PIDs & DOs

Willem Elbers
The Language Archive
MPI for Psycholinguistics, Netherlands
Slides inspired by Daan Broeder and
Tobias Weigel

SEVENTH FRAMEWORK PROGRAMME

The Language Archive

EUDAT

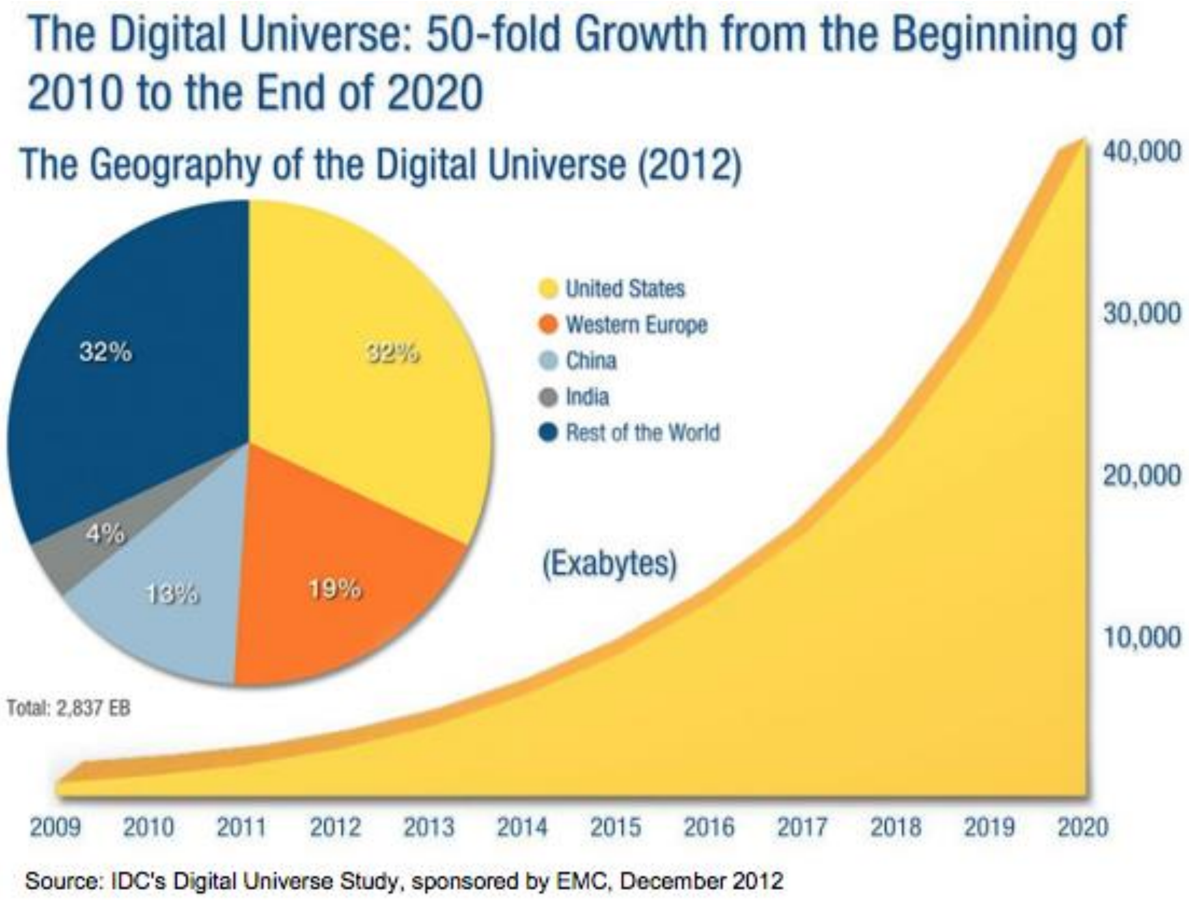
EUDAT Training, Rome, 28 October 2013

Contents

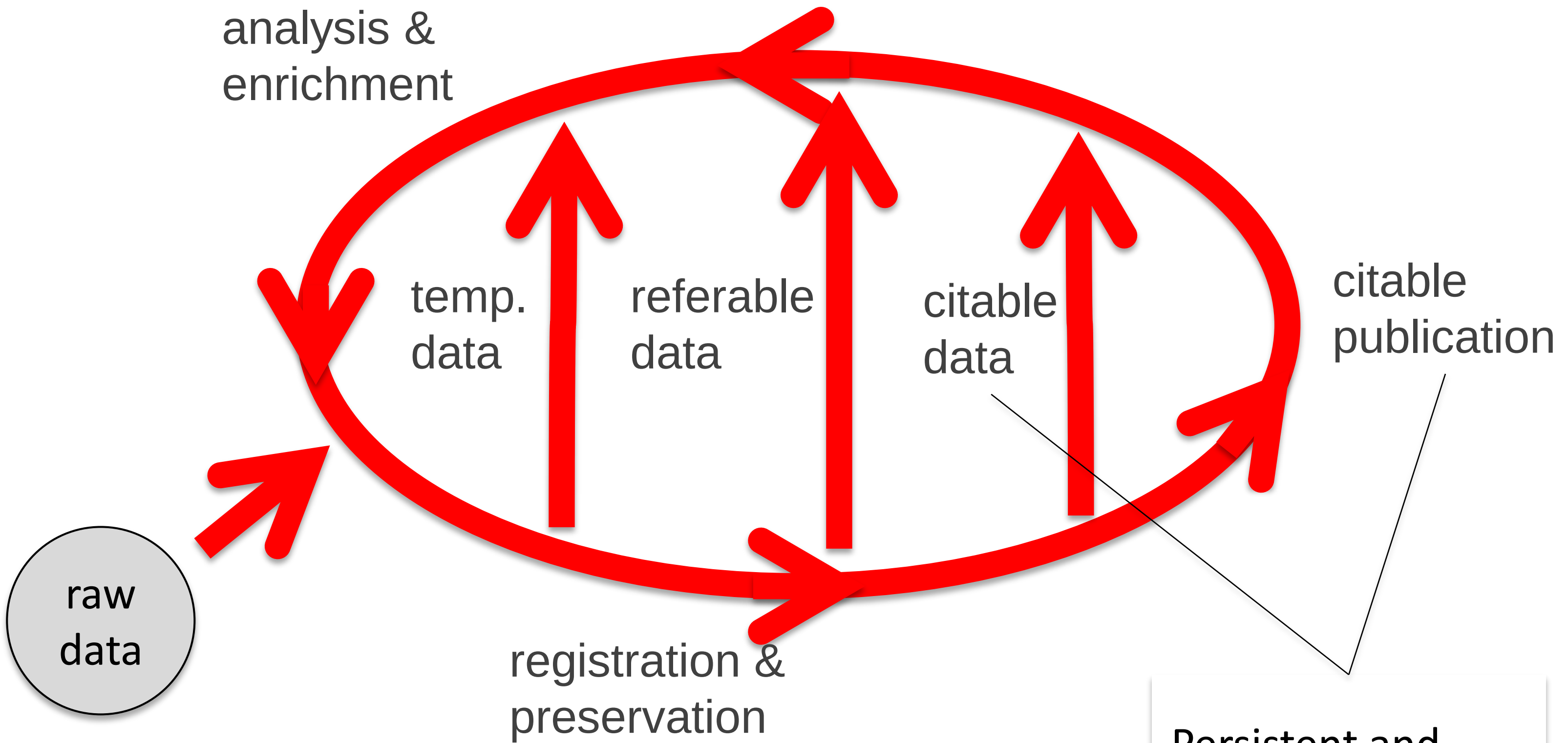
EUDAT

Data

- The increase of data volume also brings an increase in the number of data objects

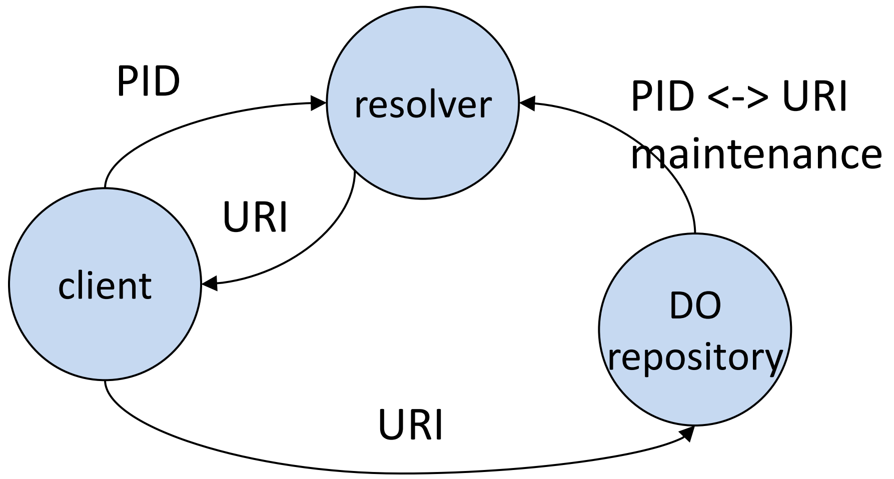


Data creation cycle



Persistent and
Robust
Identification
in global context

Referencing Resources



Standard we use URIs (URLs) for referencing resources. However: the resource is moved - host name change or file system changes

- Problem for embedded references inside the archive
- …but especially outside the archive
- Can be seen as an organizational problem
  - W3C proposes Cool-URIs
- But difficult to solve, hence the PID frameworks

PID Frameworks: PURL,HS, ARK

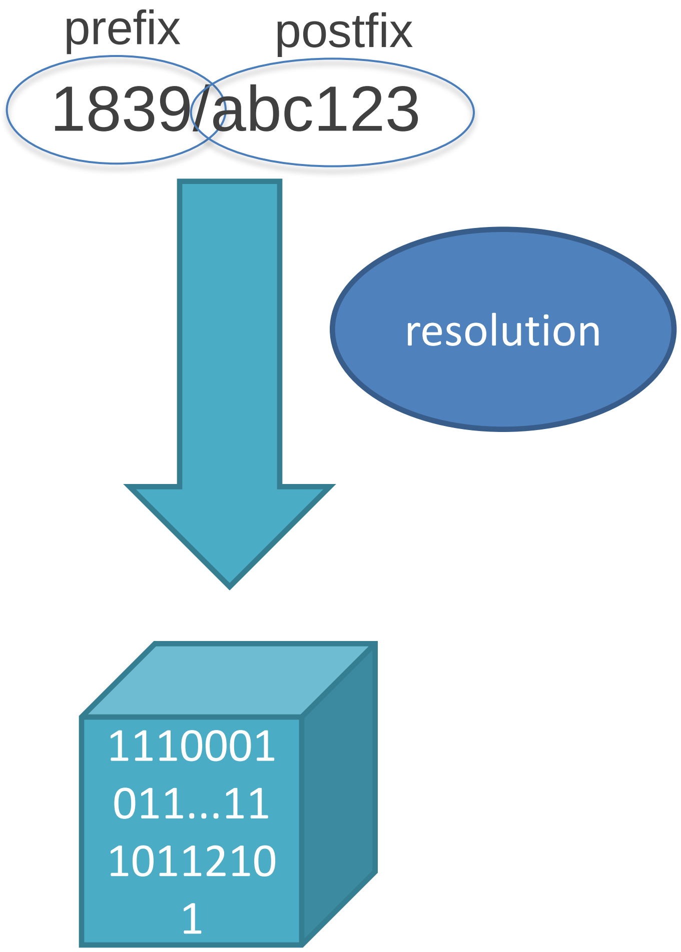
- Give every resource a unique persistent identifier: PID
- Every PID associated with one (or more) URLs
- Resolving process built into applications or available through plug-ins.

This comes at a cost:

- Added layer of infrastructure
- Must be managed
- Must run with high availability
- Must be very sure that this can be handled by our archives also in the long term.
- But can be used for extra services

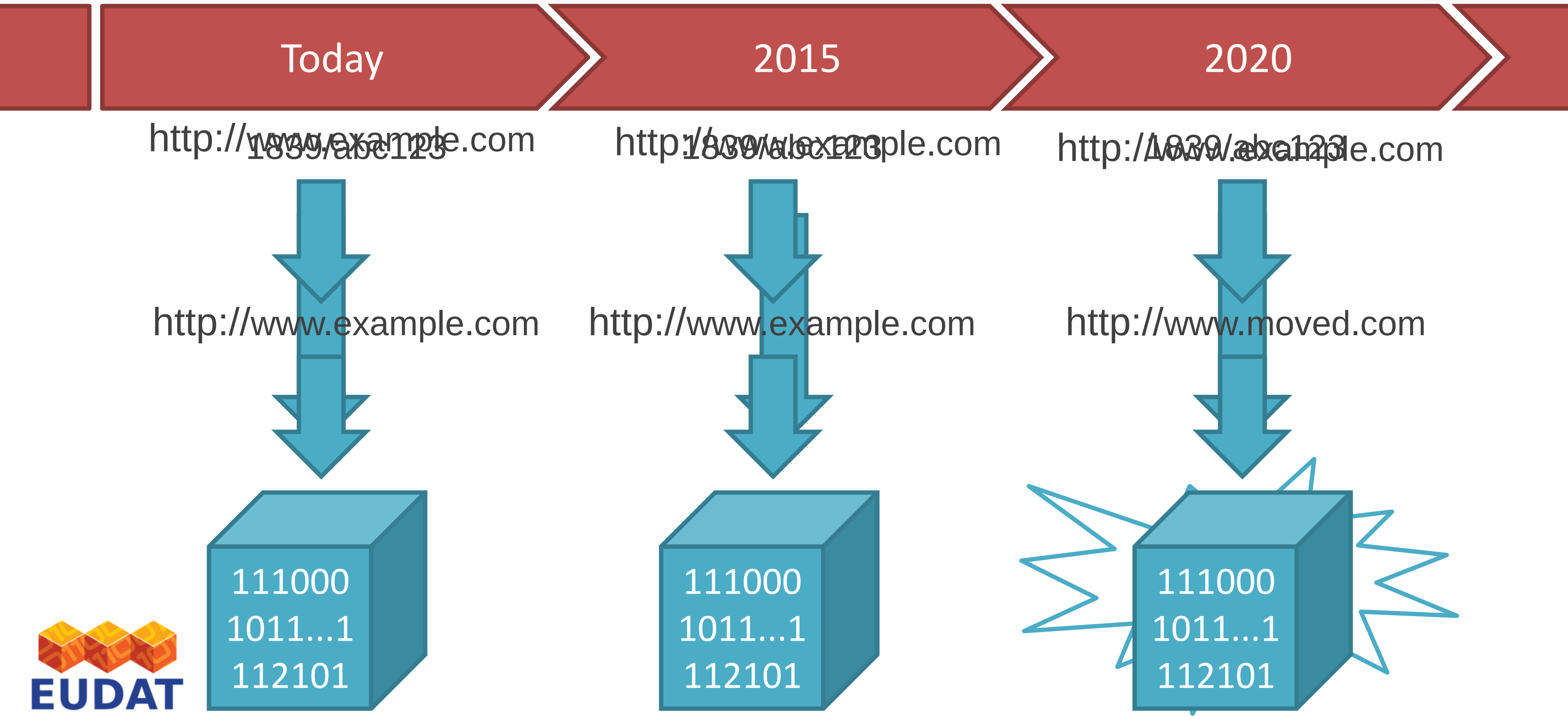
What is a persistent identifier?

- Identifier pointing to a resource
  - No knowledge of the resource
- Responsibility of the owner, identified by the prefix, to keep it up-to-date
- PIDs are globally unique

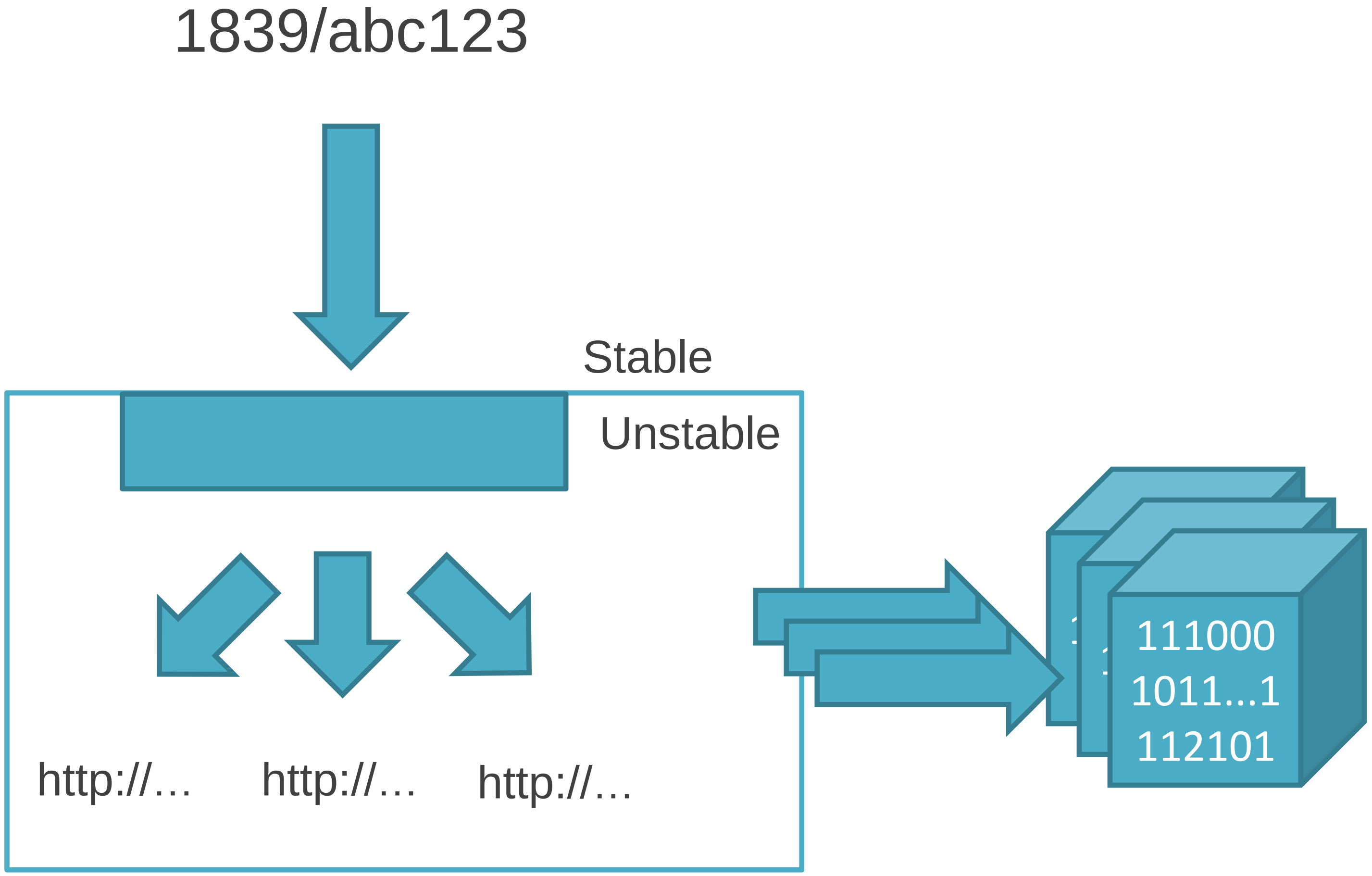


What is the problem?

- URLs have proven not to be stable over time: “Link rot”
- PIDs are stable over time



Redirection layer



PID requirements

- Attach multiple URLs to a PID
- Allow part identifiers for complex objects. Granularity issue.
- Allow attaching of extra data records to the PID (MD5 check,…)
- Actionable (URLified) PIDs
- HTTP proxy for resolving (use port 80 only)
- REST or SOAP interface for administration of PIDs from applications
- Secure, fine grained, administration
- Delegation of PID administration to other organizations
- Distributed, robust, highly-available, scalable
- No single-point of failure
- Acceptable non-commercial business model
- Control by user community

http://pidresolver.gwdg.de/mypid

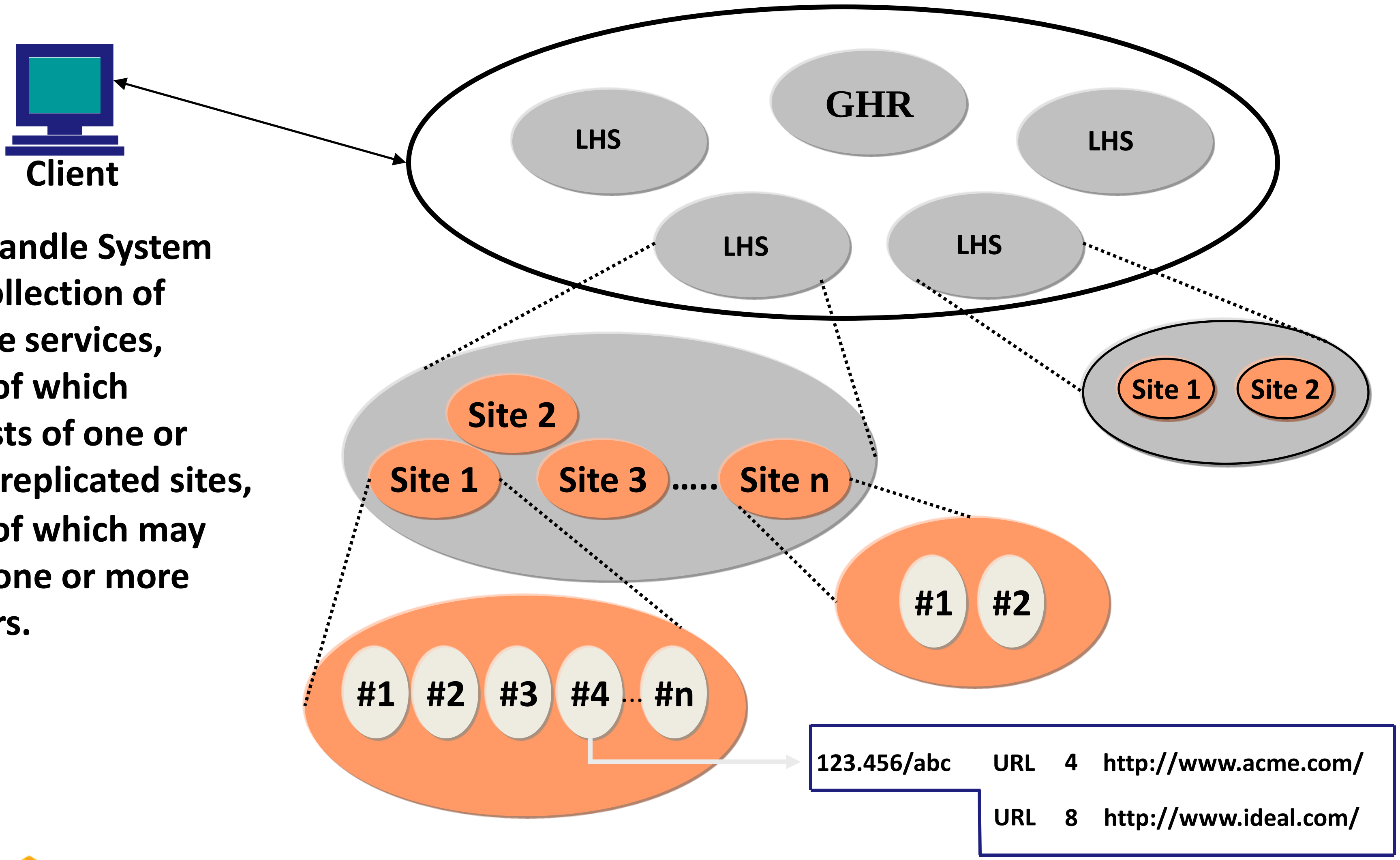
EUDAT

Handles Resolve to Typed Data

Handle	Data type	Index	Handle data
10.123/456	URL	1	http://acme.com/….
	URL	2	http://a-books.com/….
	DLS	9	acme/repository
	HS_ADMIN	100	acme.admin/jsmith
	XYZ	12	10011100111110

EUDAT

Handle Resolution



123.456/abc	URL	4	http://www.acme.com/
	URL	8	http://www.ideal.com/

The Handle System is a collection of handle services, each of which consists of one or more replicated sites, each of which may have one or more servers.

Shared PID service

- In principle possible for every repository to run its own full PID service
- but not every organization is willing or able to do that
- also there is an advantage of increased reliability by replicating services
- etc..

- DataCite & EPIC offer services based on HS for data PIDs since 2010
- Both offer APIs for creating and managing PIDs (handles)
- DataCite targeted to complete data-sets and includes also a specific metadata scheme for data-set publication
- EPIC targeted more at data management of individual resources allowing association of extra data with the PID: checksum, link to flexible metadata, …
- EPIC is only a steward for the PIDs, no lock-in
- There are more offers but status unsure: PERSID, ARK
- Some communities & infrastructures use several PIDs

EUDAT

Granularity

At what level of granularity do we issue PIDs for data?

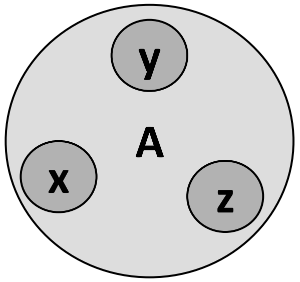
Some recommendations from CLARIN community

- An existing identifier scheme for a type of resources e.g. ISBN, suggests that level of granularity should be retained,
  - no new PIDs should be issued without very good reasons, such as for chapters. Those should addressed using **part identifiers**
- If the resource is associated with the complete content of a digital file, an individual PID should probably be assigned for this resource.
- If the resource is autonomous and exists outside a larger context, it deserves a PID
- If a resource should be citable apart from any containing resource, an individual PID should probably be assigned for this resource.

EUDAT

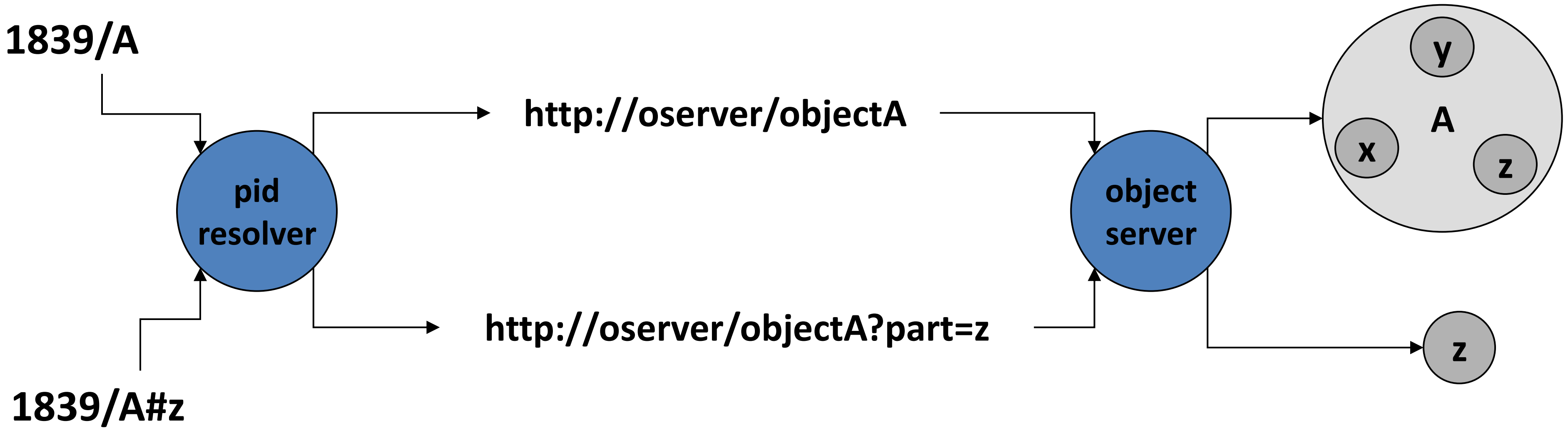
Part identifiers

1839/A
1839/A#x
...
1839/A#z



1839/A: 1839/A#x, 1839/A#y, 1839/A#z

- Wasteful to issue a pid for each part (think of 100k entries in a lexicon). So use part identifiers.
- Resolver can make an adequate translation
  "A#z" -> "objectA?part=z" This requires enough flexibility from the resolver to accommodate the object server.
- The syntax of "Z" should be standard for the specific data type. Loan from existing fragment identifier syntax standards.



PIDs and data architecture

DOs => multiple copies, versions, representations

PID is not only about access but also about identity

- DO copy: bitstream equality
- DO version: difficult to administrate
  - ARK syntax offers facilities for variants
- DO representation: HTTP content negotiation, difficult to administrate
- Pragmatic approach
  - PIDs should allow resolving + some tightly coupled metadata
  - Stay away from versioning policy (community specific)
  - But …

EUDAT

DOs and PIDs

Objects Should Wear Their Identifiers

A valuable technique for provision of persistent objects is to try to arrange for the complete identifier to appear on, with, or near its retrieved object. An object encountered at a moment in time when its discovery context has long since disappeared could then easily be traced back to its metadata, to alternate versions, to updates, etc

- PIDs are a registry
  - PID -> URL + metadata
- Text based resources allow embedding a PID (in the text e.g. ISBN
- How about binary files?
- Need resolving checksum to PID
  - Can be a service of PID service provider

To sum it up

- PIDs are a generic tool with clear boundaries
- Handles provide other useful features
- EUDAT is using the EPIC handle service

Thank you for your attention