



eduserv



April 2006

# Harvest Functionality

**Augmenting interoperability across scholarly repositories**

Andy Powell, Eduserv Foundation  
[andy.powell@eduserv.org.uk](mailto:andy.powell@eduserv.org.uk)  
[www.eduserv.org.uk/foundation](http://www.eduserv.org.uk/foundation)

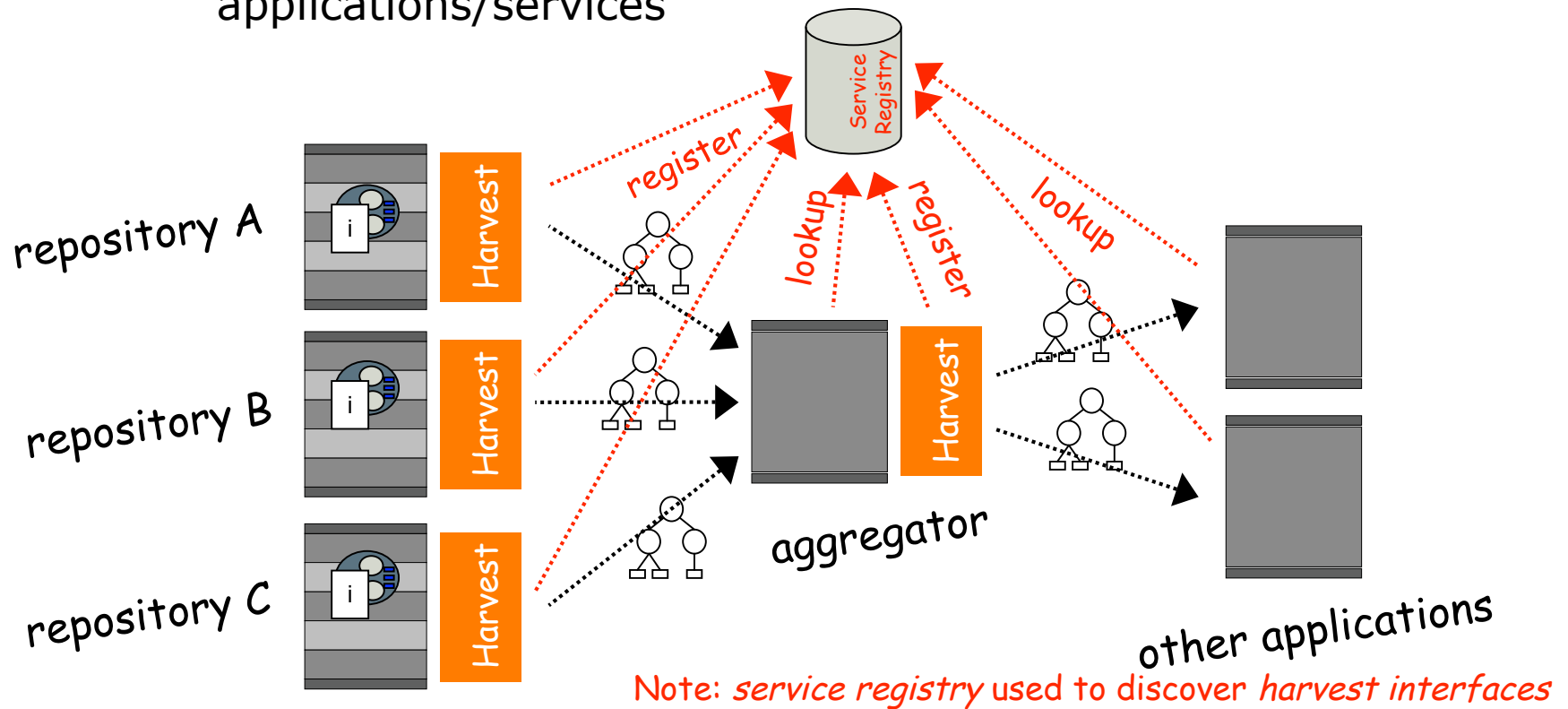


## Harvest - Functional requirement

- to allow applications/services on the network to regularly harvest *surrogates* of the *digital objects* held in remote *repositories*
- solution optimised for repeated and regular harvesting
- solution generic enough to be offered by wide range of heterogeneous *repositories* (scholarly publications, data, learning objects, images, etc.) in order to support the development of consistent external applications/services across different *digital object* types
- note: applications/services are expected to subsequently request services on the *digital objects* in the *repository* (including their *datastreams*) using the *obtain interface*, based on identifiers in the *surrogate*
- **Question:** is this separation of *harvest* vs. *obtain* correct?

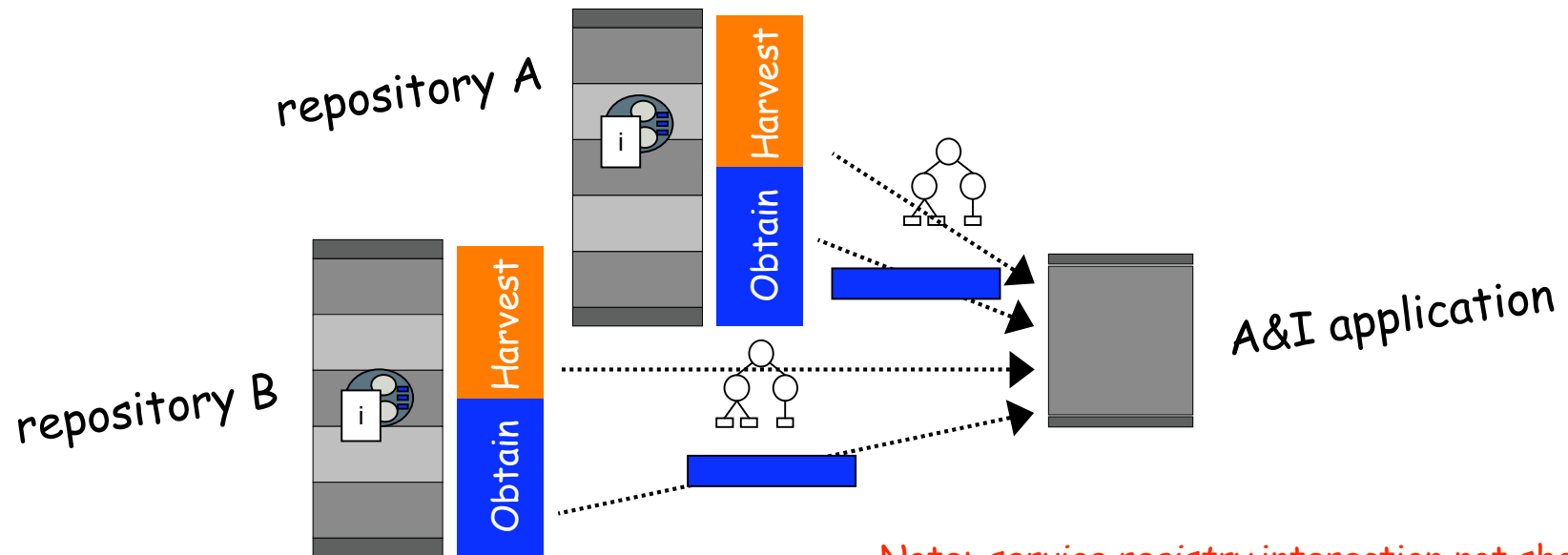
# Harvest - Scenarios/Use cases

- aggregator
  - a service that gathers *surrogates* from multiple *repositories* and exposes them for harvesting by other applications/services



# Harvest - Scenarios/Use cases

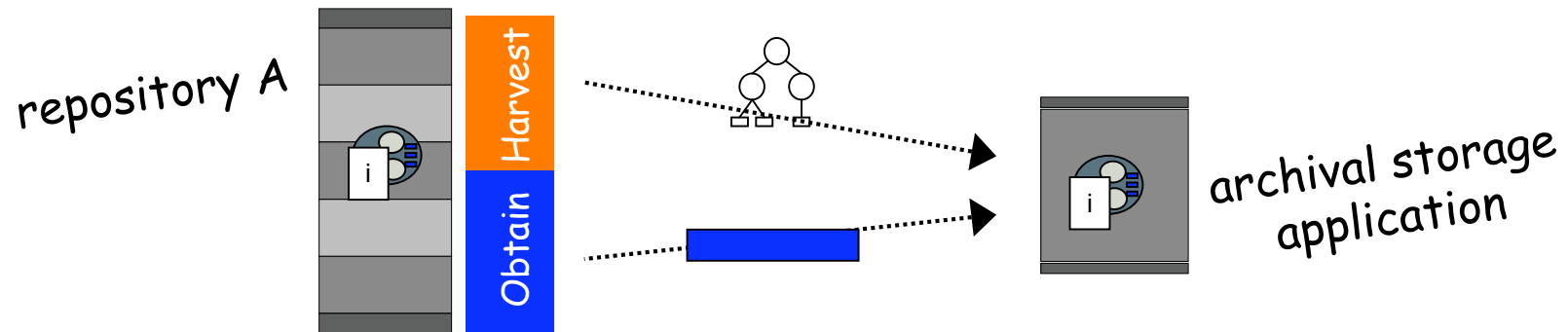
- abstracting and indexing
  - an application that gathers *surrogates* from multiple scholarly *repositories* and requests the associated *datastreams* and bibliographic metadata using the *obtain interface* – using the resulting knowledge to index, interlink and rank the available *digital objects*



Note: service registry interaction not shown

## Harvest - Scenarios/Use cases

- archival storage
  - a service that uses the *harvest interface* and *obtain interface* to regularly migrate copies of *digital objects* to an archival store with the aim of long-term preservation
- **Question:** can *surrogate* and *datastreams* be re-combined in this way to create a copy of the original *digital object*?



Note: *service registry* interaction not shown

## Abstract service/interface definition

- a service interface offered by a *repository* that exposes *surrogates* of the *digital objects* it contains for regular harvesting by other applications/services on the network
- data in
  - harvesting request (“BulkGet”, “ListIdentifiers”, “Get”) with optional parameters (“from”, “until”, “identifier”)
  - introspection request (“Explain”)
- data out
  - harvesting response (“list of surrogates”, “list of identifiers”, “single surrogate”)
  - introspection response (“repository/service/collection info”)
  - error message
- **Question:** does this capture what we want from a *harvest interface* in abstract terms?

## Abstract Interface Issues

- **Question:** does the *harvest interface* need 'Get' (i.e. harvest an individual *surrogate*) as well as 'BulkGet'?
  - 'Get' not really a "harvesting" request (discuss!)
  - however, evidence from deployment of OAI-PMH is that 'ListIdentifiers' followed by repeated 'GetRecord' is useful to implementers
  - if 'Get' is not required, then is 'ListIdentifiers' necessary?
  - if 'ListIdentifiers' and 'Get' are required, then need to decide if identifiers are for the *digital object*, the *surrogate* or something else
- **Question:** is the 'identifier' parameter for the *digital object*, the *surrogate* or something else?
- **Question:** do the 'from' and 'until' (datestamp) parameters pertain to the *digital object*, the *surrogate* or something else?

## Abstract Interface Issues (2)

- scholarly (and other) *repositories* will need to expose metadata about entities other than *digital objects* (e.g. physical people and abstract works), therefore not clear that '*digital object*' is the best label for the underlying entity. Is '*resource*' better?
- more generally, the "Architecture of the WWW" uses '*resource*' and '*representation*' approximately where we are using *digital object* and *surrogate*
- **Question:** do we need (or want) to use different terminology?

## Implementation options

- four implementation possibilities suggested here:
  - OAI-PMH
  - Atom
  - some combination of RSS, HTTP GET and content negotiation (e.g. Microsoft's Simple Sharing Extensions)
  - some combination of Google sitemaps, HTTP GET and content negotiation
- though only the first is considered in any detail (here)
- the use of OAI-PMH is in line with current digital library approaches...
- ...but the use of other technologies is arguably more in line with general Internet trends?
- **Questions:** are these a sensible set of implementation options? are there others?

## OAI-PMH issues

- the use of OAI-PMH will require a new 'metadata format' to carry *surrogates*
- it is worth noting the potential requirement for harvesting requests (i.e. 'Get') based on the *digital object* identifier – whereas current OAI-PMH requests (i.e. 'GetRecord') are made in terms of the 'item' identifier
- similarly, date-stamps in OAI-PMH pertain to the metadata 'record' (i.e. the *surrogate*) rather than to the 'object'
- it is also worth noting that the current OAI-PMH spec. is closely bound to HTTP in a couple of places and might benefit from more abstract definition
  - therefore, some work on the OAI-PMH spec. may be required whatever the outcome of this meeting

## OAI-PMH issues (2)

- one possibility is to consider moving OAI-PMH away from close ties to 'metadata harvesting'...
- i.e. make it generic enough to support harvesting of arbitrary 'representations' of *digital objects* (broadly in line with approach suggested in recent D-Lib papers)
- if so, OAI-PMH effectively becomes the OAI Harvesting Protocol
- OAI-HP could then be used to directly transfer *digital objects* and/or constituent *datastreams*
  - note: this results in a potential overlap with the *obtain interface*. Is that an issue?
- (obviously) any changes to the current OAI-PMH spec. will have to be mindful of backwards compatibility issues!