N2T + ARK = a Viable Persistent Identifier Solution

27 March 2008

John Kunze, California Digital Library

What do we want from persistent identifiers?

1. Persistent *access* to data objects
   - No more “404 Not Found”
2. Persistent *reference*, meaning that the identifier will not be re-assigned to an unrelated data object
   - Failing access to the object, next best outcome is persistent access to its metadata
3. Trust that access will be to the “authentic object”
   - Across database versions and updates

Nice “dream”, but identifiers can’t help these at all!

- All ids completely at the mercy of service failure

What identifiers *cannot* do

An identifier is just a string, so *cannot protect you against any important threat*:

- Loss of political or financial support (e.g., bankruptcy)
- War, social upheaval, natural disaster
- Power outage, disk failure, human error
- Objects that are *removed*, *replaced*, or *moved* without a redirect being set up (HTTP redirects were always there, but few providers had the will to use them)

If identifier's are useless, why keep talking about them?

The identifier was the last link to our loss

- last "person to see the victim alive" becomes the prime suspect

But rather than blame the identifier, which is only a messenger to a set of services,

- Instead focus on those *services* and
- Eliminate the few minor screwups that are still possible with string assignment, such as opaqueness errors (eg, myopic branding), accidental re-assignment, public disclosure of internal ids, unrealistic expectation setting

The important threats…

What are you doing about them, and how much time/energy/money are you spending instead of distracting resolution systems?

Is it even possible that your persistent identifier system increases instead of decreases your risk?

- Do you pay to use your ids or their resolver?
- How large and complex a system is it to maintain and when it breaks how much money will you pay to fix it (compare to what you pay when bugs in Apache or DNS are fixed)?

What identifiers *can* do

An identifier is just a string; it can provide:

- "Words" giving clues about the *object's* alleged nature (brand, subject, year, etc.)
  - Short-term feature but long-term risk
- … and giving clues about the *identifier's* alleged nature (scheme label & structure)
- … and, if actionable, access to *services* that may provide things we actually want
  - To objects, metadata, policies, versions

String features by naming regime

Uniqueness: globally unique prefix plus unique name

Registry-based prefixes from

- DNS, Handle, DOI, URN, ARK, even UUID

Plus unique name

- Local practices assisted by NOID, UUID

Opacity for longevity (NOID, UUID)

Actionability (any scheme that is embeddable in URLs)

- Identity-inert hostnames for branding (ARK)

Structure that can disclose object versions & hierarchy (ARK)

Contains a check digit (NOID, ISBN)

String extends to address metadata and policy records (ARK)

Why do identifiers break?

Whatever the string, what matters is the *thing*; if the thing's unavailable, the formerly actionable identifier is broken

- Realistically, only URLs matter, where broken means either
  1. The hostname is broken

     Server down, gone, or renamed *

     Domain name lost, provider out of business or
  2. The pathname to the thing is broken

     Thing down, gone, or renamed *
- Global name resolvers to the rescue? NO!
  - No global fix for these*, only individual providers can fix
  - And a local resolver only helps for renamed objects (eg, HTTP redirects, or simple 2-column database mapping)

A name resolver can help fix only one small problem: hostname instability

Domain name lost, provider out of business

We can help this case

Large organizations less vulnerable (eg, is mpg.de, loc.gov, or bnf.fr less stable than doi.org?)

Branding: the comfort of not seeing your hostname vs. the comfort of *seeing* your hostname

Traditional solutions: PURL, URN, Handle, DOI

Solutions tied to special-purpose technology, sometimes complex and proprietary

N2T (Name-to-Thing)

N2T is two things at once

A consortium of cultural memory organizations

… and …

A small, ordinary web server, mirrored in several instances globally for reliability

Basic idea: protect 200 organizations' URLs from *hostname instability* with 200 rewrite rules

How: simple HTTP redirects, one per organization

Why Name-to-Thing (N2T)?

Challenge: current id resolver systems are risky, expensive, exclusionary, and incomplete

Their main contribution is hostname stability via a mechanism much the same as ordinary redirection

Name-to-Thing is safe, cheap, id scheme-agnostic, and it addresses a superset of the problem addressed by URN, DOI, & Handle

Origin: first presented at Digital Library Federation (DLF) panel on global id resolvers November 2006

Name-to-Thing (N2T) overview

Establish a consortium and a small web server

Each member publishes URLs under n2t.info:

http://n2t.info/12345/foo/bar.zaf

…which redirects to member 12345's server

The URL is *protected* from local server name instability (but not from local server stupidity)

N2T registers member number + local server URL

- 200 members supported by 200 rewrite rules
- 4-6 members volunteer host mirrored N2T servers

N2T – user point of view

Each consortium member organization gets a unique number, such as, 12345.

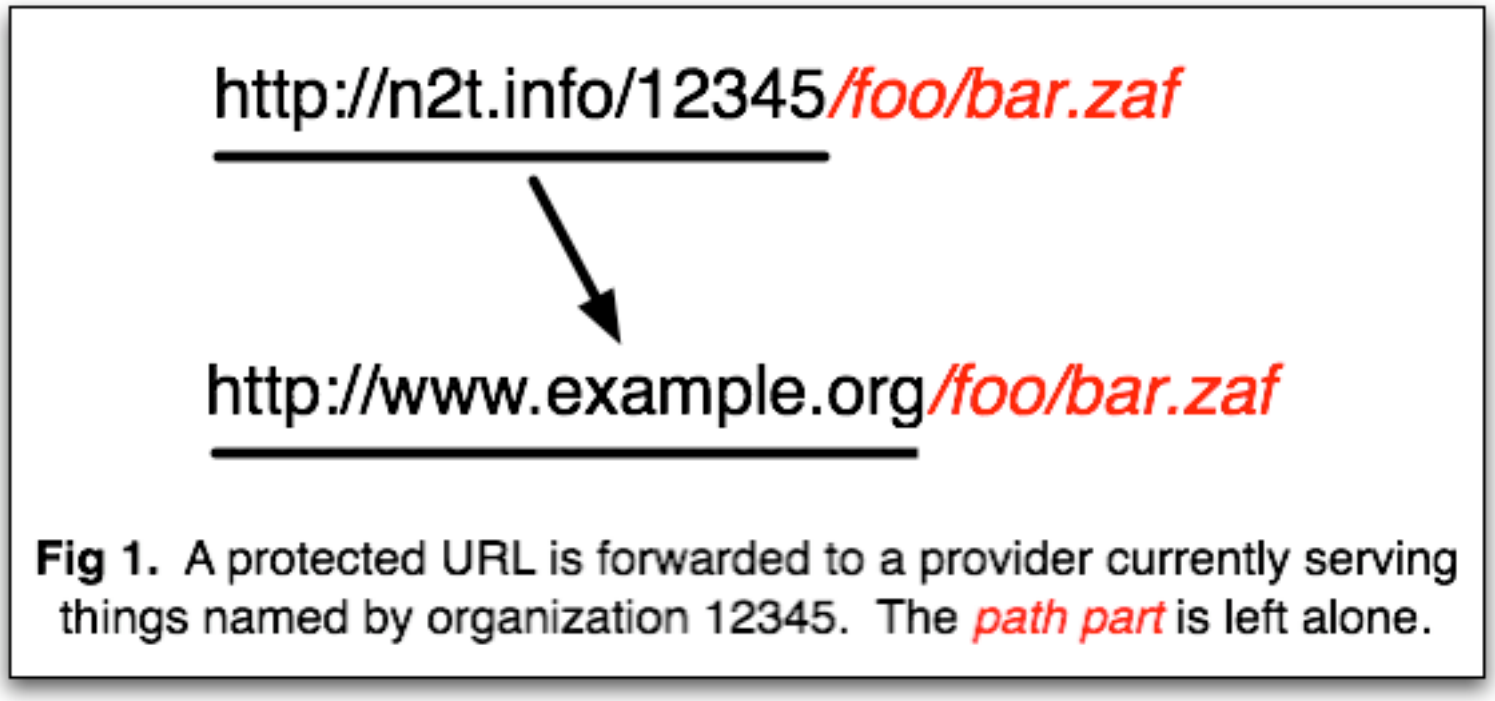


Fig 1. A protected URL is forwarded to a provider currently serving things named by organization 12345. The *path part* is left alone.

N2T – system point of view

Technically, resolution (access to a thing given its name) is simple redirection.

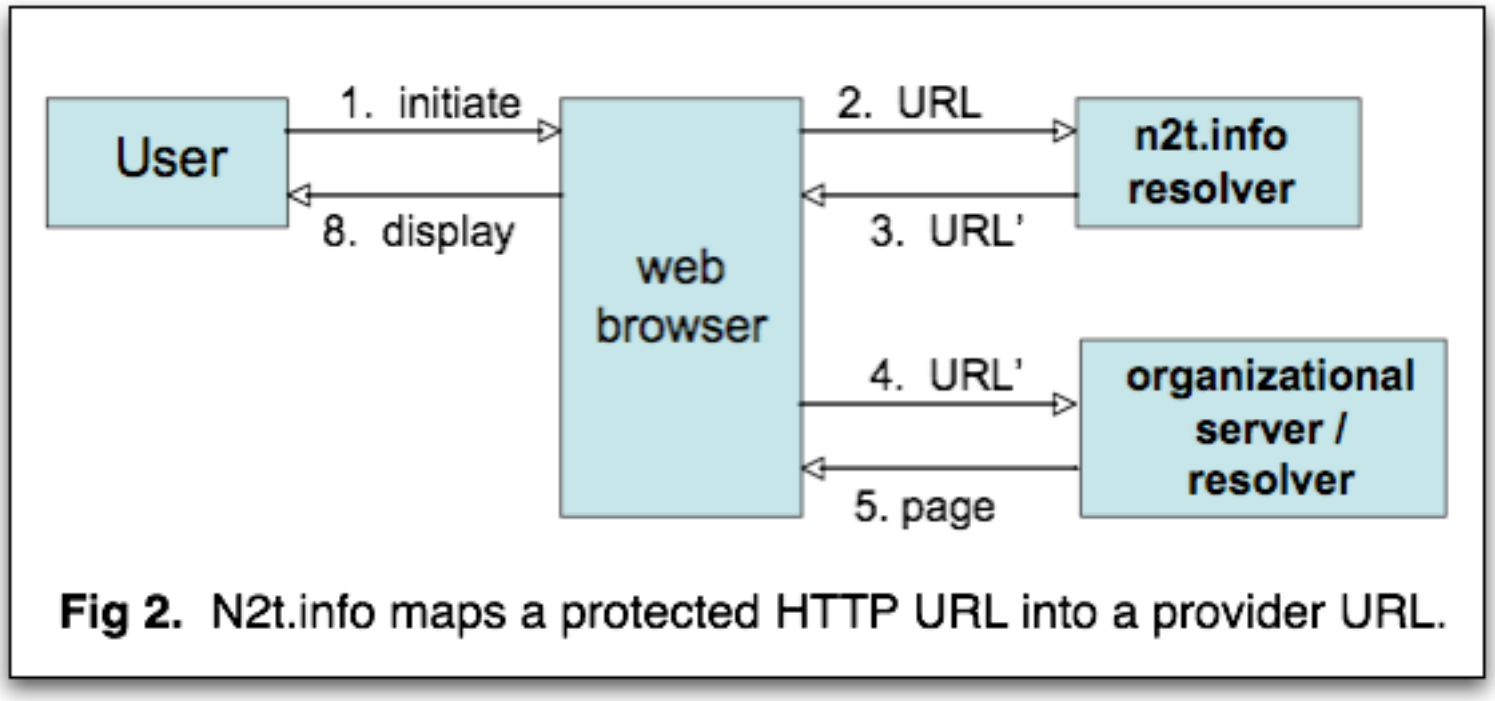


Fig 2. N2t.info maps a protected HTTP URL into a provider URL.

Examples using n2t.info

http://n2t.info/12345/de/3.html
 resolves to http://www.uni-goettingen.de/de/3.html
http://n2t.info/67890/wir/index.htm
 resolves to http://www.ddb.de/wir/index.htm
http://n2t.info/24680/bilderBerichteDokumente/index.html
 resolves to http://mpg.de/bilderBerichteDokumente/index.html
http://n2t.info/urn:nbn:se:uu:diva-2475
 resolves to http://publications.uu.se/reports/abstract.xsql?dbid=2475
http://n2t.info/ark:/13030/tf5p30086k
 resolves to http://cdlib.org/ark:/13030/tf5p30086k
http://n2t.info/doi:10.1111/j.0307-6946.2004.00571.x
 resolves to http://dx.doi.org/10.1111/j.0307-6946.2004.00571.x

Persistence and identifier strings

Desirable characteristics

Longevity -- make them opaque (free of meaning) to avoid inevitable semantic “rot”

Transcribability -- make them short (eg, tinyurl)

- Keyboarding, typesetting, error detection

Pressure to accommodate meaning, at least selectively, even in long-term persistent identifiers

User-friendliness: non-opaque, transient qualifiers

Provider-friendliness: branding

Tool-friendliness: non-opaque filename extensions

The best place to accommodate meaning is in metadata

N2T as consortium

No one organization owns the brand

Everyone owns it

Nearly free?

$30/year rents the n2t.info domain

Volunteers maintain 200 rewrite rules

Consortium as successor safety net?

An organization that can no longer operate can look among members for a successor

Current status

Experiment with 4 mirrored resolvers

California Digital Library

State and University Library, Göttingen

New York University

National Library of Australia

N2T and branding

The resolver (redirecting server) is at n2t.info

- *n2t.info* is short, good for usability
- *n2t* is reasonably opaque and non-branded, good for longevity

Accommodating branding *and* longevity?

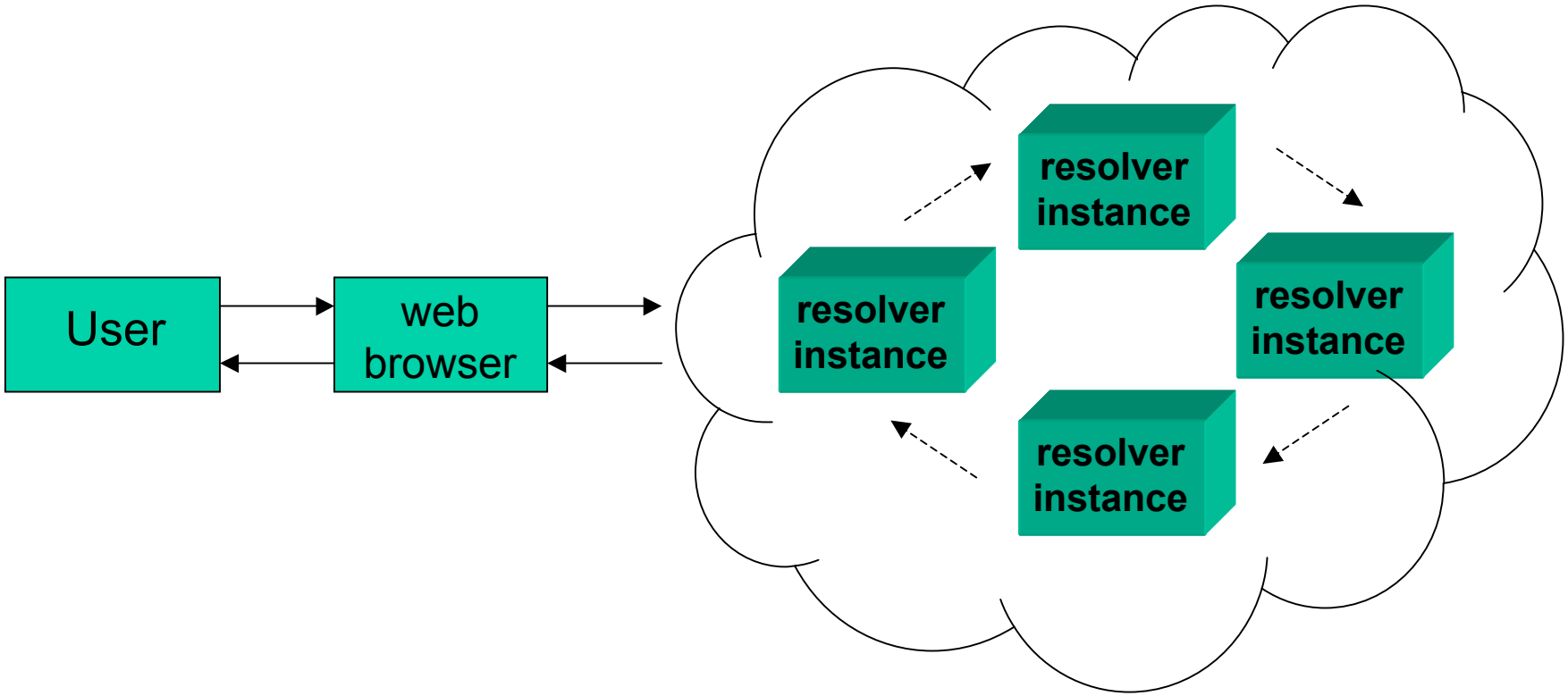
http://n2t.info/12148/wt8gh2br ?

http://n2t.bnf.fr/12148/wt8gh2br ?

http://bnf.n2t.info/12148/wt8gh2br ?

N2T – global point of view

Regional (eg, Europe, Asia, North America) clusters of mirrored resolver instances, with round-robin failover for redundancy, fault-tolerance, and load-sharing



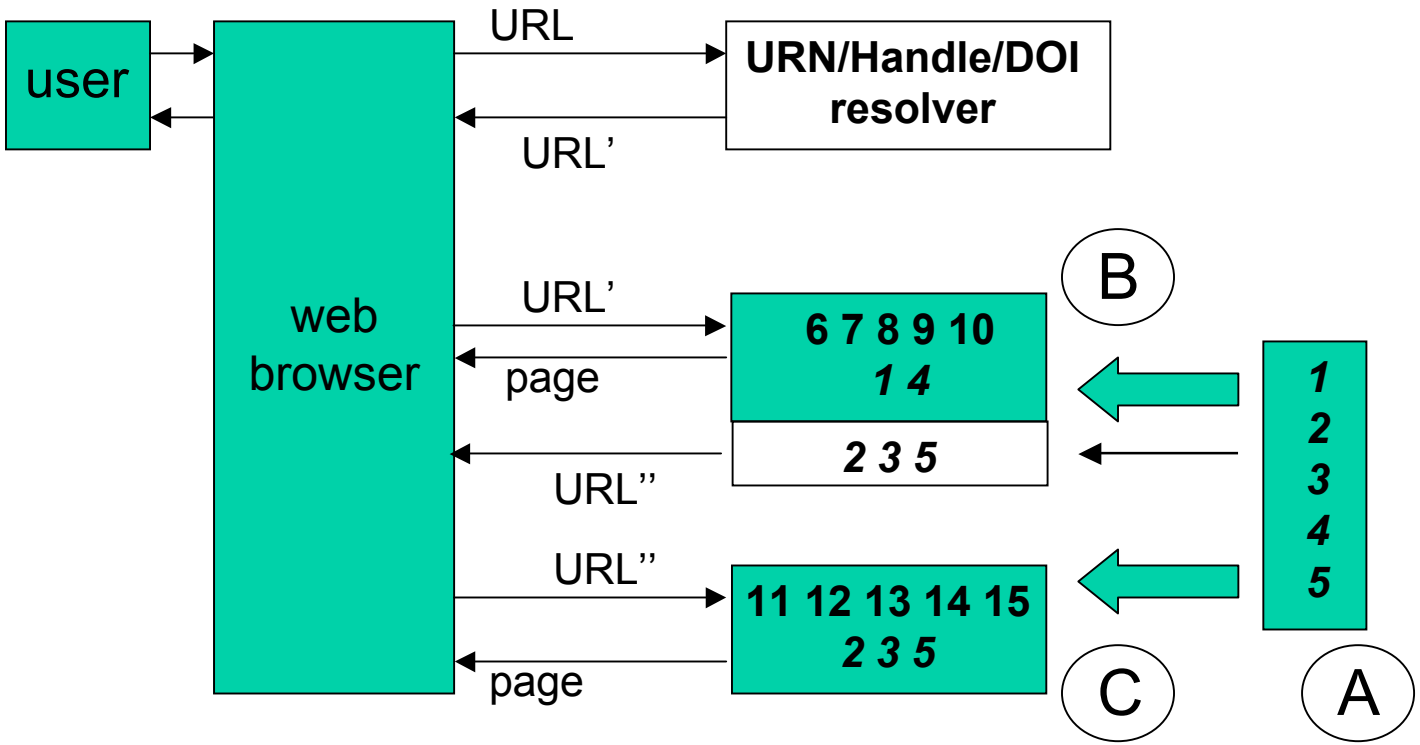
N2T and multiple id schemes

All depend on Name Assigning Authority

- http://n2t.info/NAA/...
- http://n2t.info/ark:/NAA/...
- http://n2t.info/urn:NAA:...
- http://n2t.info/hdl:NAA/...
- http://n2t.info/doi:NAA/...
- http://n2t.info/purl:/NAA/...

Where to register NAA number (eg, 12345)?

Namespace Splitting Problem



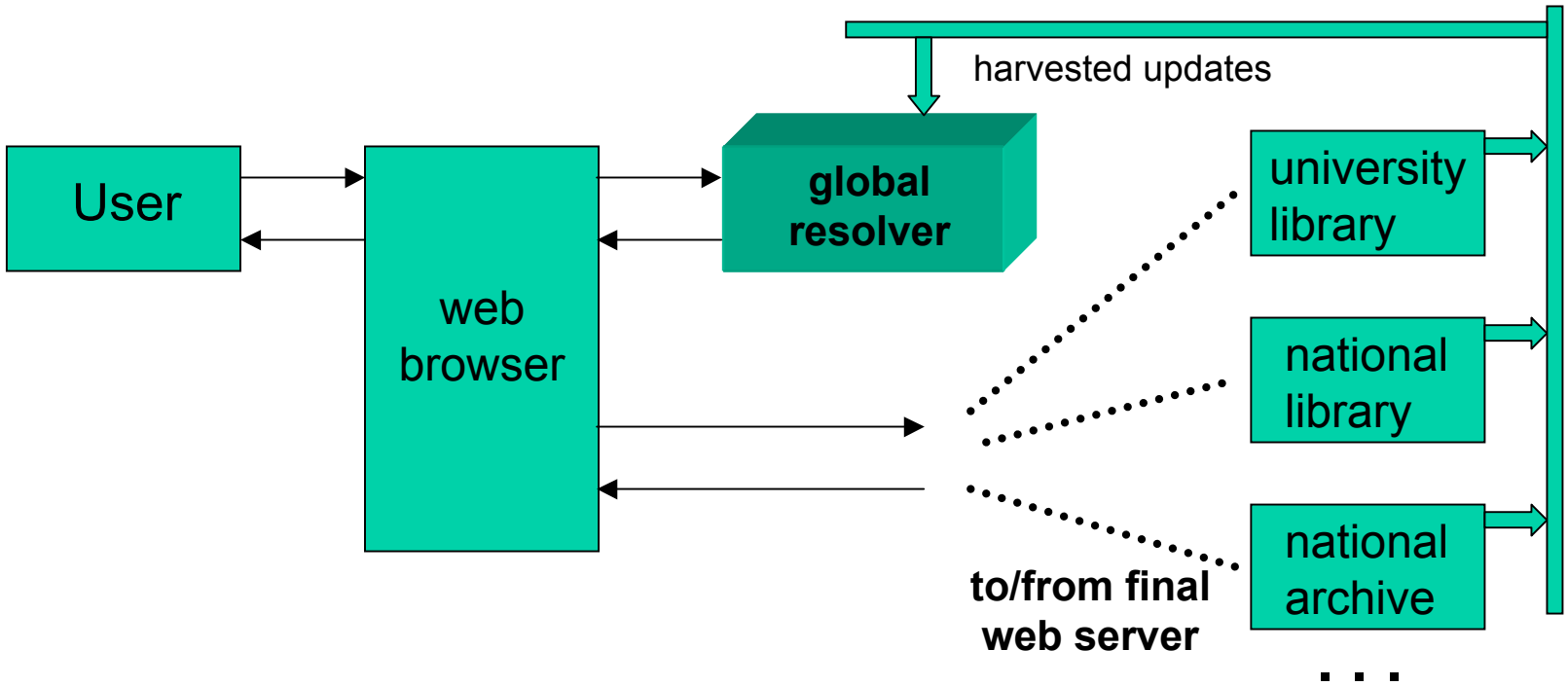
Org'n A's namespace splits when B and C inherit its objects.
Under the URN/Handle/DOI model, B must still forward to C.
A table is needed where it can be supported, e.g., N2T.

Global resolver updating

But to become a per-object resolver needs bulk updates

Periodic harvest (e.g., daily) of table mappings

From well-known provider-side web server files, e.g., tools and conventions similar to Google sitemaps



n2t.info summary

Persistent identifier resolution using a very simple architecture

No proprietary, special-purpose infrastructure to carry forward as a liability to persistence

No browser modification required

Identifier scheme-agnostic

What's a persistent identifier?

An *identifier* that is *valid* for long *enough*

- *valid, enough:* these are service/user dependent

What's an *identifier*? It's an association between a string and a thing. It follows that:

- An id is not a string of data (good)
- An id is a matter of opinion, not fact; there will be at least one other provider, serial if not in parallel, or your objects die with you (inconvenient)
  - Same thing, two strings; or same string, two things
  - Often: same string, different metadata
  - Often: same string, parallel things diverging over time due to different preservation practices (eg, migrations)

Accepting some disorder with persistent identifiers

Long term preservation won't happen unless objects can change residence and diverge

- Digitized book with identical distributed instances
- Replication good for preservation

In general, Producer loses control of copies

- Multiple opinions and practices will flourish
- Static, id-based persistence claims soon irrelevant
  - "urn:…", "hdl:…", etc. reflect hopes of people long gone

Not pretty, but the alternative (loss) is worse

Agreeing to disagree

What we say, but shouldn't (not loudly):

- Don't re-assign a persistent id to something else
- **Or** don't replace a persistent object with another

What we do:

- Knowingly replace our persistent objects (typos, drafts, format conversions, home page redesign)
- Honestly provide a real kind of persistence, but with very different replacement policies
- Won't have one way within one providing organization, let alone outside it

Diverse persistence practice

How dissimilar must two objects be before they get different ids?

Requirement: need to be able to tell users what flavor of permanence is in effect

Persistent identifiers should…

Identify, with or without the object “at hand”

- May not be convenient, helpful, or permitted for you to inspect object itself -- metadata needed

Convey different flavors of permanence

Lead to access (if authorized)

- Not strictly an “identification” problem, but it is the “404 not found” that we need to fix

Be valid for some longish period

Be carried on, in, or with the object

How to choose an id scheme

All our requirements are purely about service

Candidate schemes: URL, PURL, URN, ARK, Handle, DOI, MD5, GUID, ISxx, …

- *There is no direct service help from any of these schemes; no scheme or syntax confers persistence of any kind*

We then ask which schemes are lowest cost and lowest risk?

Scheme costs and risks

Every modern service needs to support indefinitely and find or be given replacements for at least

- Web server, web browser, and DNS

In addition, URN, Handle, and DOI resolution need a global proxy or a plugin *for every* access

ARK might be able to use a plugin, but doesn't need it, and no plugin addresses the namespace splitting problem

Handle and DOI also require

- You to maintain an extra local server
- Your community to maintain a set of global servers

Risks

- Handle and DOI come with highest risk
- ARK comes with lowest risk

Persistence - indirect factors

Persistence requirements call for an identifier scheme (not a service) connecting users to

- metadata
- *whether* persistence and what *kind* of persistence
- sub-object and variant inferences
- core object identity on proxy failure (graceful degradation)

Scheme impact: ARK provides these

  - A scheme is not a service (DOI is not CrossRef)
  - When choosing a scheme, we wanted to remain independent of extra external service providers

Opaque ids with semantic extensions

Tension between acces and longevity

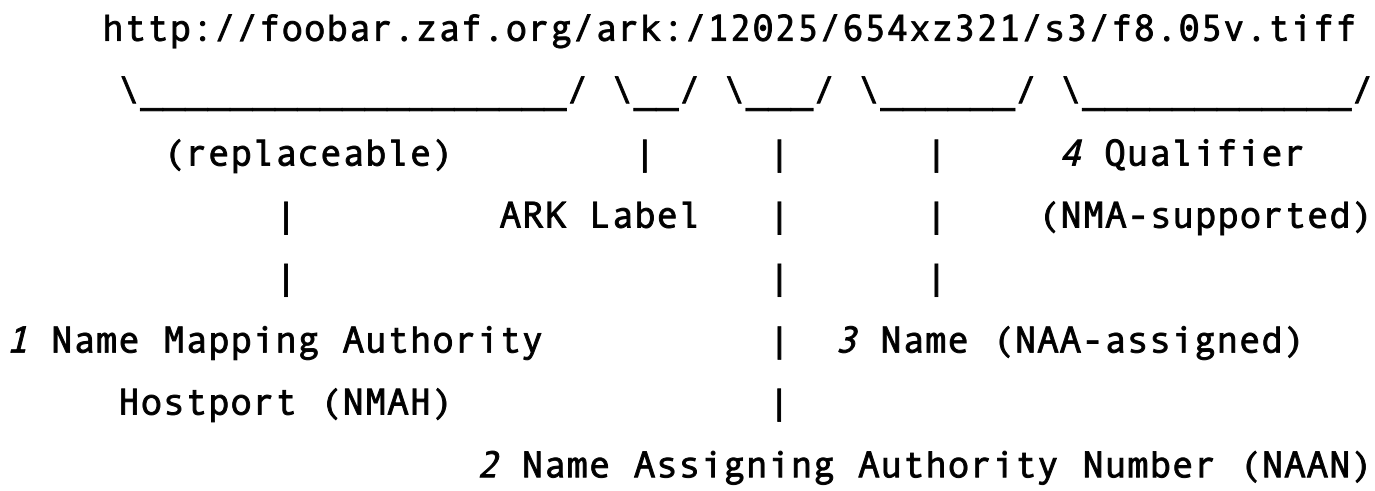
- opaque ids are needed for names that age and travel well
- Semantically laden ids are helpful in providing many id services

Hybrid:

- opaque ids are used to name abstract preservation objects
- Semantic and sometimes transient extensions address components inside of objects (the set of components evolves over time anyway)

ARK Summary

Instead of one Name Authority: Assigning Authority + Mapping Authorities



1 = current service provider; identity inert; replaceable

2 = organization that originally assigned the id

3 = name originally assigned to the abstract object, often opaque

4 = extension disclosing object hierarchy & variants, often non-opaque

ARK usage

Two ARKs accessing the same thing

`http://loc.gov/ark:/12025/654xz321`

`http://rutgers.edu/ark:/12025/654xz321`

Access to metadata -- add a ‘?’

`http://loc.gov/ark:/12025/654xz321?`

Access to support statement -- add ‘??’

`http://loc.gov/ark:/12025/654xz321??`

3 minimal requirements to be an ARK

- An archive that can’t do all 3 -- trustworthy?
- Is an ARK persistent? Maybe. Have to *ask*.

Persistence and opaqueness

Do ARKs have to be this ugly (opaque)?

```
http://foobar.zaf.org/ark:/12025/654xz321/s3/f8.05v.tiff
 \_____________________/ \__/ \___/ \_____/ \____________/
         NMAH          Label  NAAN   Name     Qualifier
```

No, but it is encouraged. Persistence is all about managing associations between strings and things

- And the landscape is littered with links that *were required* to die for political, legal, or social reasons
- the appearance, deliberate or even accidental, of once-true assertions that are now misleading, infringing, offensive makes it hard for our descendants *to continue managing*

Pain of opaque ids mitigated by strongly bound metadata

ARK lexical goodies

Hyphens ignored

- Neutralizes harm done by typesetters

Too many search results?

Providers may disclose (or not)…

- Sub-object hierarchy using reserved ‘/’
- Variant objects using reserved ‘.’
- Usual %hh (hex encoding) as an escape

Revealing hierarchy in ARKs

Sub-object hierarchy using reserved *x/y*

Meaning *x* is containing object for *y*, for some defined containment relationship

Works for chapters, sections, pages, etc

Persistence of that relationship?

Saying ark:/12025/654/xz/321 implies also

- ark:/12025/654/xz
- ark:/12025/654

Revealing variants in ARKs

Sub-object hierarchy using reserved *x.y*

Meaning *x* is an object basename with variant *y*, for some defined variant relationship

Works for formats, versions, languages, etc

Persistence of that relationship?

Saying ark:/12025/654.20v.78g implies also

- ark:/12025/654.20v
- ark:/12025/654

ARK namespaces reserved

12025	National Library of Medicine
12026	Library of Congress
12027	National Agriculture Library
13030	California Digital Library
13960	Internet Archive
13038	World Intellectual Property Organization
78319	Google
61001	University of Chicago
28722	University of California Berkeley
15230	Rutgers University Libraries
64269	UK Digital Curation Centre
62624	New York University Libraries
52327	National Library and Archives of Quebec
27927	Portico/Ithaka Electronic-Archiving Initiative
12148	National Library of France

Reserve a namespace by email to ark@cdlib.org