# URIs for Named Information draft-farrell-ni

Stephen Farrell
stephen.farrell@cs.tcd.ie
Crhistian Dannewitz
cdannewitz@upb.de
Borje Ohlman
Borje.Ohlman@ericcson.com
Dirk Kutscher
kutscher@nec.de

# Motivation

- Emerging need for naming resources uniquely
  - Without any notion of location -- IP addresses, domain names don't work
- P2P applications and DECADE
  - Replicating objects (and their fragments)
  - Need a way to identify them uniquely
  - But in a location-independent way
- DECADE architecture:
  - `The name of a data object is derived from the hash over the data object's content (the raw bytes), which is made possible by the fact that DECADE objects are immutable.`

# Naming in DECADE

- Architecture describes naming requirements and fundamental concepts
  - Each DECADE protocol spec expected to provide details on format and semantics

- DECADE architecture naming scheme
  - **`type`**: indicates that the name is the hash of the data object's content and the particular hashing algorithm used
  - **`content hash`**: by applying the algorithm and (possibly) a specific presentation format

# Deployment Considerations

- In DECADE we are mainly interested in uniqueness property
  - DECADE servers should not be required to calculate the hash
  - Just use the name for identification, de-duplication etc.

- As crypto algorithms evolve, we will see different types of names
  - Also: different P2P apps might evolve to use different types
  - DECADE implementations should be able to use those names, without understanding all possible variations (if hash calculation is not used)

- Want a Uniform Format – allowing for
  - Representing and using all names
  - Understanding the semantics (for checking hashes by applying the right crypto algorithm)

# How/Examples

- DECADE and other protocols for naming information objects need names, but keeping it simple is required; URIs seem an obvious choice
  - Or...maybe URNs?

- Sometimes, you might want a name to include a hash of something; used in many applications, each invents its own way to do that, good to standardise
  - No implication that applications MUST check anything; algorithm-agility; can specify input to hash; can specify “inner” content type if hash input was an envelope

- This 11-page I-D defines a way to include hashes in HTTP-like URIs
  - `ni://tcd.ie/cs8053-exam-2012`
  - `ni:///weather-in-dublin-today`
  - `ni://tcd.ie/sha256:NDVmZTMzOGVkY2JjZGQ0ZmNmZGFlODQ5MjkyZDM0ZTg2ZDI5YzllMmU5OTFlNmE2Mjc3ZTFhN2JhNmE4ZjVmMwo`
  - `ni://sha256:NDc0NzgyMGVmOGQ3OGU0MmI2MWYwZjY3MDAzNDJmZTY0NzhhMGY0OTBhMDRiNzA0YTY0MWY0MzVkODQzZWUxMAo:id:sshpk/thing`
  - `ni://tcd.ie/sha256:NDVmZTMzOGVkY2JjZGQ0ZmNmZGFlODQ5MjkyZDM0ZTg2ZDI5YzllMmU5OTFlNmE2Mjc3ZTFhN2JhNmE4ZjVmMwo:signeddata:application%2Fjpeg`

# To-Do/Process/An Ack

- I-D:
  - More on truncated hashes
  - New URI scheme (ni:) or URN?
  - Maybe define a few things that could be hashed
- Process:
  - DECADE might adopt, or try get this AD-sponsored
  - Not keen to try spin up a naming WG
- ACK: Work funded by FP7 SAIL project
  - So we'll be coding this up (or whatever emerges as the right thing)

# Backup: ABNF

```
ni-name = scheme ":" hier-part

   hier-part = "//" [authority] "/" *(local-part "/") ["/"]
   scheme = "ni"

   authority = hash-string | other-string ;(delimiters %-encoded)
   local-part = hash-string | other-string ;(delimiters %-encoded)

   hash-string = hashalg ":" b64value
                      [ ":" function-identifier [ ":" mime-type ] ]

   hashalg = identifier
   function-identifier = identifier

   identifier = ALPHA *( ALPHA / DIGIT / "+" / "-" / "." )
   mime-type = type %2f subtype
   ...
```