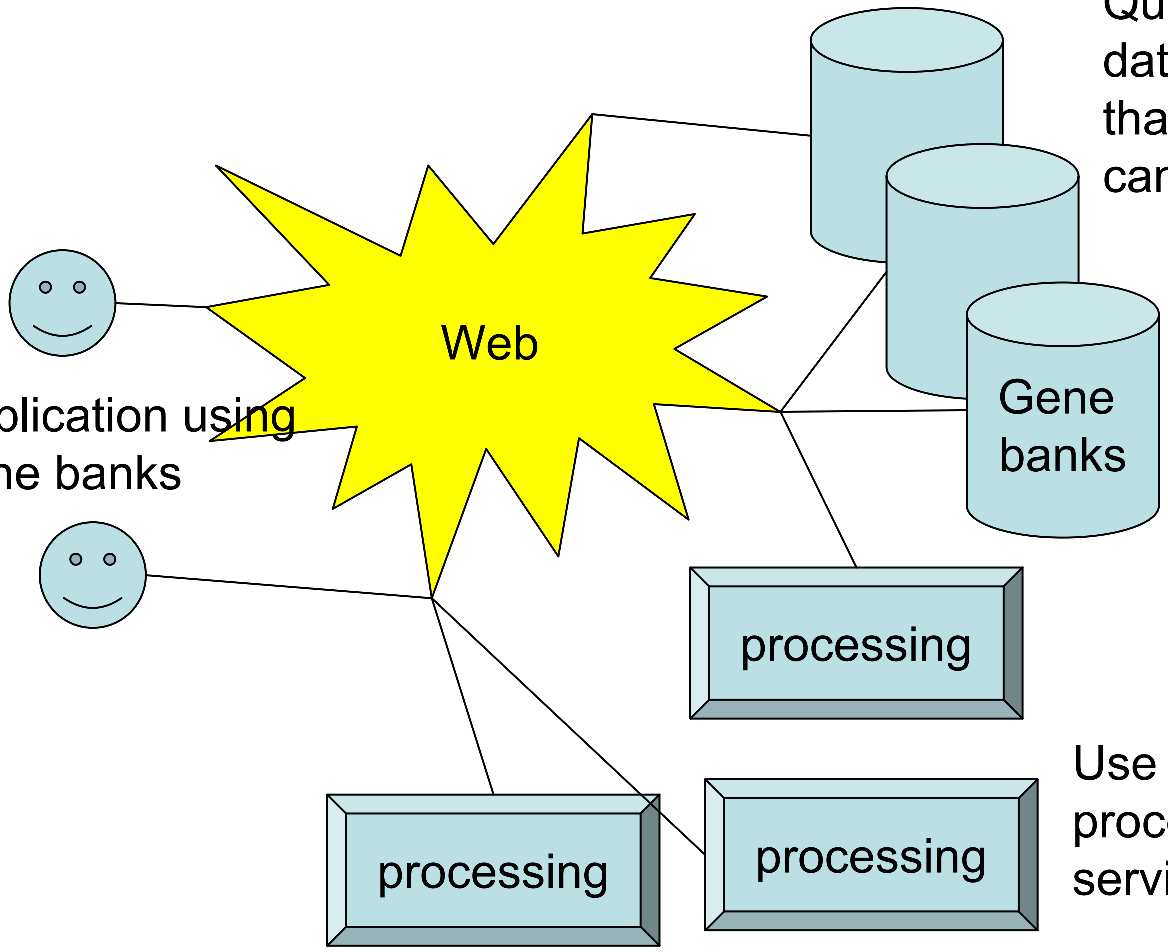
- ***Ubiquitous XML distributed computing infrastructure***

- *Something like Corba but simpler and on the Web*
- Most of the noise around e-commerce
- With XML and Web services, it is possible
  - To get information from virtually anywhere
  - To provide information to virtually anywhere

Accessing remote information


Multi formats + multi protocoles
Query some data services that provide candidate genes
Application using gene banks
Gene banks
processing
processing
processing
Use some processing services

Same with Web services



Query some data services that provide candidate genes

Application using gene banks

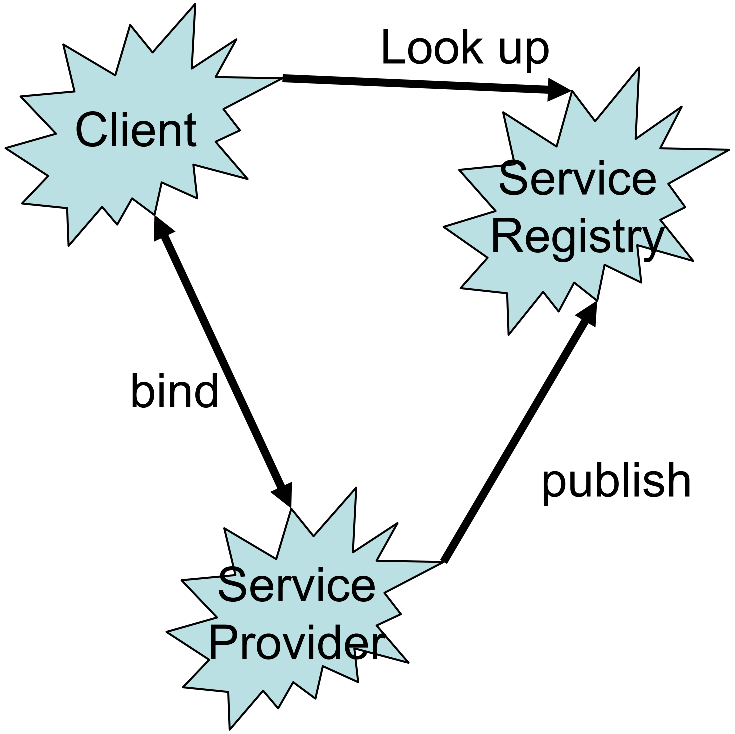
Use some processing services

The main roles

Looking for information about Gismos

1. Query some yellow-pages: Where can I find Gismos?
2. Negotiate with specialists
   - Nature of the service
   - Quality, cost
3. Get the information
   - Order, payment, delivery
   - Integration in information system
4. Eventually publish information

… and all this automatically…



Life is tough: Jargon

WSFL

XHTML

.NET

XML

DTD

RDF

RosettaNet

XSL-FO

Xschema

namespace

XSL

XSLT

ebXML

HTTPS

SOAP

HTTP

OASIS

OAGIS

ICE

MIME

WSDL

UDDI

WSDL

RSS

Organization

1. The context: XML and Web services
2. **Introduction**
3. Active XML
4. Architecture and implementation
5. Four technical issues in brief
    a) Data exchange
    b) Lazy service calls and query optimization
    c) Distribution and replication
    d) Security and access control
6. Illustration: some applications
7. Conclusion

The basis

AXML is a declarative language for distributed information management and an infrastructure to support the language in a P2P framework

Simple idea: ***XML documents with embedded service calls***

- Intensional data
  - Some of the data is given explicitly whereas for some, its definition (i.e. the means to acquire it when needed) is given
- Dynamic data
  - If the data sources change, the same document will provide different information

Besides the authors of the paper, a number of participants:

Iona Manolescu, Bernd Amann, Jerome Baumgarten and others

Example
(omitting syntactic details)

```
<resorts state=‘Colorado’>
  <resort>
    <name> Aspen </name>
    <scond> Unisys.com/snow(“Aspen”) </scond>
   <depth unit=“meter”>1</depth>
    <hotels ID=AspHotels > ….
    Yahoo.com/GetHotels(<city name=“Aspen”/>)
    </hotels>
  </resort>   …
</resorts>
```

May contain calls
to any SOAP web service :

- e-bay.net, google.com…

to any AXML web services

- ***to be defined***

Active means intensional

Manon: What's the capital of Brazil?
Dad: Let's look it up in the dictionary!

- Exchange of knowledge
  - "If you give him a fish, he can eat today. If you teach him to fish he can eat forever."
- Distributed computing

Active means dynamic

Manon: How do I get a cheap ticket to Galapagos?
Dad: Let’s place a subscription on LastMinute.com!

- Dynamic information
- With a subscription, I don’t need to ask LastMinute.com every day

Active means flexible

Manon: What are the countries in the EC?
Dad: France, Germany, Holland, Belgium, and hum… I am missing some; look in Google   !

- We can answer even if we did not finish computing the answer
- We can give the means to complete the answer

Not included
Universal brokerage by scientific service
Limited brokerage by scientific service
No brokerage (scientific data service established)
No brokerage (no scientific data service established)
No access at all

Not a new idea in databases
Not a new idea on the Web

- Mixing calls to data is an old idea
  - Procedural attributes in relational systems
  - Basis of Object Databases
- In HTML world
  - Sun’s JSP, PHP+MySQL
- Call to Web services inside XML documents
  - Macromedia MX, Apache Jelly

Organization

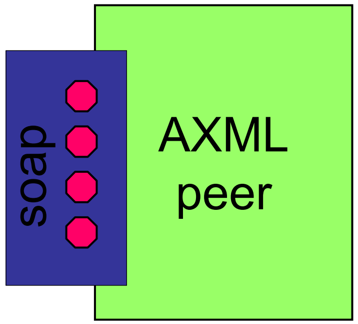
1. The context: XML and Web services
2. Introduction
3. **Active XML**
4. Architecture and implementation
5. Four technical issues in brief
    a) Data exchange
    b) Lazy service calls and query optimization
    c) Distribution and replication
    d) Security and access control
6. Illustration: some applications
7. Conclusion

A language and a system

- A language that may be used by systems that want to exchange more than static data
  - Dynamic + intensional + flexible data
- A P2P system based on exchanging AXML data

Here, we describe the system to illustrate what can be done with the language

Active XML peer



- Peer-to-peer architecture
- Each Active XML peer
  - **Repository**: manages Active XML data with embedded web service calls
  - **Web client**: uses Web services
  - **Web server**: provides (parameterized) queries/updates over the repository as web services
- Exchange of AXML instead of XML

AXML peer as a client

Call the services inside a document

Some issues in call activation

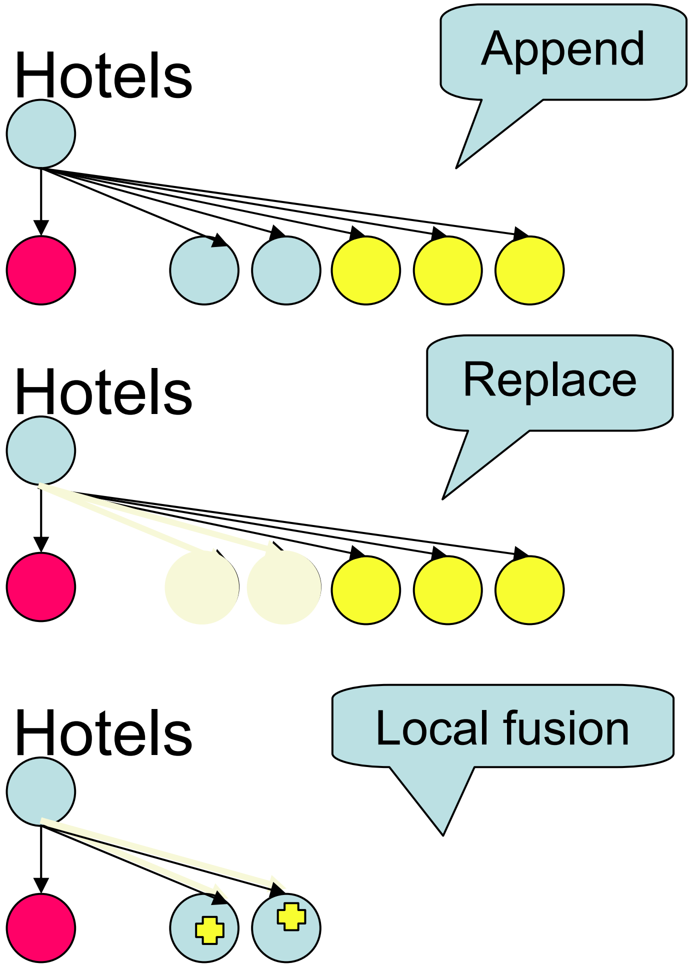
- When to activate the call?
- What to do with its result?
- How long is the returned data valid?
- Where to find the arguments?
  - Under the service call: XML,XPATH or a service call

When to activate the call

- Explicit pull mode
  - Frequency: Daily, weekly, etc.
  - After some event: e.g., when another service call completed
  - This aspect of the problem is related to ***active databases***
- Implicit pull mode : Lazy
  - When the data is requested
  - Difficulty : detect the relevant calls
  - This is related to ***deductive databases***
- Push mode
  - E.g., based on a query subscription; the web server pushes information to the client
  - E.g., synchronization with an external source
  - This is related to ***stream and subscription queries***

What to do with its result (1)

- Hotels is a data container
- Its red child is its implicit definition
- The result, a forest, is placed under Hotels
- When called more than once, one needs to define the merge policy (as an attribute of sc)
  - Policy: a web service that takes two forest (old and new) as input
  - E.g., append, replace, fusion…



How long is the returned data valid

- 0
  - Just long enough to answer a query
  - *Mediation*
- 1 day, 1 week, 1 month…
  - *Caching*
- Unbounded
  - It may remain forever: archive
  - It may remain until the service is called again in replace mode
  - Until some explicit deletion
  - *Warehousing*
- *Different policies for various portions of the document*
  - *Hybrid*

Specified as attributes

(a less simplified syntax)

```
<resorts state=‘Colorado’>
  <resort>  <name> Aspen </name>
    <scond>
    <sc valid=“1 day” mode=“lazy” >
          Unisys.com/snow(“Aspen”) <sc>
    </scond>
    <hotels ID=AspHotels >
    <sc valid=“1 week” mode=“immediate” >
         Yahoo.com/GetHotels(<city name=“Aspen”/>) </sc>
    </hotels>
  </resort>
  …
</resorts>
```

AXML peer as a server

Support for queries and updates (provided proper access rights)

Publish query and update services

- In XOQL, XPATH, Xupdate
- Also: XSL/T and Java
- Future: Xquery
- Example: a query service over the repository

```
let service Get-Hotels($x) be
  for $a in
            document(“my.resorts.com/resorts.axml")/resorts/resort,
      $b in $a//hotels/hotel
   where $a@name=$x
   return <h> {$b/name} {$b/price} </h>
```

Push mode

- The service may be activated by the client (pull)
- The service may be activated by the server (push)
  - pub/sub mechanism
  - Subscribe and receive a flow of data (stream)
- Change control
  - Management of replication, synchronization
  - Cache
- Asynchronous services
- Continuous queries
  - Send me each week the list of new movies in town

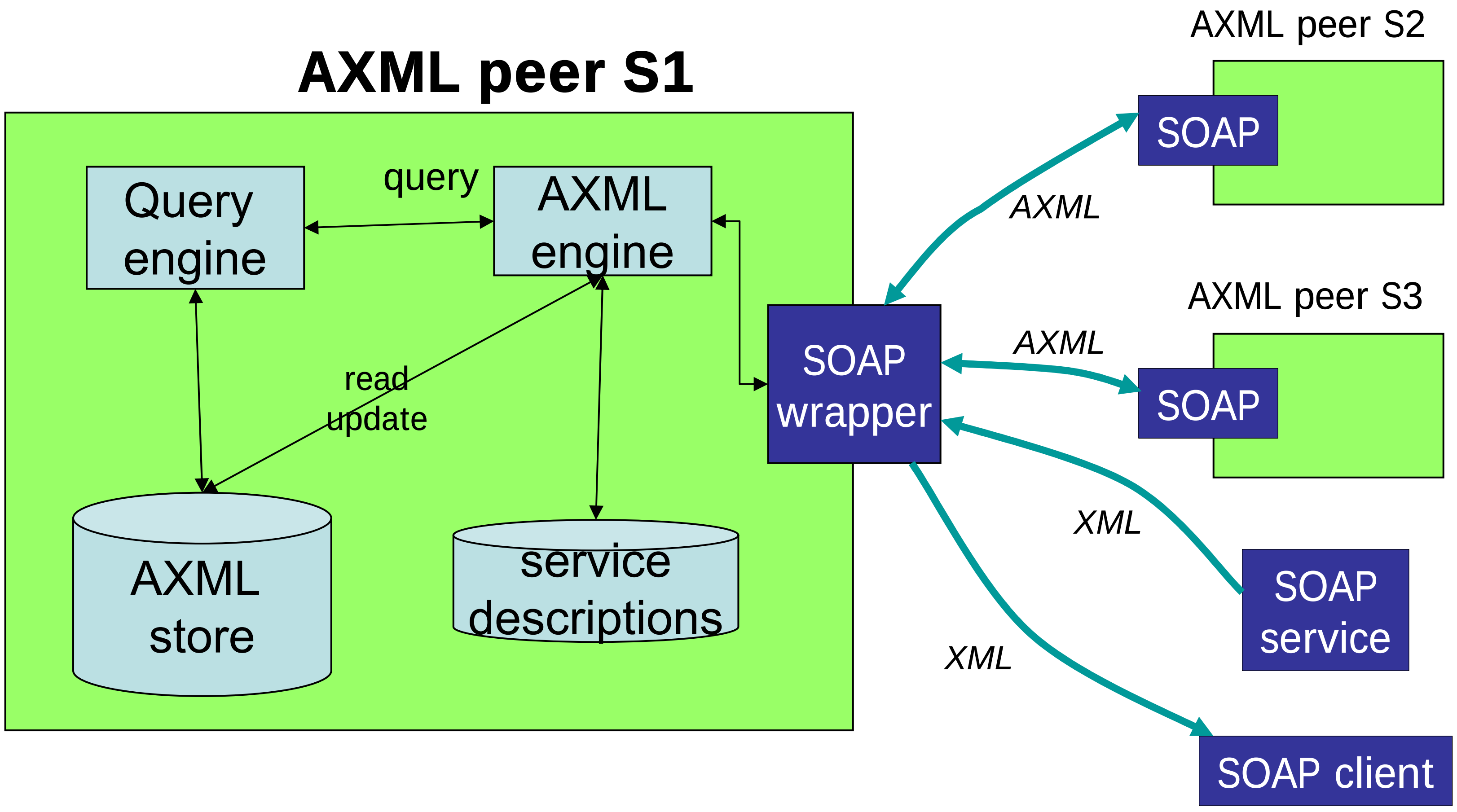
Underlying foundations

- Underlying foundations for positive AXML [pods'04]
- No order, no update, only positive queries
- Semantics defined based by rewriting systems
- Systems are confluent but possibly infinite
- Termination is undecidable
- Positive results for an important fragment based on tree automata

Organization

1. The context: XML and Web services
2. Introduction
3. Active XML
4. **Architecture and implementation**
5. Four technical issues in brief
    a) Data exchange
    b) Lazy service calls and query optimization
    c) Distribution and replication
    d) Security and access control
6. Illustration: some applications
7. Conclusion

Global architecture



Implementation

- SUN’s Java SDK 1.4
  - XML parser
  - XPath processor, XSLT engine
- **Apache** Tomcat 4.0 servlet engine
- Apache Axis SOAP toolkit 1.0
- X-OQL query processor
  - persistent DOM repository
- JSP-based user interface
  - JSTL 1.0 standard tag library

What can be an AXML peer?

- PC
  - Persistence in file system and X-OQL
- PDA or cell phone
  - Persistence in file system and XPATH
- On going: An AXML peer with mass storage
  - Data is stored in Xyleme [an XML native repository]
  - Services specified in Xquery or XyQuery
- On going: KadoP system
  - Data is stored in a P2P network
  - Kadop is much more (Dynamic Hash Table + Ontologies)
- More: cell phone; java card; a relational database…

Organization

1. The context: XML and Web services
2. Introduction
3. Active XML
4. Architecture and implementation
5. **Four technical issues in brief**
    - a) Data exchange
    - b) Lazy service calls and query optimization
    - c) Distribution and replication
    - d) Security and access control
6. Illustration: some applications
7. Conclusion

(a) Data exchange

[Sigmod03a]

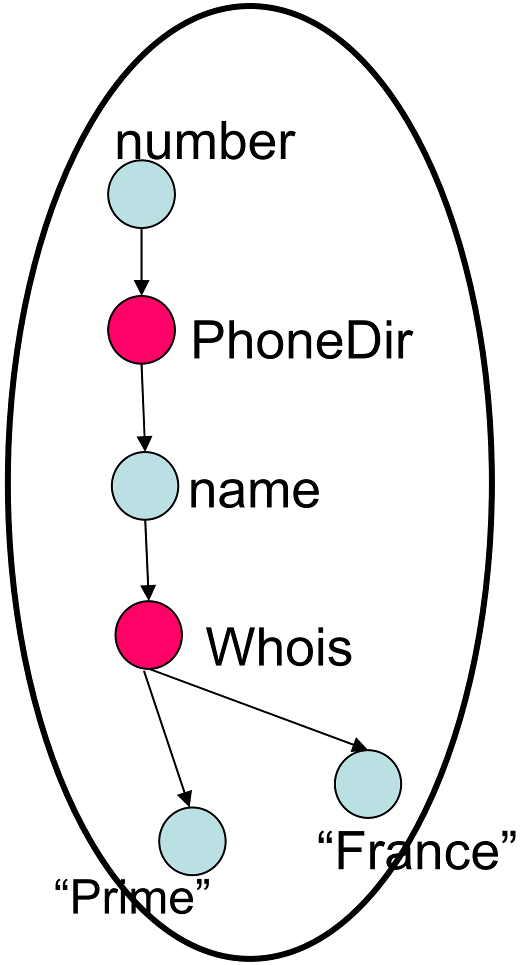
Fun technical issue: what to send? [Sigmod03]

- Send some AXML tree t
  - As result of a query or as parameter of a call
- The tree t contains calls, do we have to evaluate them?
  - If I do, I may introduce service calls, do we have to evaluate all these calls before transmitting the data?

Hi John, what is the phone number of the Prime Minister of France?

- *Find his name at whoswho.com then look in the phone dir*
- *Look in the yellow pages for Raffarin's in phone dir of www.gov.fr*
- *(33) 01 56 00 01*

To call or not to call



- Alternative1
  - Send <number>www.gov.fr/PhoneDir(<name> whoswho.com/Whois (“Prime”, “France”) </name></number>)
- Alternative2
  - Call whoswho.com/Whois(“Prime”, “France”)
  - Send <number>www.gov.fr/PhoneDir (<name>Raffarin</name>)</number>
- Alternative3
  - Call whoswho.com/Whois(“Prime”, “France”)
  - Call www.gov.fr/PhoneDir(<name>Raffarin</name>)
  - Send <number>*(33) 01 56 00 01* </number>
- Allow to control who does what

Why control the materialization of calls?

- Because of constraints
  - I don’t have the right credentials to invoke it,
  - It costs money,
  - Maybe the receiver doesn’t know Active XML!
- For added functionality, e.g.
  - Intensional data allows to get up-to-date information.
- For performance reasons, e.g.
  - A proxy can invoke services on behalf of a PDA.
- For security reasons.
  - I don’t trust this Web service/domain
- … and many more reasons you can think of!

Example: security

- Peers exchange AXML documents containing service calls
- A server (resp. client) might ask the client (resp. server) to do something « bad »:

```
<sc>www.qod.com/QuoteOfDay </sc>
<quote date="july 8th 2002">
    My heart was bumping <context>Tskitishvili, picked 5th in the
    NBA draft by the Denver Nuggets</context>
    <sc>buy.com/BuyCar(« BMW Z3 »)</sc>
</quote>
```

- We do not trust www.qod.com; we want it to evaluate all calls before sending us some data

To call or not to call

- Definition of an extension of XML schema that distinguishes between number and a call returning a number (name) → number
- What is expected by the client?
  - … Phone: number …

  Evaluate all calls and return phone number
  - … Phone: (name) → number

  Get the name of the president
  - … Phone: any

  Do not evaluate any call and return result

To call or not to call

- Given some data to send d
- Given some agreed type t for the exchange in **WSDL_{int}**
- Given the published types of the services that are used

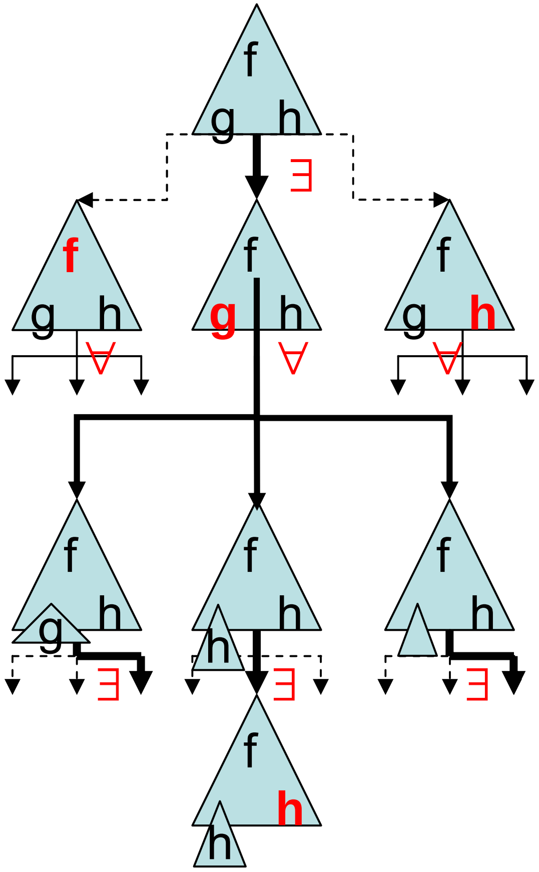
Find a rewriting of d of type t

- Safe rewriting: one that for sure leads to t
  - We know without making any call
- Possible rewriting: one that possibly leads to t
  - Depending on the answers of the services
  - I may need to try more than one rewriting to succeed

Safe rewritings and alternating games

- Strategy works as follows
- I choose a call g to perform (∃ move)
- The adversary may choose any answer to g of the correct type (∀ move)
- I choose a new call to perform, and so on
- Winning strategy: guaranteed to get to a document of the target type

- Difficulties
  - Infinite search space: vertical; horizontal
  - The result of a Web service call is unknown – we just know its signature
  - We want an efficient solution: parallelism



Results

- The general problem is undecidable
- Restrictions in the implementation
  - Left-to-right rewriting: No “going back and forth”
  - K-depth rewriting: bound on the nesting of function calls
  - Search space still infinite but finitely representable
- Under these restrictions
  - Algorithm (based on automata) for finding a strategy for safe rewriting if it exists
  - Ptime for “deterministic” schemas
- Related work
  - Context-free games [MuschollSchwentickSegoufin04]

(b) Query optimization

[Sigmod04]

On going work – extension of Query-Subquery [Vieille]

Fun technical issue: answer fast

- **Lazy mode: call a service only if necessary**
- **Push queries**
  - Materialize only the minimal set of relevant data
- Why is it not trivial?
  - Dynamically during query evaluation: we have to block the query processor during the evaluation of calls (a bad idea)
  - Before query evaluation: not easy to find the lazy service calls that may contribute to the query

    A service call may contain more service calls – recursion
  - Distribution

A simple sub-case: Datalog

- Relations and deductive databases
- Datalog program

  r(x,y):- s(x,z),t(z,y)

  r(x,y):- a(x,y)

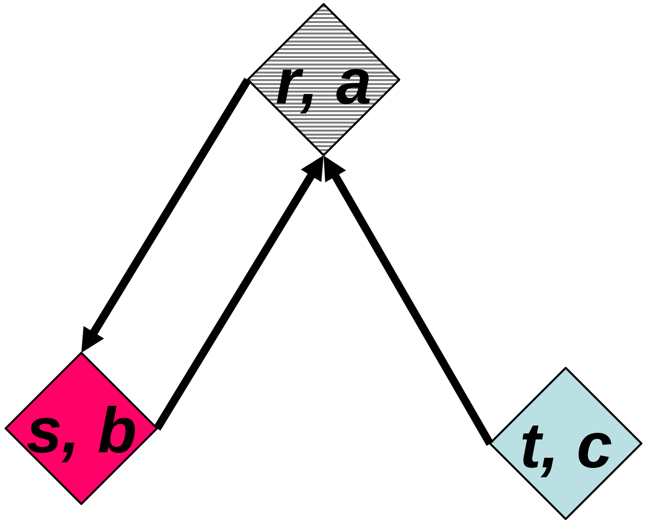
  t(x,y):- c(x,y)

  s(x,y):- r(x,y), b(y,z)
- Distributed datalog

  r and a on grey site

  s and b on red site

  t and c on blue site



r(x,y):- s(x,z),t(z,y) r(x,y):- a(x,y)
t(x,y):- c(x,y) s(x,y):- r(x,y), b(y,z)

Classical QSQ rewriting

q(y) :- r'(a,y)
inr'(a) :-

h10(x) :- inr'(x)
h11(x,z) :- h10(x), s'(x,z)
h12(x,y) :- h11(x,z), t'(z,y)
ins'(x) :- h10(x)
int'(z) :- h11(x,z)
r'(x,y) :- h12(x,y)

h20(x) :- inr'(x)
h21(x,y) :- h20(x), a(x,y)
r'(x,y) :- h21(x,y)

h30(z) :- int'(z)
h31(z,y) :- h30(x), c(x,y)
t'(z,y) :- h31(z,y)

h40(x) :- ins'(x)
h41(x,y) :- h40(x), r'(x,y)
h42(x,z) :- h41(x,y), b(y,z)
inr'(x) :- h40(x)
s'(x,z):- h42(x,z)

Materialize only relevant data
Push queries
Sideway information passing

Distributed QSQ rewriting (one possible way)

```
r(x,y):- s(x,z),t(z,y)      r(x,y):- a(x,y)
t(x,y):- c(x,y)             s(x,y):- r(x,y), b(y,z)
```

r, s, t on three sites – grey, red, blue

Site r

```
q(y) :- r’(a,y)
inr’(a) :-
h10(x) :- inr’(x)
r’(x,y) :- h12(x,y)
h20(x) :- inr’(x)
h21(x,y) :- h20(x), a(x,y)
r’(x,y) :- h21(x,y)
h41(x,y) :- h40(x), r’(x,y)
inr’(x) :- h40(x)
```

Site s

```
h11(x,z) :- h10(x), s’(x,z)
ins’(x) :- h10(x)
h40(x) :- ins’(x)
h42(x,z) :- h41(x,y), b(y,z)
s’(x,z):- h42(x,z)
```

Site t

```
h12(x,y) :- h11(x,z), t’(z,y)
int’(z) :- h11(x,z)
h30(z) :- int’(z)
h31(z,y) :- h30(x), c(x,y)
t’(z,y) :- h31(z,y)
```

A-QSQ

- Extensions of QSQ
  - Distribution: the rewriting may be achieved locally
  - Trees: unification and query composition
- Detection of termination becomes an issue
- We can start computing and getting results before the rewriting is finished
- We can answer intensionally
  - Provide the intension instead of the extension
  - E.g. to facilitate the detection of termination
- We can move knowledge around
- We can exchange knowledge
  - E.g. rule 2 done, 3 pending (w.com not answering)

(c) Distribution and replication

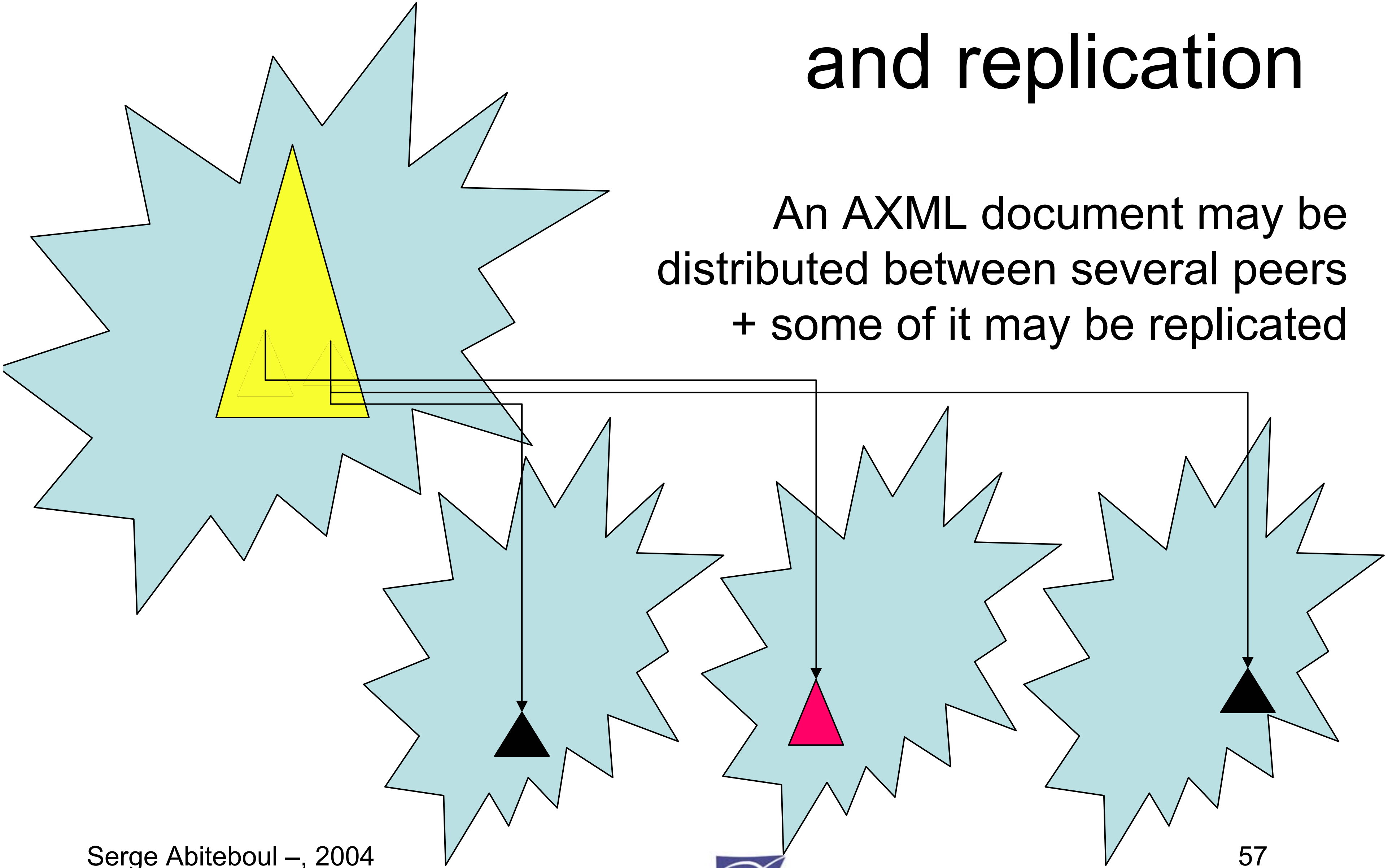
[Sigmod03b]

Distribution and replication

- Devices with limited capabilities
  - Cell phone, pda, home appliances…
  - Storage space
  - Computational power
  - Network bandwidth
- Therefore, we need to:
  - **Distribute** the work among devices, by:
    - Calling external services (done !)
    - Distributing documents across several devices (peers)
  - **Replicate** documents and services, to allow for "local" computation and improve parallelism

Distribution and replication

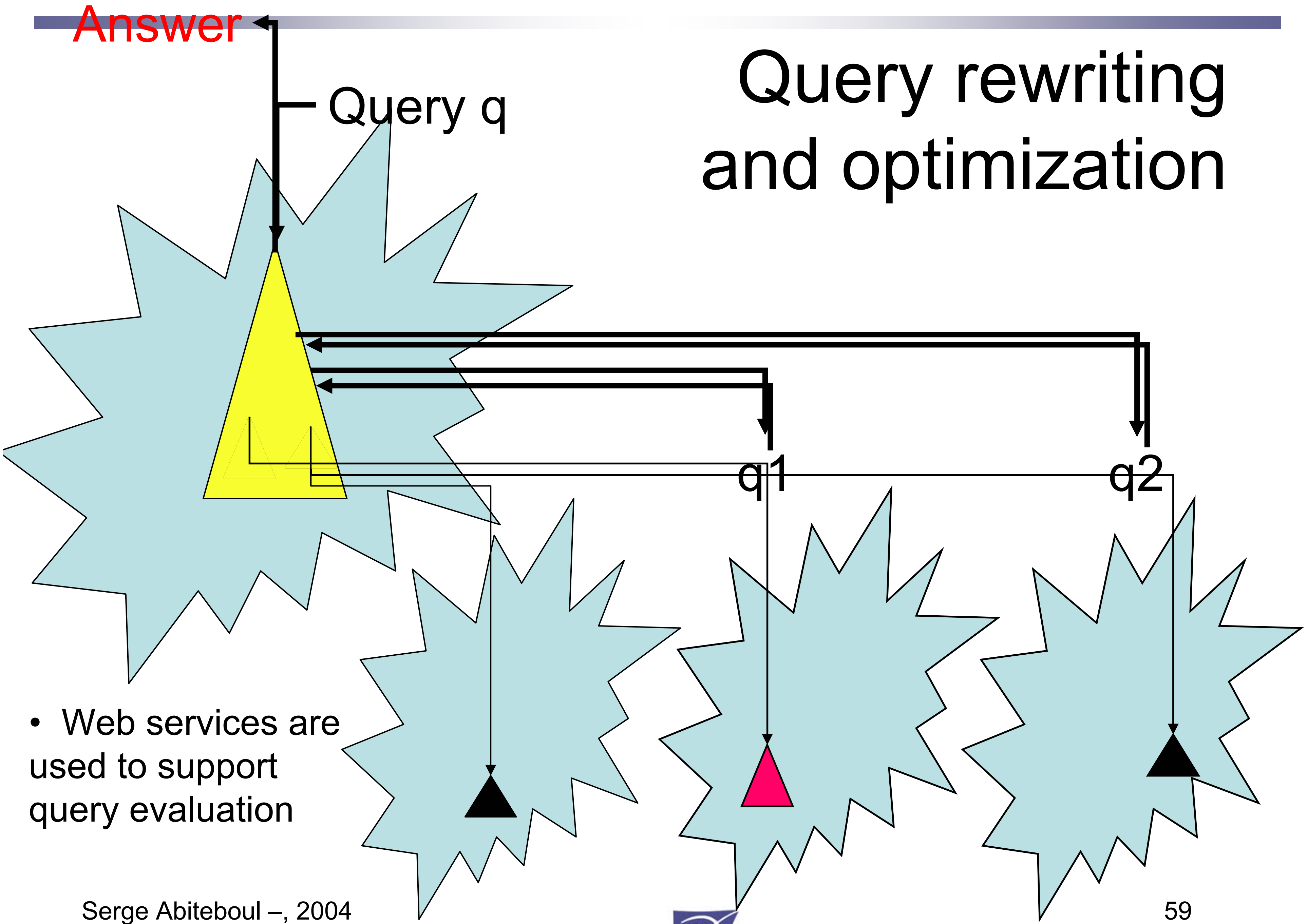
An AXML document may be distributed between several peers + some of it may be replicated



Example

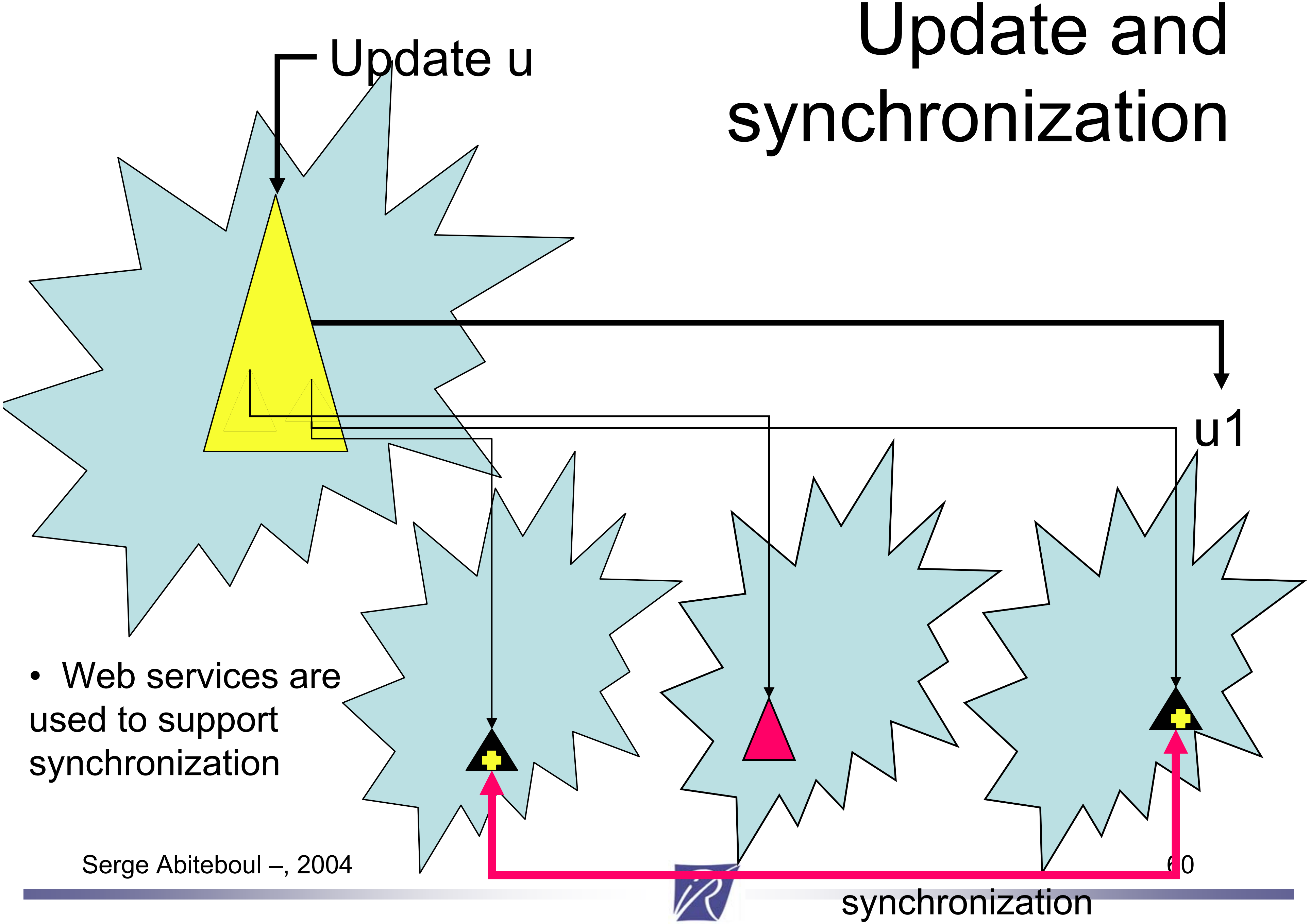
- Suppose that access to guides of resorts in Colorado is charged
- I may want to replicate the Aspen guide on my PDA (some of the data is intensional)
- I want it also replicated on a proxy
- Some of it may be only on the PDA (e.g., some pictures)
- The intensional data (e.g., temperature) has to be refreshed regularly on my PDA
- When I annotate the guide in my PDA, I want the annotations to be replicated on the proxy to be used by the entire family and my friends

Query rewriting and optimization



- Web services are used to support query evaluation

Update and synchronization



- Web services are used to support synchronization

Technical issues

- A *data model* for AXML with distribution and replication
  - Query and update language; by default, ignore distribution + replication
  - Means to specify explicitly a particular copy
  - Supported by AXML Web services
- Query evaluation
  - Cost model
  - Optimization and load balancing when there is replication
- Update propagation to support replication
- Decide which data and services to replicate to improve performances
  - When replicating a service, need to replicate data that it uses for improving performances, need to adapt the code

(d) Security and access control

Security on the Web

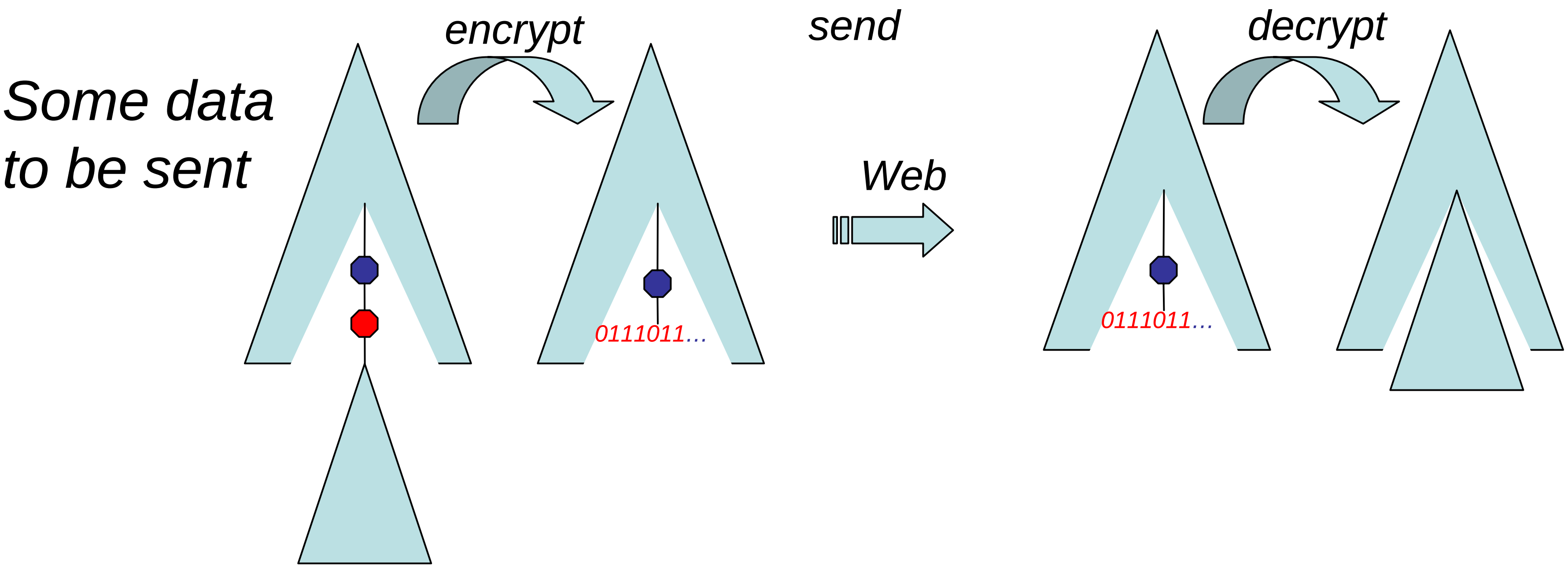
- Lots of proposed standards around XML
  - W3C XML key encryption
  - W3C XML encryption specification
  - W3C XML signature specificatin
  - Oasis Security Assertion markup language
- Active XML support
- Example: encryption of part of an XML tree using public key cryptography

```
<EncryptedData Id? Type?
   MimeType? Encoding?>
<EncryptionMethod/>
<ds:KeyInfo>
     <EncryptedKey>
     <AgreementMethod>
     <ds:KeyName>
     <ds:RetrievalMethod>
     <ds:*>
</ds:KeyInfo>
<CipherData>
     <CipherValue>
     <CipherReference URI?>
</CipherData>
<EncryptionProperties>
</EncryptedData>
```

Simple example

- publicKey@anypeer(user) → string
- privateKey@mypeer(user) → string
- encrypt@anypeer(publicKey,data) → encryptedData
- decrypt@mypeer(privateKey,encryptedData) → data

Simple example



- decrypt@p2(privateKey@p2(Alice), **…**)
- encrypt@p1(publicKey@p2(Alice),data))

Encryption does not even have to be visible by applications

Controlling the evaluation

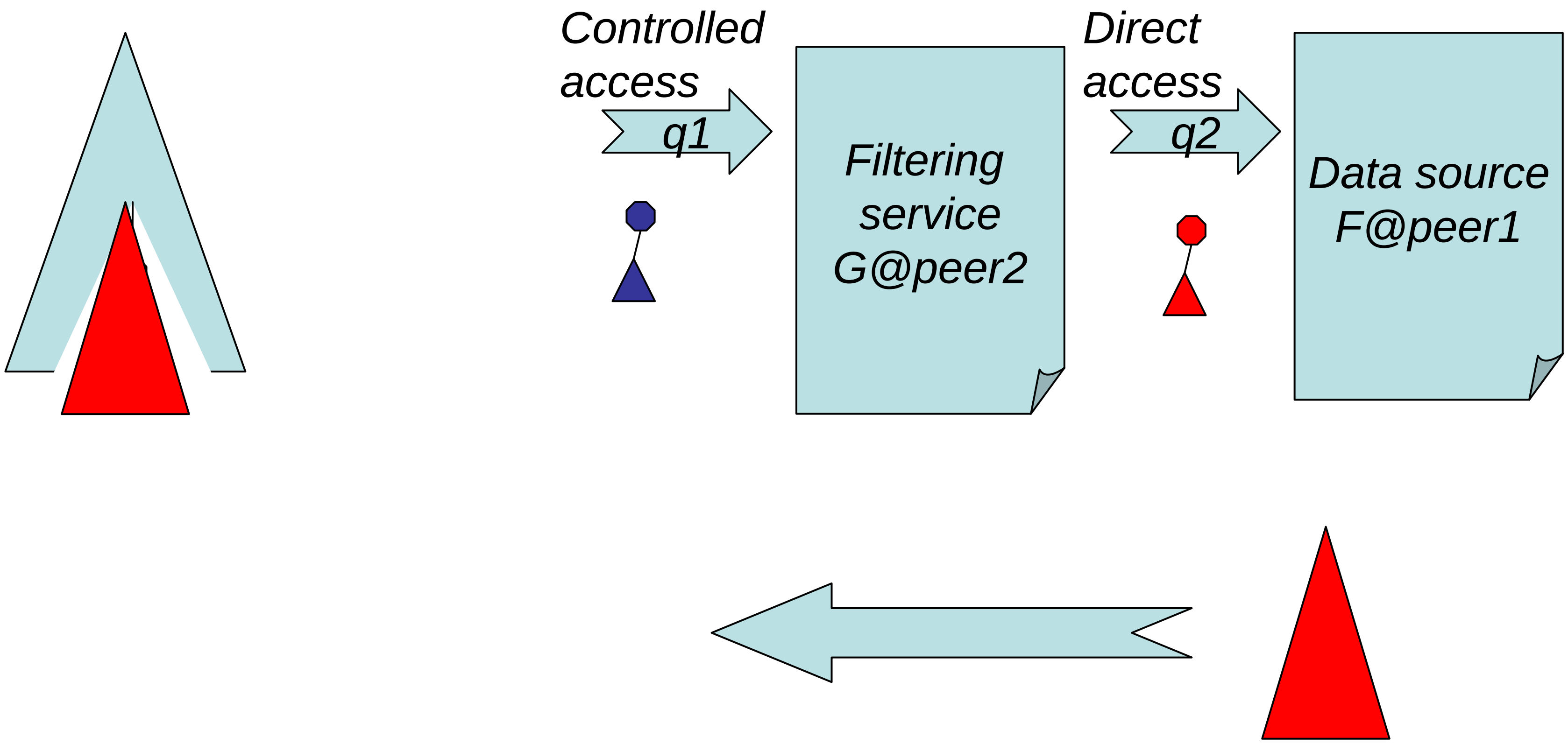
- Based on the type of the exchange
  - The type determines that the privateKey is obtained and the data is encrypted before being sent
  - The type determines that the data is not decrypted before being sent
    - In fact, cannot be performed (privateKey not available)
- Risky
  - A type error may lead to sending the private key
- Current work: rewriting techniques
  - Security is concentrated in security rules
  - The rules determine which portion of data to encrypt and how
  - Rules may also be used for other aspects: transaction, optimization, provenance…

Security: more

- More complex scenarios
- Signature
- Authentication
- Delegation

- Remark: from the point of the client, the fact that the data is encrypted is not visible

Access control – based on joint work with Lucent



Controlled access

q1

Filtering service G@peer2

Direct access

q2

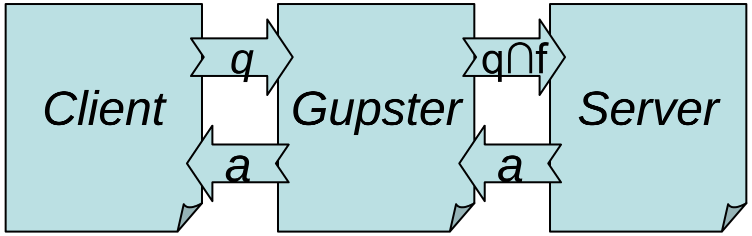
Data source F@peer1

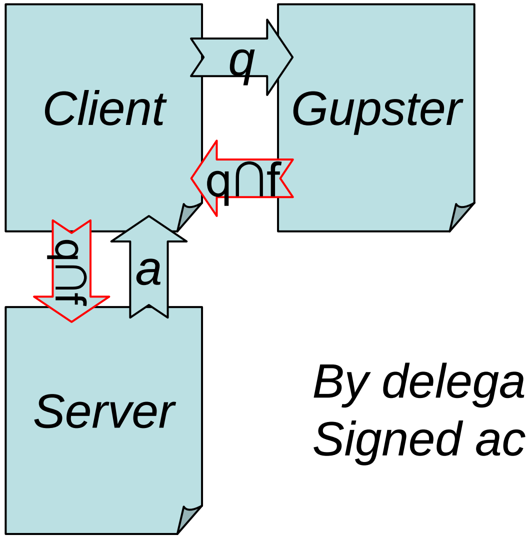
Example

- Use of the Gupster system [Lucent]

  Query q & AccessFilter f

  → q ∩ f
- Gupster is closed under intersection





By delegation
Signed access rights

Organization

1. The context: XML and Web services
2. Introduction
3. Active XML
4. Architecture and implementation
5. Four technical issues in brief
   a) Data exchange
   b) Lazy service calls and query optimization
   c) Distribution and replication
   d) Security and access control
6. **Illustration: some applications and current work**
7. Conclusion

Some applications

– Data mngt. in mobile peers	in EC Project **DbGlobe**
• AXML peer on a cell phone	
• Context awareness	
– Web warehousing	in RNTL project **e.dot**
• Use AXML to build and enrich a warehouse	for a warehouse on food risk and in [ecdl-demo’03]
– P2P auctioning	in [vldb-demo’02]
– News brokering	in [vldb-demo’03a]
– Distributed workspace mngt.	in [vldb-demo’03b]

Other applications considered by/with partners

- Software distribution
  - Distribution and customization of software packages
  - Linux distribution with MandrakeSoft
  - In EC Project **Edos**
- Network configuration
  - Exchange information to configure hard/software components
  - In **Swan** Project by INRIA-Rennes, Alcatel, FT et al.
  - On-going: Error diagnosis using Petri-net unfolding and AQSQ
- Personal data management
  - Access control with Lucent

Organization

1. The context: XML and Web services
2. Introduction
3. Active XML
4. Architecture and implementation
5. Four technical issues in brief
   a) Data exchange
   b) Lazy service calls and query optimization
   c) Distribution and replication
   d) Security and access control
6. Illustration: some applications
7. **Conclusion**

Distributed Information Management

Information used to live in islands but it is changing

- Golden triangle: XML, Web services, Queries…
- More semantics needed: semantic Web
- Mine of new problems in
  - Query optimization, security, man-machine interface, change control, transaction management
- Theoretical tools
  - Database theory, automata, tree automata, type theory, logic programming…

Active XML simple idea – complex problems

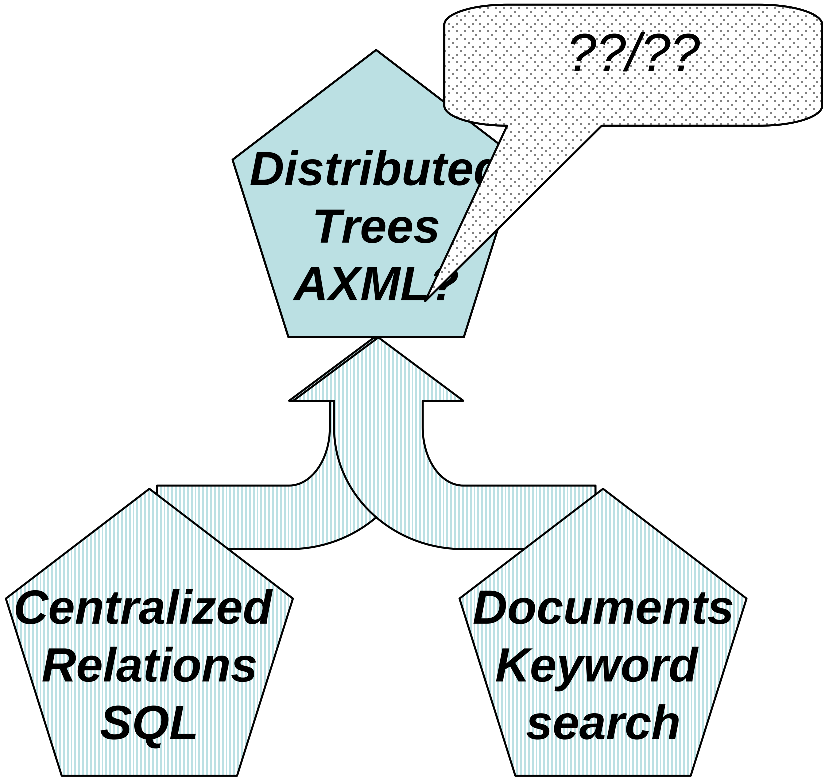
- XML + embedded service calls
- A powerful means of rapidly deploying data-centric, distributed applications
- Brings together in a unique setting
  - Document processing
  - Deductive databases
  - Active databases
  - Distributed databases
  - Stream data and pub/sub
- Is this reasonable?

If you give him a fish, he can eat today.
If you teach him to fish he can eat forever

Languages for data exchange

- Centralized databases
  - Data: relations
  - Query: FOL/SQL

- Web data - Officially:
  - Data: XML
  - Query: ??/Xquery
  - I am not convinced
  - OK for XML repositories?
  - Not enough for the Web



Now open source (part of Object Web consortium)

http://activexml.net

Obrigado