Dynamic XML documents with distribution and replication

Angela Bonifati (currently in Icar-CNR, Italy)

Joint work with:

Serge Abiteboul, Gregory Cobéna, Ioana Manolescu (INRIA), Tova Milo (INRIA &Tel Aviv U.)

Outline

- Motivation
  - Dynamic XML documents and Active XML
  - Peer-to-peer data management
- Distributing and replicating AXML documents
  - Data model
  - Replication / distribution alternatives
- Peer-to-peer querying of AXML^{DR} documents
- Automatic (self-configurable) replication
- Conclusion and perspectives

Dynamic documents

- Documents having
  - An extensional (static) part
  - An intensional (dynamic) part
- Many existing applications
  - HTML + embedded active components
    - Dynamic page generation from databases (ASP, PHP, JSP)
    - Pages with embedded "update" features
  - XML-structured pages + active components
    - CGI, beans, …, web services (e.g. DreamWeaver)

Distribution and replication

- Interest of data distribution
  - Move data fragments geographically close to their users
  - Allows for distributed processing
- Interest of data replication
  - Higher availability
  - Better performance

The focus of this work

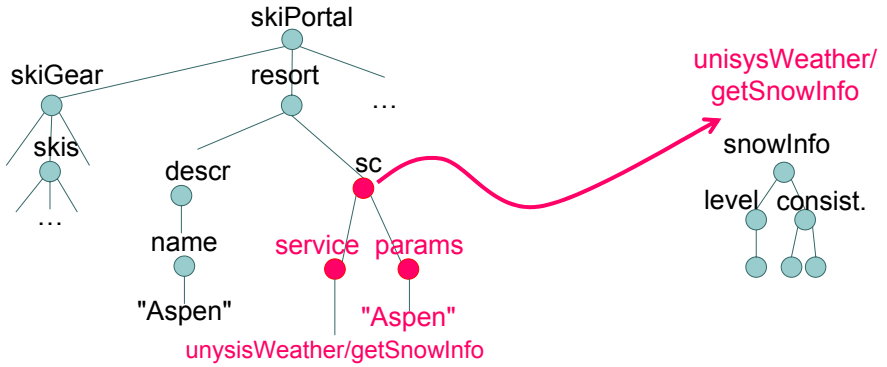
- Distribution and replication of dynamic documents.
- Focus on the new aspects

Context: Active XML (AXML)

- A language: XML with embedded service calls
- A peer-to-peer system
- Each peer
  - Repository of intensional (AXML) documents
  - Server: provides Web services (XQuery)
  - Client: when invoking the embedded service calls
  - And many more cool features
    - exchanging intensional data (yesterday morning)
    - continuous services
    - etc.
- AXML peers have already inherent distribution

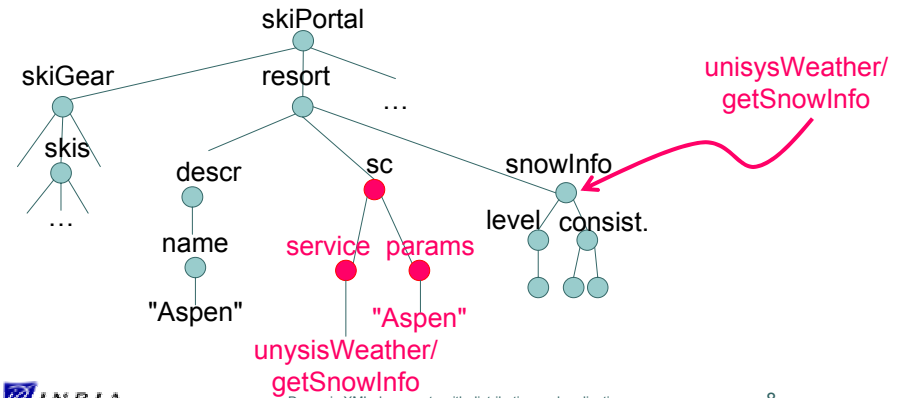
Active XML documents

- XML documents embedding calls to external Web services



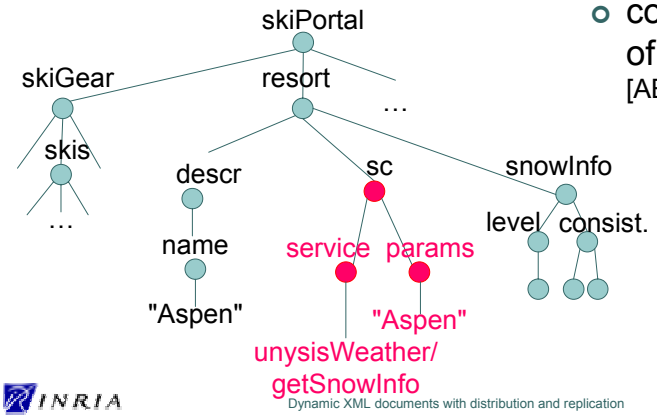
Active XML documents

- XML documents embedding calls to external Web services



Active XML documents

- XML documents embedding calls to external Web services
- control of activation of service calls [ABM01, ABM+02]



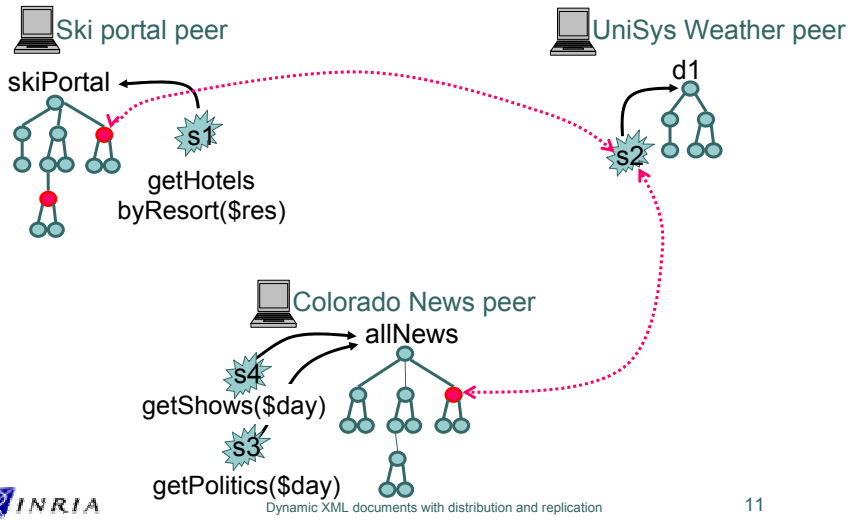
AXML web services: declarative XML queries

- Many useful services may be defined as queries over (A)XML documents

- Define service uniSysWeather / getSnowInfo($resortName) as:

```
for $s in document("weather.xml")//snowInfo,
    $n in $s//resort/ @name
where $n=$resortName
return $i
```

Peer to peer deployment of AXML documents



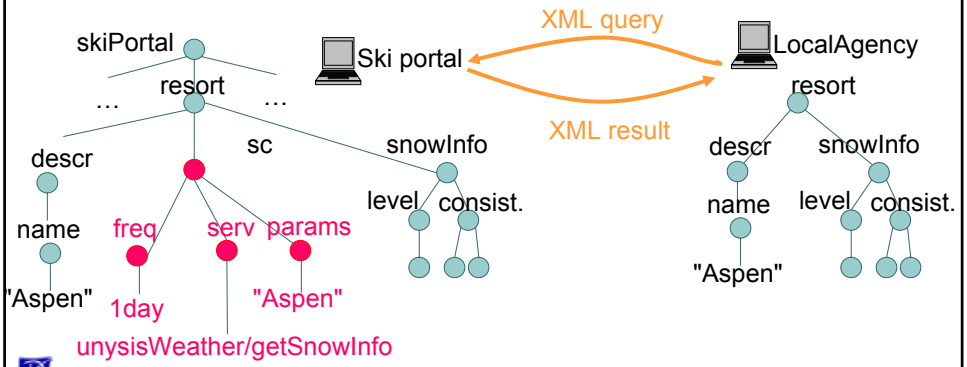
Distributing and replicating dynamic documents by example

Plan

- Replication of dynamic data
- Distribution of dynamic data
- Querying distributed and replicated data
- Cost model
- Self-configurable replication

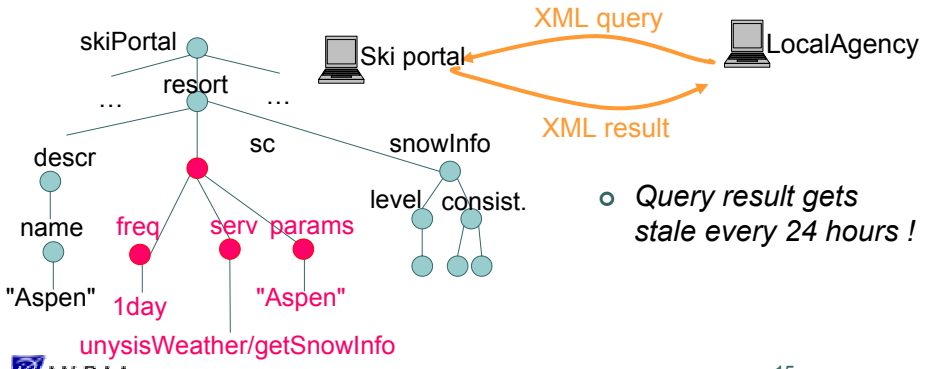
Replication of dynamic documents

- Peer localAgency wishes to hold the data regarding resorts with high snow levels
- naive solution: localAgency queries skiPortal, stores result



Replication of dynamic documents

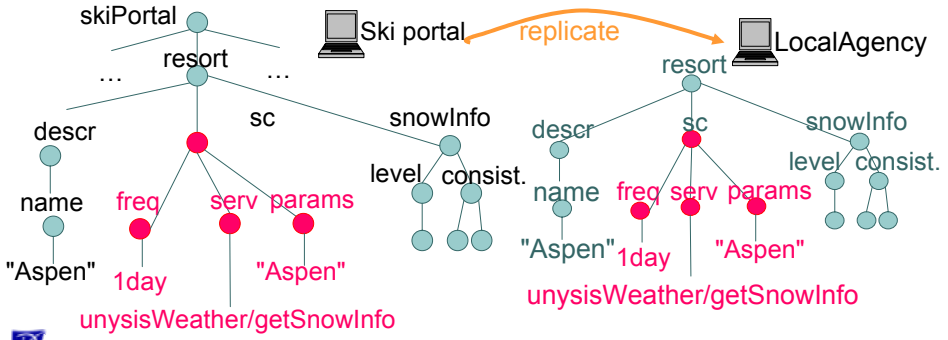
- Peer localAgency wishes to hold the data regarding resorts with high snow levels
- naive solution: localAgency queries skiPortal, stores result



- *Query result gets stale every 24 hours !*

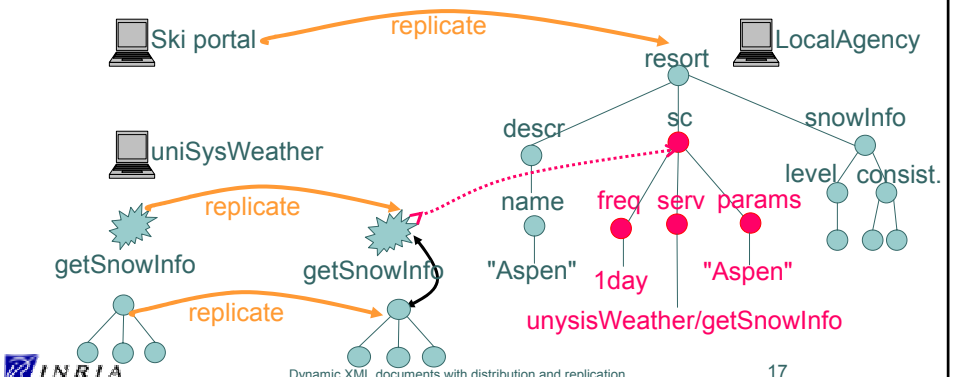
Replication of dynamic documents

- Avoiding stale data:
  - propagating the changes
  - replicate data needed for the service call, and (adjust) the query



Replication of dynamic documents

- Providing complete independence to the LocalAgency peer:
  - Replicate *also* the service definition and the data necessary in order to make the call (*adjust the service definition !*)



Replication issues

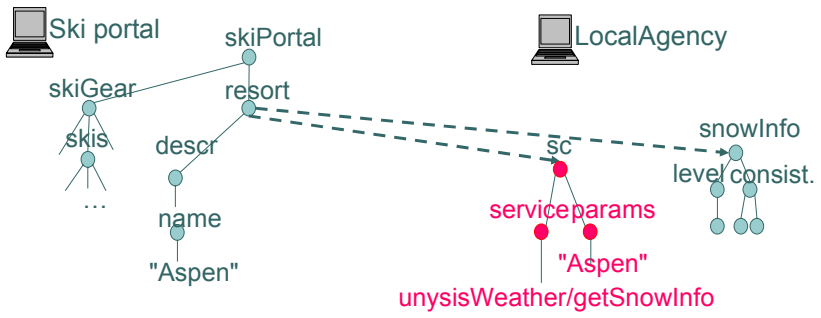
- Issues:
  - Which data / services to replicate ?
  - How to adjust the definition of the replicated services ?
- Contributions: algorithms and cost model

Plan

- Replication of dynamic data
- Distribution of dynamic data
- Querying distributed and replicated data
- Cost model
- Self-configurable replication

Distribution of dynamic documents

- Not all data can be replicated everywhere (space, copyright etc.)
- Data or service calls may be distributed among several peers.



Distribution issues

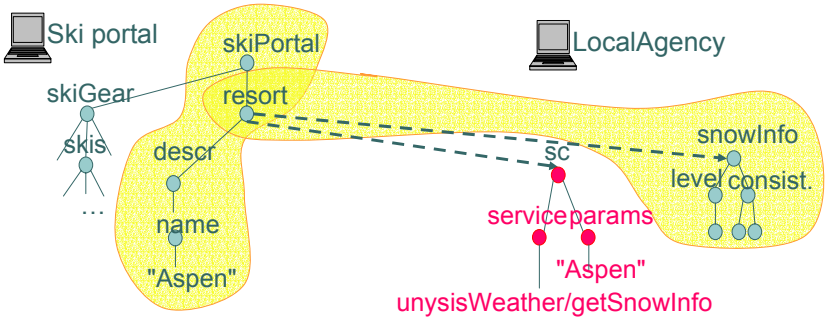
- Issues
  - The system must choose the peers contributing to query answering
  - The user may want to restrict to just some peers
- Contributions
  - XQuery extension XQuery^{DR}
  - Cost model
  - Distributed P2P query processing algo

Plan

- Replication of dynamic data
- Distribution of dynamic data
- Querying distributed and replicated data
- Cost model
- Self-configurable replication

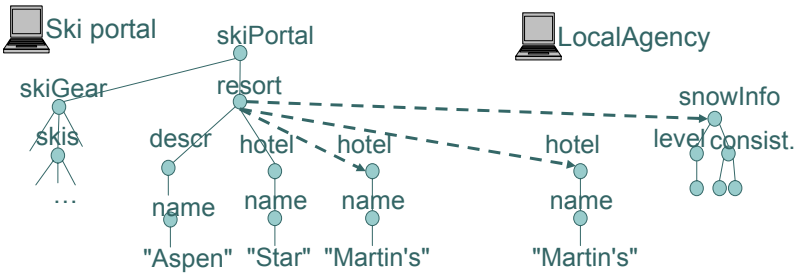
Querying distributed dynamic data

- Doc("skiPortal.xml")//resort[descr/name="Aspen"]/snowInfo
- The query must be split over the peers



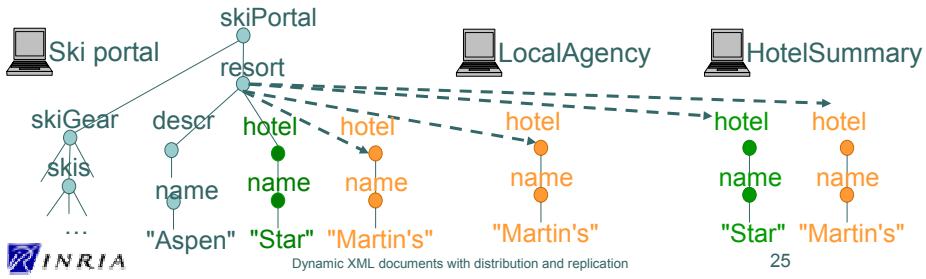
Location aware model and query language

- Universally unique IDs
- Enhance path expressions, specifying versions to query, e.g.:
  - {Doc("skiPortal.xml")/resort//hotel}@ANY;
  - {Doc("skiPortal.xml")/resort}@MASTER{//hotel}@LOCAL



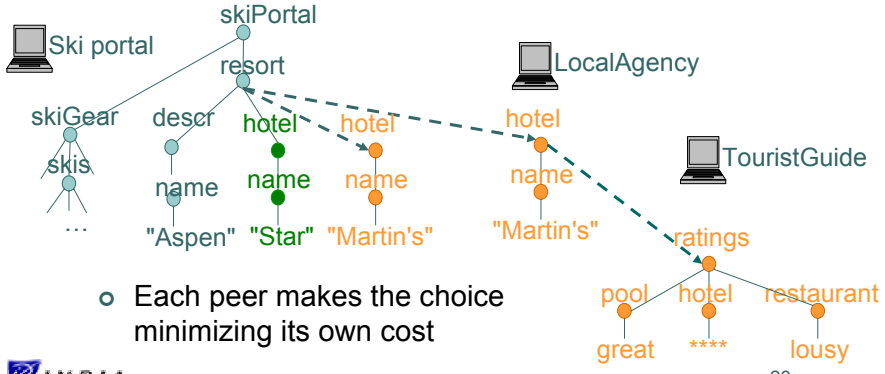
Query processing (1/2)

- Query at skiPortal: {Doc("skiPortal.xml")/resort//hotel}@ANY;
- skiPortal identifies local fragment: Doc("skiPortal.xml")/resort
- The remaining query //hotel may be evaluated at: SkiPortal, or LocalAgency, or HotelSummary
- Choice based on cost



Query processing (2/2)

- Ski portal may delegate //hotel to LocalAgency
- … which in turn may delegate to TouristGuide



- Each peer makes the choice minimizing its own cost

Plan

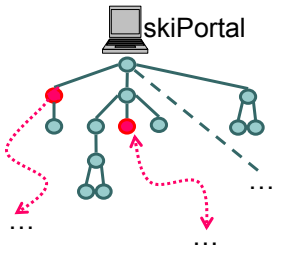
- Replication of dynamic data
- Distribution of dynamic data
- Querying distributed and replicated data
- Cost model
- Self-configurable replication

Cost model

- Principles: 1.*unify service call activations and queries*
  2. *cut queries/services into 1-peer pieces*
- Cost of (peer, query) pair (p_i, q_i)
  from the perspective of a peer p_k
- Cost function includes
  - Objective parameters: real costs (network, storage etc.) at all peers
    - Many of these are not known to p_k
  - Subjective parameters: weights of objective costs from the perspective of p_k

Cost Model: Observable performance

- On each peer
  - Service calls in local documents
  - Queries are asked
  - Local services can be called from outside



- Cumulated daily costs = peer's *observable performance*
- *May be changed through replication*

Plan

- Replication of dynamic data
- Distribution of dynamic data
- Querying distributed and replicated data
- Cost model
- Self-configurable replication

Self-configurable replication

- Improving performances of a given query:
  - Replicate some data (including service calls)
    - Replicate the service definitions and the data used by them
- Recursively

Details of Cost-based algorithm in the paper

Implementation status

- AXML environment:
  - SUN's Java JDK 1.4 (includes XML parser, XPath processor, XSLT engine)
  - Apache Tomcat 4.0 servlet engine
  - Apache Axis SOAP toolkit 1.0 beta 3
  - X-OQL query processor, persistent DOM repository
  - JSP-based user interface, using JSLT 1.0 standard tag library
- AXML D&R-specific (ongoing):
  - A collaborative distributed workspace demo (VLDB03)
  - A light AXML peer for CLDC devices (kXML-RPC, XPath proxy)
- AXML is online:
  http://www-rocq.inria.fr/verso/Gemo/Projects/axml/

Conclusion & future work

- Generic framework for peer-to-peer management of dynamic documents
  - Focus on dynamic aspects
  - Context: AXML
- Work ongoing
  - Optimal query split
  - Collaborative index construction

Grazie

References

- (if people ask)
- [ABM01] S.Abiteboul, O.Benjelloun, T.Milo. "", FMII workshop, 2001.
- [ABM+02] S.Abiteboul, O.Benjelloun, I.Manolescu, T.Milo and R.Weber "", in "Web Dynamics", 2003.
- [MAA+03] T.Milo, S.Abiteboul, B.Amann, O.Benjelloun and F.Dang Ngoc. "Exchanging Intensional XML Documents", SIGMOD 2003
- [XLink] http://www.w3c.org/xlink
- [LDAP] O.Kapitskaia, R. Ng and D. Srivastava. “Evolution and revolutions in LDAP directory caches”, EDBT, 2000.

Related work

- LDAP, XLink, query/data shipping