Loop-Lifted XQuery RPC with Deterministic Updates

Peter Boncz

Ying Zhang

Outline

- Goals
- The XRPC language extension
- Loop-lifted Implementation of XRPC
- Deterministic distributed updates with XRPC
- Related work
- Conclusion

Goals

- Distributed XML DBMS or even P2P XML DBMS

- minimal while orthogonal extension of XQuery
- support for XML Updates (W3C XUF)
- interoperability with Service Oriented Architectures

- (performance, scalability)

XRPC Syntax

Syntax extension:

`execute at {` *Expr* `} {` *FunApp*(*ParamList*) `}`

URI string

New grammar rule:

```
PrimaryExpr ::= ... | FunctionCall | XRPCCall | ...

XRPCCall ::= "execute at" "{" ExprSingle "}" "{ FunctionCall "}"

FunctionCall ::= Qname "(" (ExprSingle ("," ExprSingle)*)? ")"
```

Example

A single XRPC call

```
import module namespace film="filmdb"
            at "http://x.org/film.xq"
<films>
  execute at {"http://y.org"}
    {film:filmsByActor("Sean Connery")}
</films>
```

Query result

```
<films>
  <filmName>The Rock</filmName>
  <filmName>Goldfinger</filmName>
</films>
```

filmDB.xml@y.org

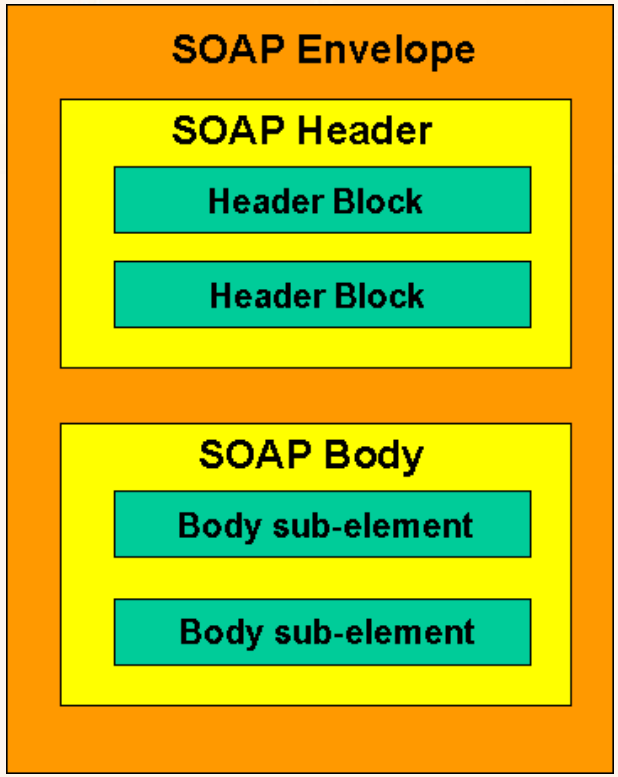
```
<filmDB>
  <film>
    <filmName>The Rock</filmName>
    <actorName>Sean Connery</actorName>
  </film>
  <film>
    <filmName>Goldfinger</filmName>
    <actorName>Sean Connery</actorName>
  </film>
  <film>
    <filmName>Green Card</filmName>
    <actorName>Gerard Depardieu</actorName>
  </film>
</filmDB>
```

film.xq

```
module namespace film="filmdb";
declare function film:filmsByActor
            ($actor as xs:string) as node()*
{
  doc("filmDB.xml")//filmName
                      [../actorName=$actor]
};
```

The SOAP Protocol

- Simple Object Access Protocol
- XML-based protocol over HTTP
- Commonly used by web service applications
- Interoperability
- A natural choice as the underlying protocol of XRPC
  - XML is ideal for distributed environment
  - Easy to be processed by an XQuery engine
  - Passing values of all XDM data type



SOAP XRPC Message Format

Full support for XML Data Model

```
<?xml version="1.0" encoding="utf-8"?>
<env:Envelop
  xmlns:xrpc="http://monetdb.cwi.nl/XQuery"
  xmlns:env=".../2003/05/soap-envelope"
  xmlns:xs=".../2001/XMLSchema"
  xmlns:xsi=".../2001/XMLSchema-instance"
  xsi:schemaLocation=
     "http://monetdb.cwi.nl/XQuery
      http://monetdb.cwi.nl/XQuery/XRPC.xsd">
<env:Body>
<xrpc:request
  xrpc:module="filmdb"
  xrpc:method="filmsByActor"
  xrpc:location="http://x.org/film.xq">
<xrpc:call>
<xrpc:sequence>
<xrpc:atomic-value xsi:type="xs:string">
  Sean Connery
</xrpc:atomic-value>
</xrpc:sequence>
</xrpc:call>
</xrpc:request>
</env:Body>
```

```
<xrpc:response
  xrpc:module="filmdb"
  xrpc:method="filmsByActor">
 <xrpc:sequence>
  <xrpc:element>
    <filmName>The Rock</filmName>
  </xrpc:element>
  <xrpc:element>
    <filmName>Goldfinger</filmName>
  </xrpc:element>
 </xrpc:sequence>
</xrpc:response>
```

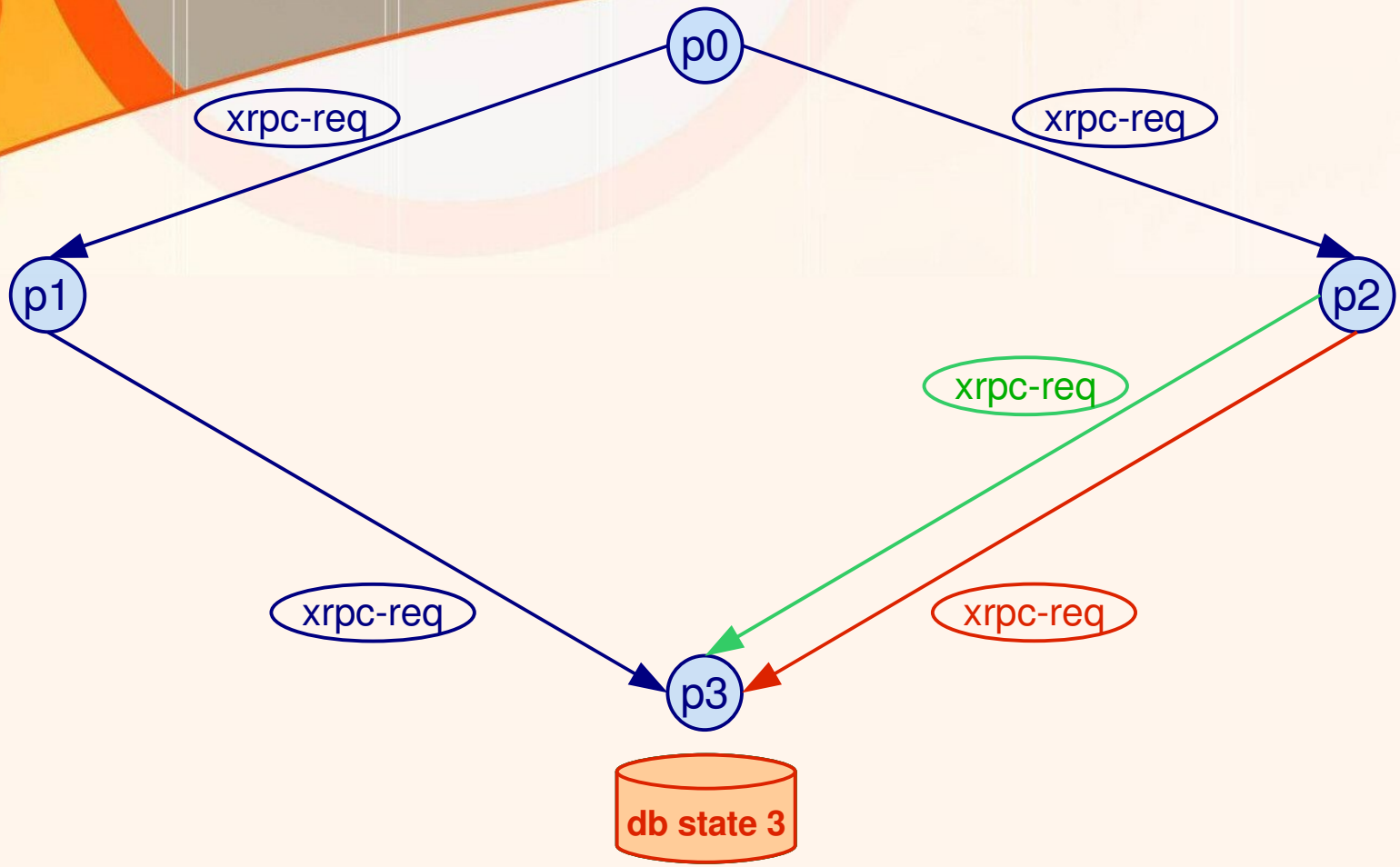
Basic Non-updating XRPC Call

$$\frac{\begin{array}{c} send_{p_0 \to p_x} request(m, f, ParamList) \\ db^{now} @ p_x \vdash f(ParamList) \Rightarrow val \\ send_{p_x \to p_0} reply(val) \end{array}}{db @ p_0 \vdash f(ParamList) @ p_x \Rightarrow val} \quad (\mathcal{R}_{\mathcal{F}_r})$$

Basic Updating XRPC Call

$$\frac{\begin{array}{c} send_{p_0 \to p_x} request(m, f, ParamList) \\ db @ p_x \vdash f(ParamList) \Rightarrow \Delta_f \\ db @ p_x := \texttt{commit}(db @ p_x, \Delta_f) \\ send_{p_x \to p_0} reply() \end{array}}{db @ p_0 \vdash f(ParamList) @ p_x \Rightarrow ()} \quad (\mathcal{R}_{\mathcal{F}_u})$$

Nested and Concurrent XRPC Calls



Some applications need isolation

XRPC Formal Semantics with Isolation

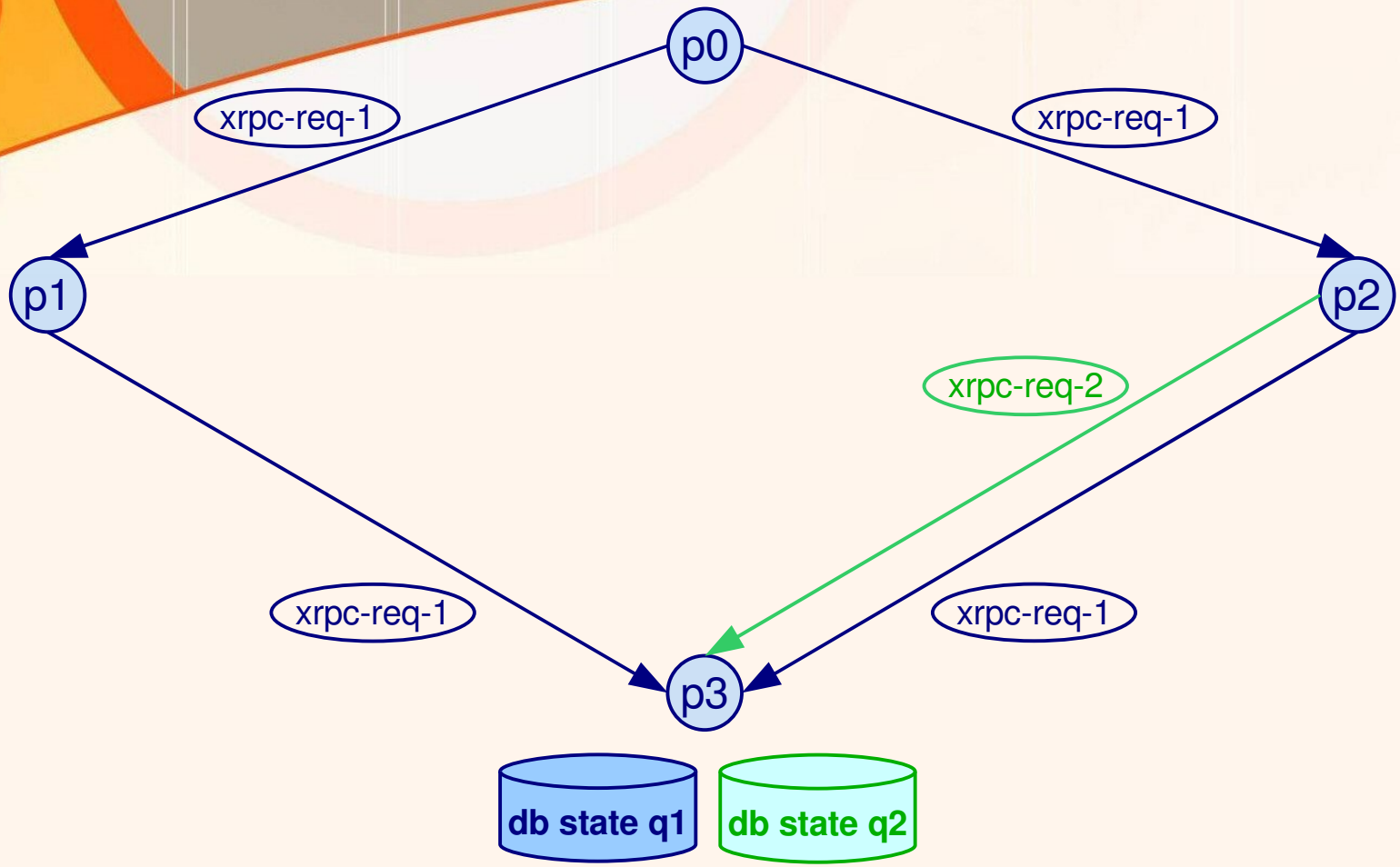
Non-updating XRPC Call with Repeatable-Reads:

$$\frac{\begin{array}{c} send_{p_0 \to p_x} request(q, m, f, ParamList) \\ db_q@p_x \vdash f(ParamList) \Rightarrow val \\ send_{p_x \to p_0} reply() \end{array}}{db_q@p_0 \vdash f(ParamList)@p_x \Rightarrow val} \quad (\mathcal{R}_{\mathcal{F}'_r})$$

Updating XRPC Call with Isolation:

$$\frac{\begin{array}{c} send_{p_0 \to p_x} request(q, m, f, ParamList) \\ (db_q@p_x, \Delta_q@p_x) \vdash f(ParamList) \Rightarrow \Delta_f \\ \Delta_q@p_x := \texttt{mergeUpdates}(\Delta_q@p_x, \Delta_f) \\ send_{p_x \to p_0} reply() \end{array}}{(db_q@p_0, \Delta_q@p_0) \vdash f(ParamList)@p_x \Rightarrow ()} \quad (\mathcal{R}_{\mathcal{F}'_u})$$

Nested and Concurrent XRPC Calls with Isolation



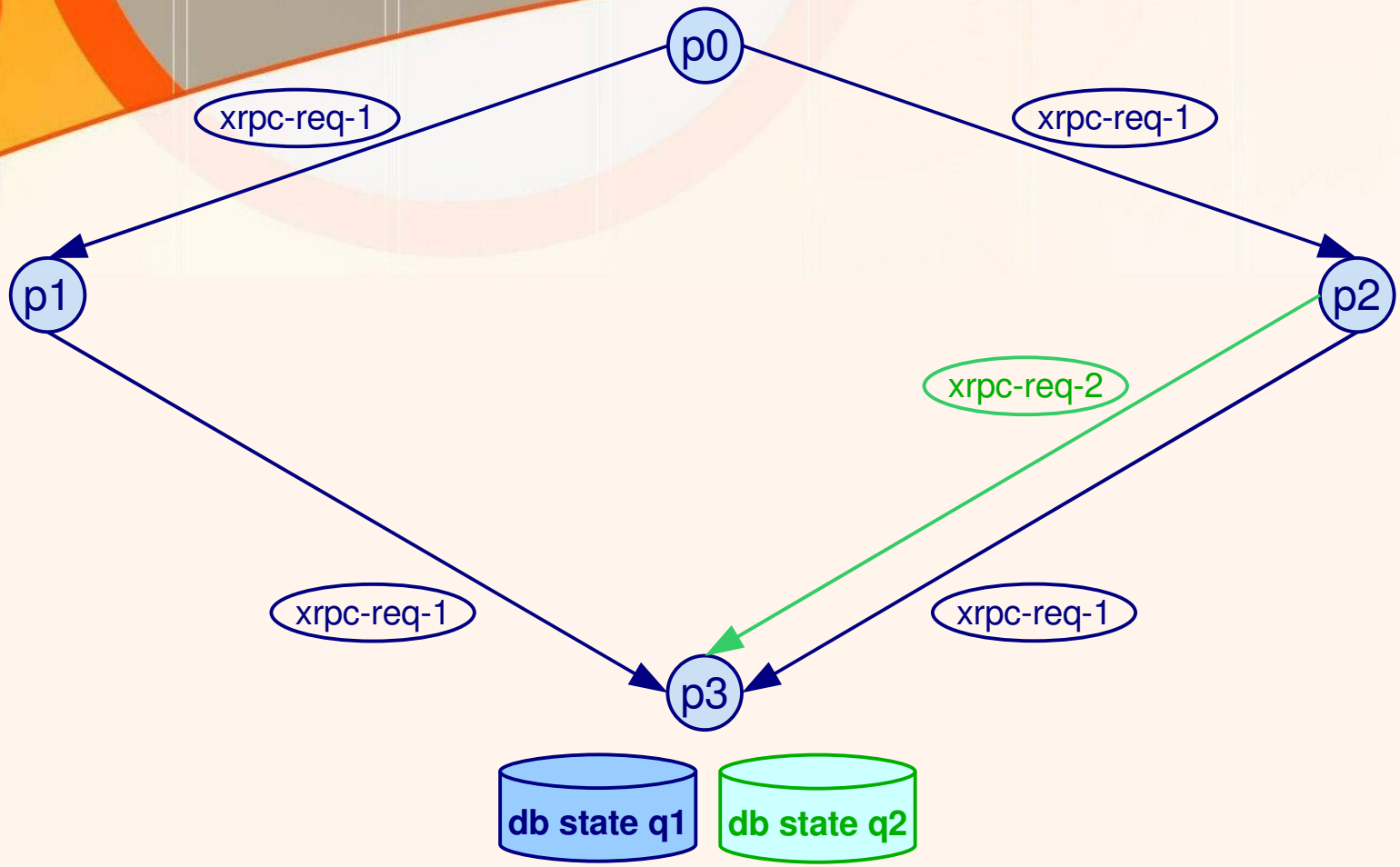
SOAP XRPC Message Format


```
<xrpc:request  xrpc:module  ="filmdb"
               xrpc:method  ="filmsByActor"
               xrpc:location="http://x.org/film.xq">

  <queryID host="y.org" timestamp="123456" timeout="180"/>

  <xrpc:call>
    <xrpc:sequence>
      <xrpc:atomic-value xsi:type="xs:string">
        Sean Connery
      </xrpc:atomic-value>
    </xrpc:sequence>
  </xrpc:call>
</xrpc:request>
```

Nested and Concurrent XRPC Calls with Isolation



Outline

- Goals
- The XRPC language extension for XQuery
- ***Loop-lifted Implementation of XRPC***
- Deterministic distributed updates with XRPC
- Related work
- Conclusion

- A pure relational XML DBMS
- Loop-lifting technique

  $\Rightarrow$ Efficient ***Bulk*** operation plans

- P. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger and J. Teubner. *MonetDB/XQuery: A Fast XQuery Processor Powered by a Relational Engine.* In *SIGMOD*, June 2006.

Loop-Lifted SOAP XRPC Message


```
import module namespace film="filmdb"
                    at "http://x.org/film.xq"

for $actor in ("Julie Andrews", "Sean Connery")
  return
    execute at "y.org" xquery film:filmsByActor($actor)
```

```
<xrpc:request location="http://x.org/film.xq"
              module="filmdb" method="filmsByActor">
  <xrpc:call>
    <xrpc:sequence>
      <xrpc:atomic-value xsi:type="xs:string">
        Jolie Andrews
      </xrpc:atomic-value>
    </xrpc:sequence>
  </xrpc:call>
  <xrpc:call>
    <xrpc:sequence>
      <xrpc:atomic-value xsi:type="xs:string">
        Sean Connery
      </xrpc:atomic-value>
    </xrpc:sequence>
  </xrpc:call>
</xrpc:request>
```

Bulk RPC

Performance


```
for $y in  1 to $x
   execute at { “foo.org” } { add($y, $y) }
```

	$x=1	$x=1000
one-at-a-time	35	3497
bulk	35	400

XRPC time of in msec

Outline

- Goals
- The XRPC language extension for XQuery
- Loop-lifted Implementation of XRPC
- ***Deterministic distributed updates with XRPC***
- Related work
- Conclusion

XQuery Update Facility

- Defines update actions
  - insert into, insert as first|last, insert before|after
  - delete, replace, rename, ...
- Update actions are stored in Pending Update List
- At commit time, the pending update list is applied
- **order of update action matters; we use the intuitive order**
- **(W3C XUF Draft actually leaves this open)**

“Julie”-s First or “Connery”-s First?


```
import module namespace film=”filmdb”
                  at “http://x.org/film.xq”

for $name in (“Julie”, “Sean”)

  let $connery := concat($name, “ Connery”)
  let $andrews := concat($name, “ Andrews”)
  return
      (execute at “http://y.org” xquery film:insertLog($connery),
       execute at “http://y.org” xquery film:insertLog($andrews))
```

Desired order:

```
1.1 insertLog( “Julie Connery”)
1.2 insertLog( “Julie Andrews”)

2.1 insertLog( “Sean Connery”)
2.2 insertLog( “Sean Andrews”)
```

Loop-lifted order:

```
1.1 insertLog(“Julie Connery” )
2.1 insertLog( “Sean Connery” )

1.2 insertLog(“Julie Andrews” )
2.2 insertLog( “Sean Andrews” )
```

Distributed Pending Update Table


```
import module namespace film="filmdb"
                at "http://x.org/film.xq"

for $name in ("Julie", "Sean")
  let $connery := concat($name, " Connery")
  let $andrews := concat($name, " Andrews")
  return
      (execute at "http://y.org" xquery film:insertLog($connery),
       execute at "http://y.org" xquery film:insertLog($andrews))
```

$\mathcal{P}$	$\mathcal{A}$	$\mathcal{T}$
"http://y.org"	insertLog("Julie Connery")	1.1.1
"http://y.org"	insertLog("Sean Connery")	1.2.1

fid_1@"y.org"

$\mathcal{P}$	$\mathcal{A}$	$\mathcal{T}$
"http://y.org"	insertLog("Julie Andrews")	1.1.2
"http://y.org"	insertLog("Sean Andrews")	1.2.2

fid_2@"y.org"

$$\Downarrow sort_{\mathcal{T}}(\Delta_q @ p_1)$$

$\mathcal{P}$	$\mathcal{A}$	$\mathcal{T}$
"http://y.org"	insertLog("Julie Connery")	1.1.1
"http://y.org"	insertLog("Julie Andrews")	1.1.2
"http://y.org"	insertLog("Sean Connery")	1.2.1
"http://y.org"	insertLog("Sean Andrews")	1.2.2

Extension of SOAP XRPC Message


```
<xrpc:request location="http://x.org/film.xq"
              module="filmdb" method="insertLog">
  <queryID host="a.org" timestamp="123456" timeout="180"/>

  <xrpc:call tag="1.1.1">
    <xrpc:sequence>
      <xrpc:atomic-value xsi:type="xs:string">
        Jolie Connery
      </xrpc:atomic-value>
    </xrpc:sequence>
  </xrpc:call>

  <xrpc:call tag="1.2.1">
    <xrpc:sequence>
      <xrpc:atomic-value xsi:type="xs:string">
        Sean Connery
      </xrpc:atomic-value>
    </xrpc:sequence>
  </xrpc:call>

</xrpc:request>
```

Related Work

- XQueryD
  - `ExprSingle` ::= “`execute at`” <URL>
    [“`xquery`” “`{`“ `ExprSingle` “`}`” | ...]
  - A runtime rewriter scans and substitute variables
- Galax Yoo-Hoo
  - Map between XQuery functions to/from WSDL
  - Stub is XQuery UDF
  - *One* destination for each imported web service module
- ActiveXML
  - Service calls, `sc`, embedded in XML documents
  - Result XML fragment is inserted as a sibling of the `sc`
  - *Lazy* evaluation model
  - *Dynamic* documents
- DXQ (Distributed XML-Query network)
  - XQD (XML-Query Distributor) and XDP (XML-Document Provider)
  - Client sends query to XQD, all registered XDPs exec. the query
  - An HTTP-like communication protocol

Conclusion

XRPC provides a simple and effective basis for P2P XQuery processing

See also: www.cwi.nl/~boncz/xrpc.pdf

Future work

- scalability with more peers (dealing with failures)
- standardization of distributed extension
- distributed update transactions (need 2PC)
- coupling with P2P data structures