XPointer and HTTP Range

A possible design for a scalable and extensible RDF Data Access protocol.

Bryan Thompson
4-21-2004 - **draft**

Presented to the RDF Data Access Working Group

XPointer and HTTP today

- The URI fragment identifier is NOT passed with a normal HTTP request.
- Therefore, the client must GET the entire representation and *then* applies an XPointer processor to interpret the fragment identifier.
- So, extensible, but not scalable.

Normal HTTP Request & Response

http://www.myorg.org/myTripleStore

Request:

```
GET /myTripleStore HTTP/1.1
Host: www.myorg.org
Accept: application/rdf+xml
```

Response:

```
HTTP/1.1 200 Ok
Content-Type: application/rdf+xml

<rdf:RDF … />
```

This request asks for the **entire** triple store, serialized as RDF/XML.

W3C XPointer Framework 1.0

- Extensible processing model for URI fragment identifiers
    - #foo
    - #element(foo/2)
    - #svgView(0,0,100,100)
    - #xmlns(x=www.myorg.org)x:my-rdf-query(…)
- Each XPointer scheme is has its own QName.
- W3C schemes are in the default namespace.
- Multiple pointers can be specified and are evaluated left to right until a match is found.

- #rdf() – possible scheme this group could define.

The HTTP “Range” header

- The HTTP/1.1 protocol defines an extensible request header named “Range”
- The client specifies a “range-unit”, e.g.,

    “xpointer”

  and a “range-value”, e.g.,

    “rdf(…)”
- The server sends back only the identified sub-resources (triples) for the negotiated content type (or a status code indicating an appropriate error).
- So, this looks like ….

RDF data access w/ HTTP Range

```
GET /myTripleStore HTTP/1.1
Host: www.myorg.org
Accept: application/rdf+xml
Range: xpointer = rdf(
    SELECT (?x foaf:mbox ?mbox)
    WHERE (?x foaf:name "John Smith") (?x foaf:mbox ?mbox)
    USING foaf FOR <http://xmlns.com/foaf/0.1/>
    )
```

```
HTTP/1.1 206 Partial Content
Content-Type: application/rdf+xml

<!– Only the selected sub-graph is transmitted to the client. -->
<rdf:RDF … />
```

Pros and Cons.

+1 Linkable

- XPointer expression is just the URI fragment identifier, e.g.,
  - http://www.myorg.org/myTripleStore#rdf(...)
- XPointer specifies how to encode the RDF query into the URI.

+1 Scalable

- Only identified sub-resources (triples) are transmitted to the client.

+1 Extensible

- DAWG can define an XPointer scheme for RDF, e.g., rdf().
- Applications can define their own RDF query schemes.
- Client can choose which query language to use.
- Same protocol can be used for graph update.

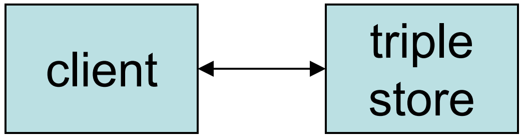
+1 Negotiable

- Client can request RDF/XML, N3, etc.
- Client can request compressed representation (GZip, etc).

Pros and Cons

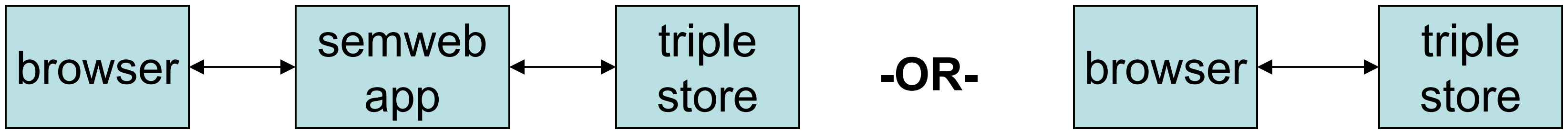
+1 Low implementation burden

- Machine clients are fine, since this is easy with nearly any HTTP client library.
- Server implementation burden for protocol is small using existing HTTP platforms.



Browser Scenarios

- Should a semantic web application expose raw RDF or a purpose built view?
- Bookmarks and links depend on GET doing the right thing:
  - Browser must negotiate for an RDF MIME Type.
  - XPointer works if browser sets the Range header.
- Using the query string implies that the RDF query is constructed while navigating a web application.
  - Will query parameters that work for the web application also work for an RDF query language?
  - XPointer will not interfere with these applications (orthogonal interface).
- GET with a query string is probably the only option if standardized data access with today's browsers from a bookmark is a paramount requirement.



Backing materials

Service Advertisement

```
HEAD /myTripleStore HTTP/1.1
Host: www.myorg.org
Accept: application/rdf+xml
```

```
HTTP/1.1 204 No Content
Content-Type: application/rdf+xml
Accept-Ranges: xpointer
Allow: HEAD, GET, ….
```

Pros and Cons.

Does not use the query string:

- +1 Lots of applications already define a query interface. XPointer will not interfere with these existing interfaces, or even with other XPointer schemes.
- +1 Very long query expressions are Ok since query is sent using an HTTP request header
  - Very long URI query strings might also be Ok, but there have historically been length limit issues.
- -1 Browsers can’t do this unless they implement the protocol extension (by mapping the fragment identifier onto the HTTP “Range” request header).
  - A protocol using POST won’t work from a bookmark either.

What is a “sub-resource” for RDF?

- Probably a triple.
- Can be represented in many notations, e.g., RDF/XML, N3, etc.
- Client can request a specific notation (content negotiation).
- Closure – In order for the response to be well-formed in a given notation, the response will need to be a set of triples, not just URIs or literals.

- These are also query language design issues.

XPointer specifies encoding for URI

- The XPointer Framework[1] specifies how a given XPointer expression must be encoded before it may be placed into an external form as the fragment identifier of a URI.
- So, we can choose *any* RDF query notation and the XPointer Framework will tell us how to represent that as the fragment identifier for a URI.

- rdf() – possible name for an XPointer scheme that this group could define triple store data access query language.

- [1] http://www.w3.org/TR/xptr-xpointer/

Pros and Cons

? Intermediaries

- Impact on existing intermediaries should be evaluated (in the field).

? Caching

- Servers can use the Vary header to indicate to caches that they may use a given response for all requests with the same Range header field.
- Caching will be an issue if/when we consider graph updates since there are likely to be multiple queries that select the same set of triples.