Requirements for accessing data in network storage

draft-ohlman-decade-add-use-cases-reqs-01

Börje Ohlman
Ericsson

Ove Strandberg
Nokia Siemens Networks

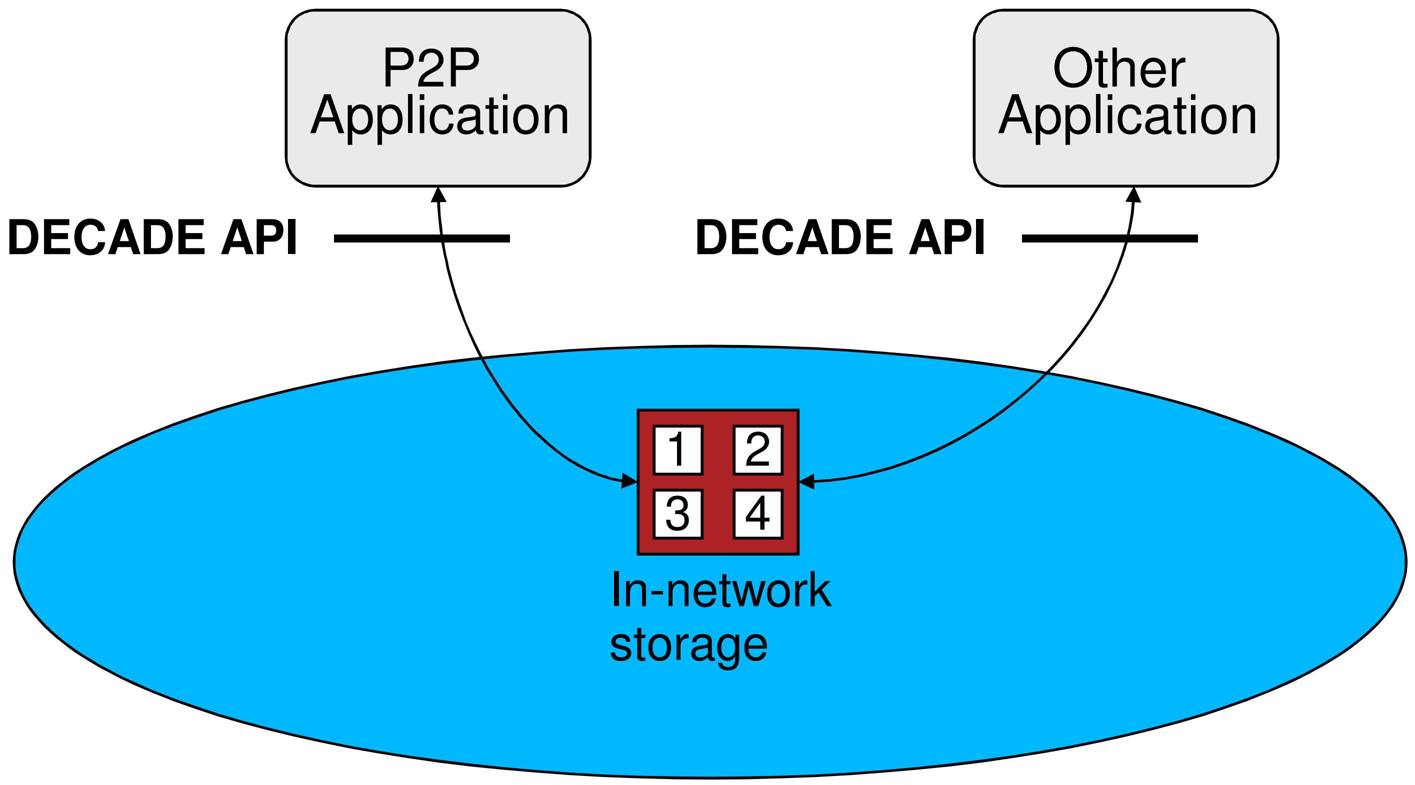
& 4WARD WP6 colleagues

IETF DECADE Maastricht
July 27th, 2010

Requirements for accessing data in network storage

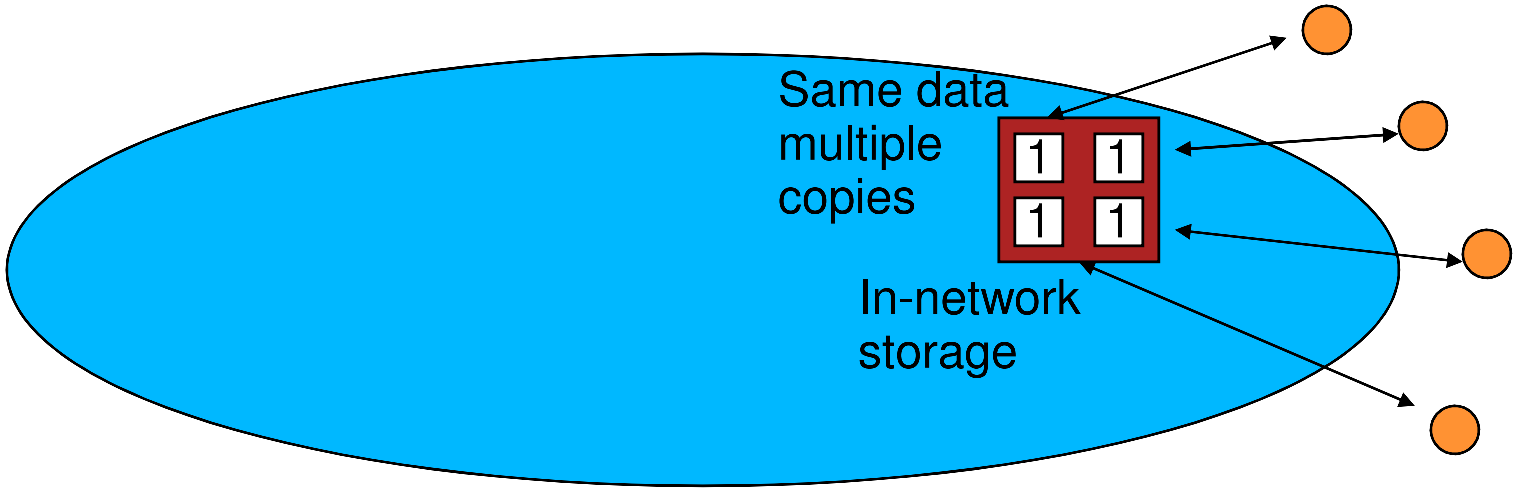
- Additional requirements to complement "draft-gu-decade-reqs-05"
- Requirements:
  - Application agnostic
  - Data reuse
  - Roaming users

Requirement: Application agnostic



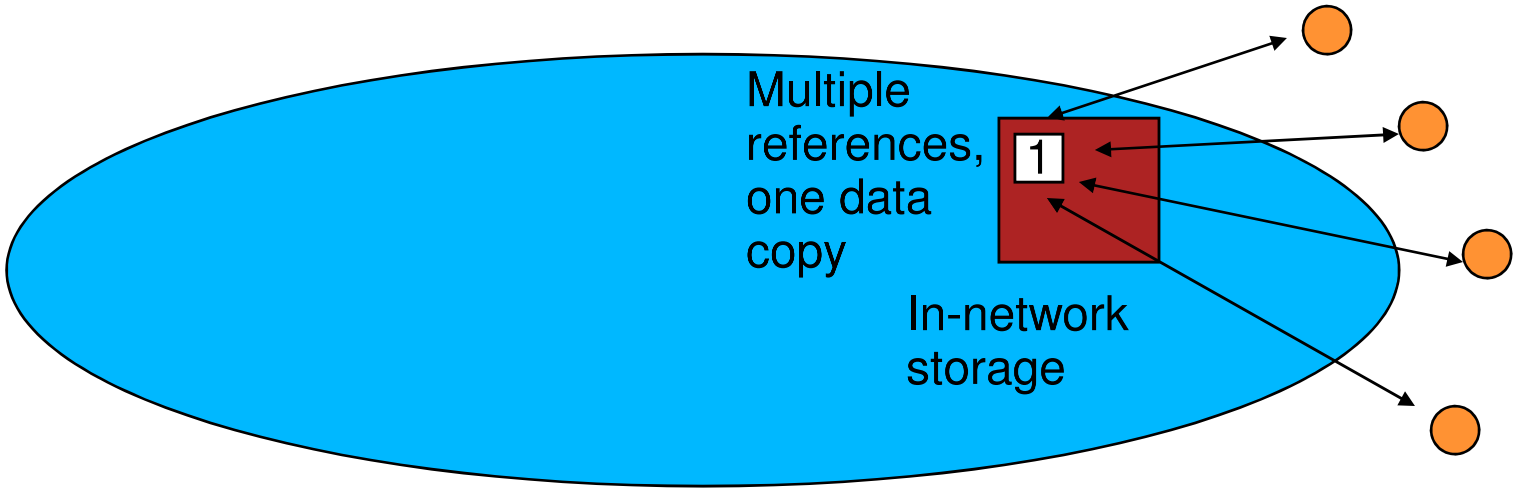
Requirement: Data reuse

With current draft

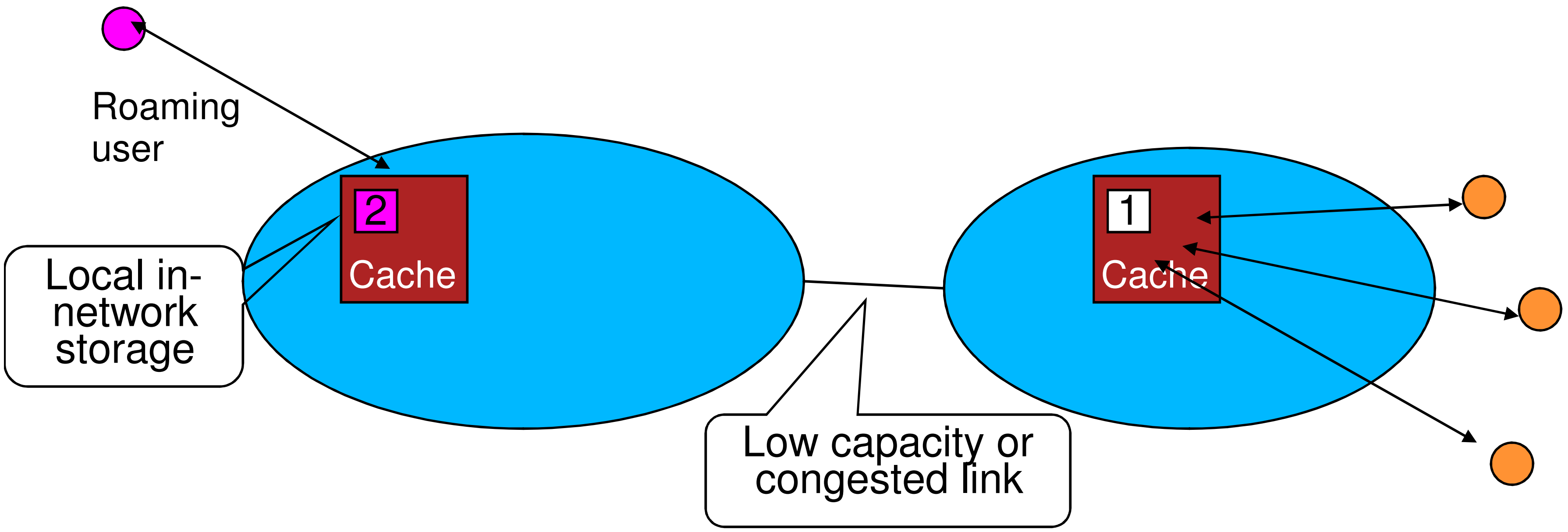


Requirement: Data reuse

With Data reuse requirement

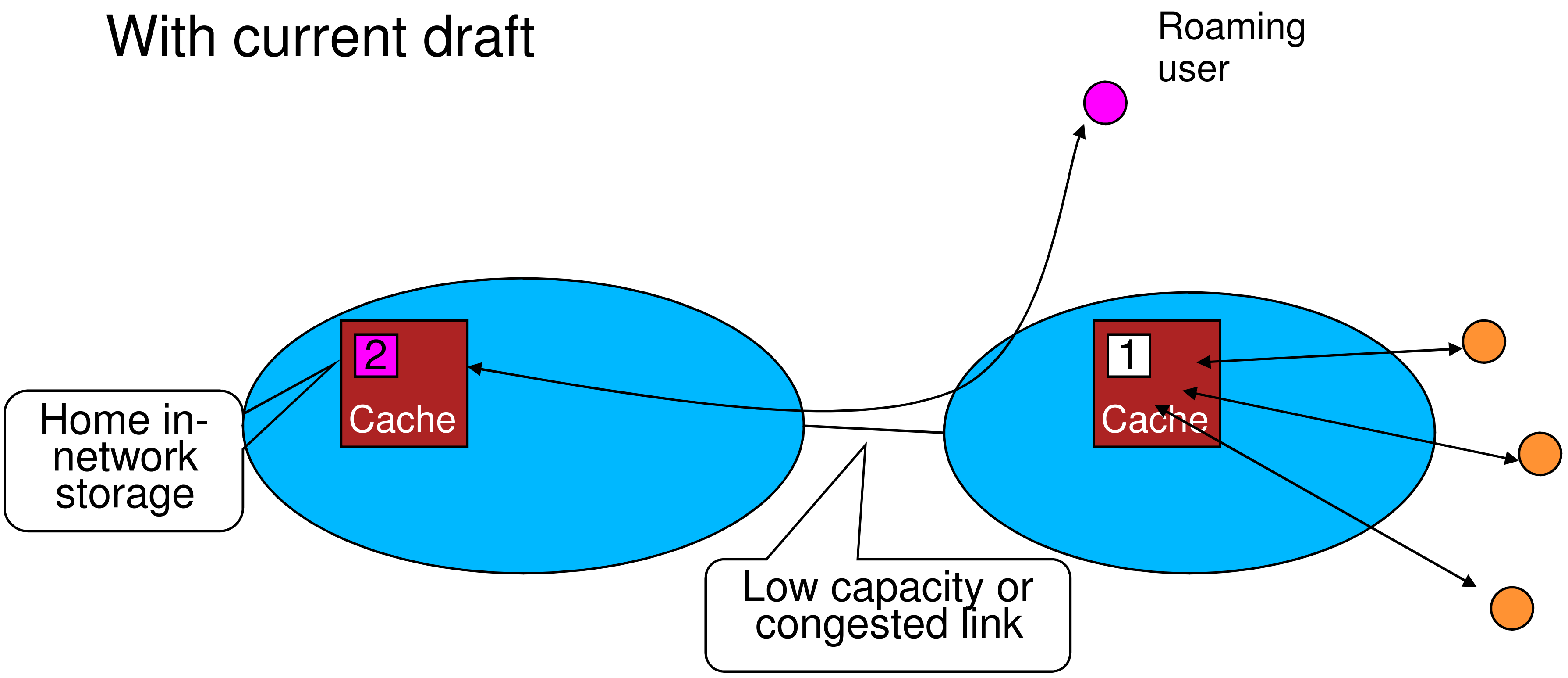


Requirement: Roaming users



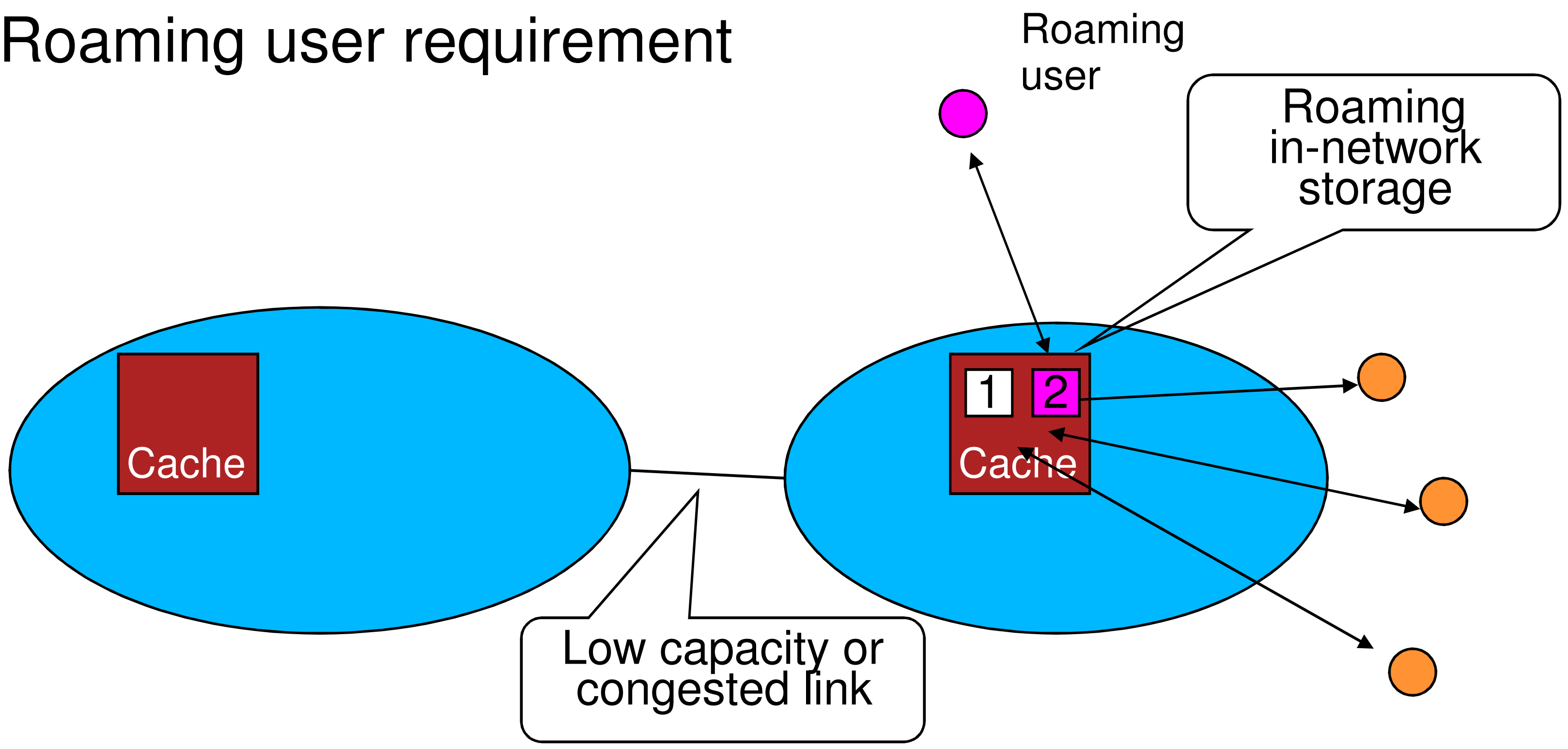
Requirement: Roaming users

With current draft



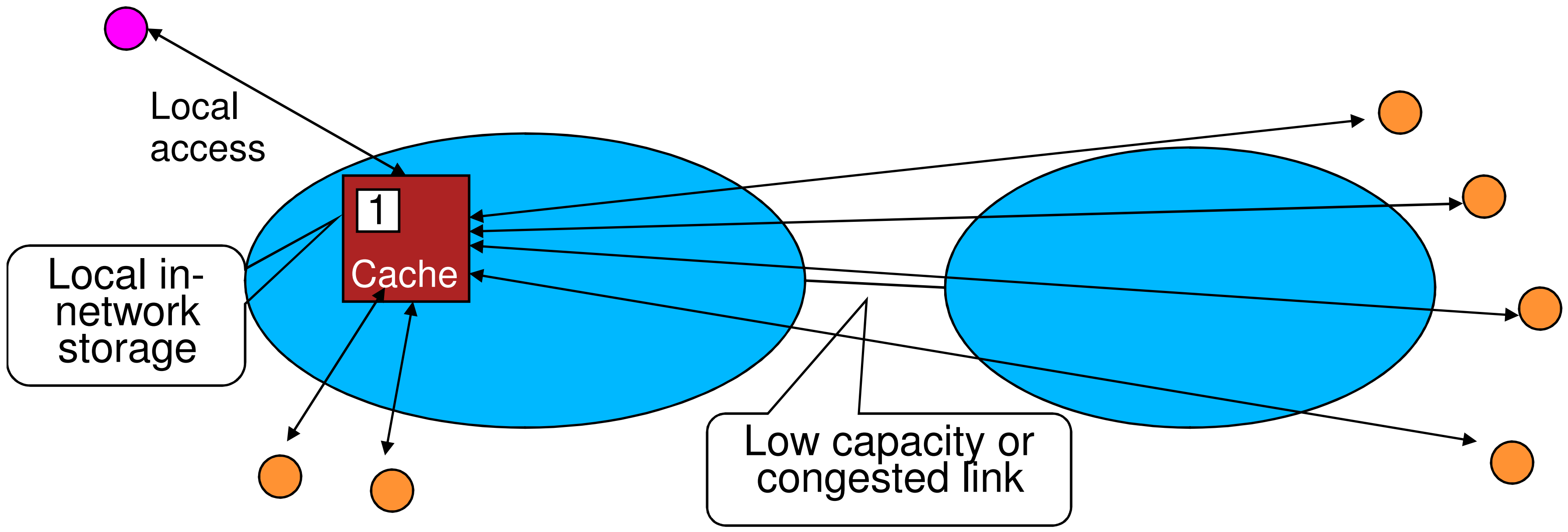
Requirement: Roaming users

With Roaming user requirement

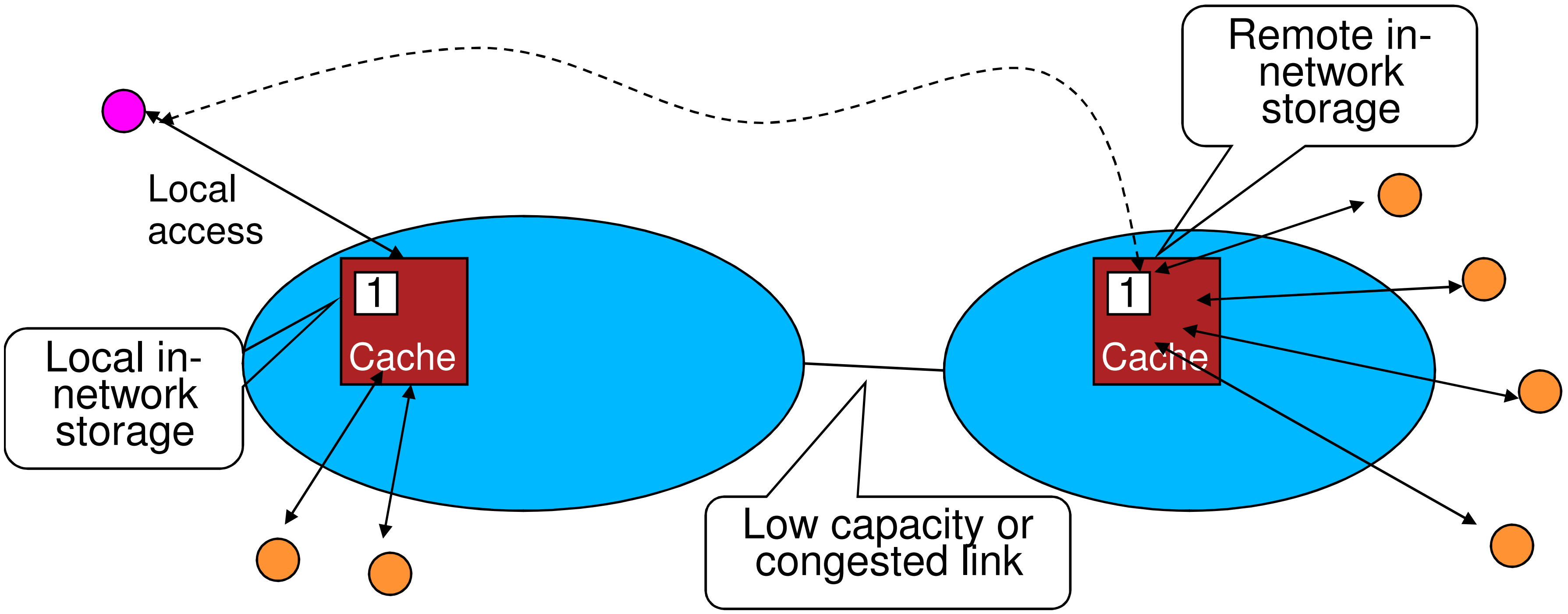


Allowing remote in-network storage

With current draft



Allowing remote in-network storage



Requirement: Application-agnostic

- The specification of DECADE SHOULD strive to make the DECADE in-network storage application-agnostic. As a verification of this effort DECADE SHOULD be specified in such a way that it can address at least one other application type, besides P2P applications.
- Rationale
  - To build separate in-network storage infrastructures for other application types, beside P2P, seems wasteful
  - IPTV Use case: distribution of video content, mainly focusing on Video-on-Demand (VoD) services and user-generated contents.
  - Characteristics of IPTV scenario:
    - support an efficient large-scale distribution of video;
    - high degree of replicated contents to a multiplicity of fixed and mobile users

Requirement: Data reuse

- When certain content is popular and used by many users, the network part of DECADE SHOULD be able to store only one (or any limited number) copy of that data. The same data can then be used to serve requests from multiple DECADE users.
- Rationale
  - Generally several copies in multiple storage good for user response – when data popular and copies in different locations
  - Target is efficient use of available data, unnecessary data copies not needed

Requirement: Roaming users

- DECADE SHOULD have the ability to support roaming for terminals/ users/applications by allowing the use of in-network storage in the visited network.
- Rationale:
  - A user of DECADE in-network storage that roams to a visited network could potentially cause very inefficient access to that user's DECADE storage.
  - It is therefore essential that the user is able to acquire new DECADE storage which is better located in the visited network.
- Discussion:
  - A related issue is the possibility to migrate content from one DECADE storage to another when roaming.
  - We believe that this is covered by the requirements on Efficient Transfer (section 3.3) and Communication among In-network Storage Elements (section 4.3) of [I-D.gu-decade-reqs].

Summary and Conclusion

- Additional requirements
  - Application agnostic
  - Data reuse
  - Roaming users
- Adopt requirements into “draft-gu-decade-reqs”
- Selection of reference non-P2P application

- Information on the related concept Networking of Information (NetInf)
  - http://www.netinf.org
  - http://www.4ward-project.eu/

Backup slides

Problems resulting from Host-centric view

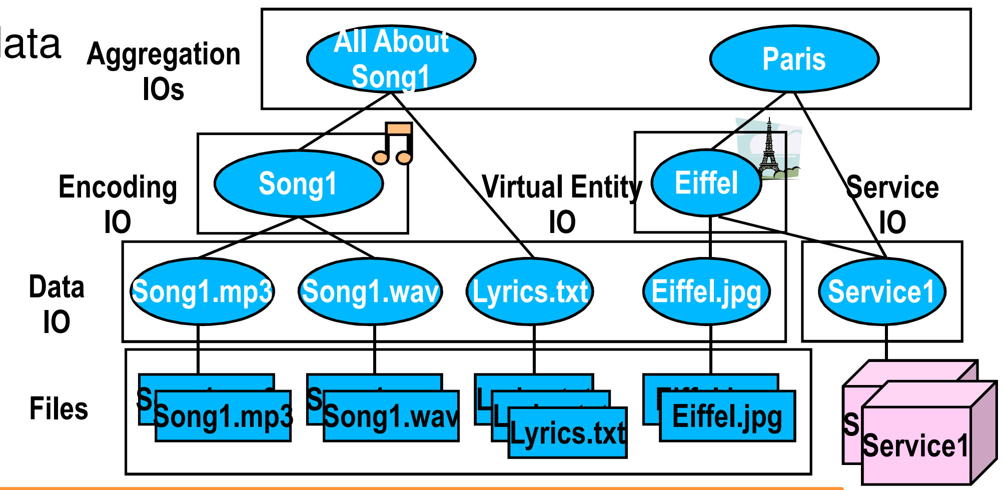
- No common *persistent naming scheme* for information
  - URLs and IPs overloaded with locator and identifier functionality
    - Moving information = changing it‘s name („404 file not found“ errors)

  => Use *flat namespace* for persistent identification
- No consistent *representation of information* (copy-independent)
  - No consistent way to keep track of *identical copies*
  - Different *encodings* (e.g., mp3, wav) worsen problem
- Information dissemination is inefficient
  - Can‘t benefit from existing copies (e.g. local copy on client)
    - Also true for *Content Delivery Networks* (e.g. Akamai)
  - No “anycast”: e.g., get “*nearest*” copy
  - Problems like *Flash-Crowd effect, Denial of Service, …*
- Security is host-centric
  - Mainly based on *securing channels* (encryption) and *trusting servers* (authentication)
  - Can’t trust a copy received from an untrusted server

Problems can be solved in a consistent manner via an information-centric architecture

Identifier and Information Modelling

- ❖ 1. Step: *Persistently identify information* via *identifier/locator split*
  - – Location-independent identifiers
  - – Represent *multiple copies*
- ❖ 2. Step: *Representation of information* via *Information Objects (IOs)*
  - – Another level of indirection
  - – Represent information independent of *specific copy* and *encoding*
    - • *E.g. a text, a song*
  - – Contains *information-specific* metadata
    - • E.g. access rights, attributes
- ❖ Information Objects can do more:
  - – Representation of:
    - • Streams
    - • Services
    - • Real-world objects (e.g., a book, person)
  - – IOs can be used to organize information



- ❖ **Enables efficient information dissemination**
  - – System can automatically choose encoding and copy (e.g. based on metadata)
  - – User can navigate information (e.g. choose encoding)

NetInf Naming Scheme

Type	**A** = Hash(PK_{IO})	**L** = {identifier tags}

Type

- Defines the format
  - e.g. Hash algorithm used (SHA1, MD5, …)

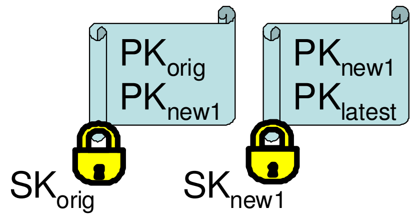
Authenticator (A)

- Binds the ID of the IO to a public key PK
  - Hash function used to compress length of PK

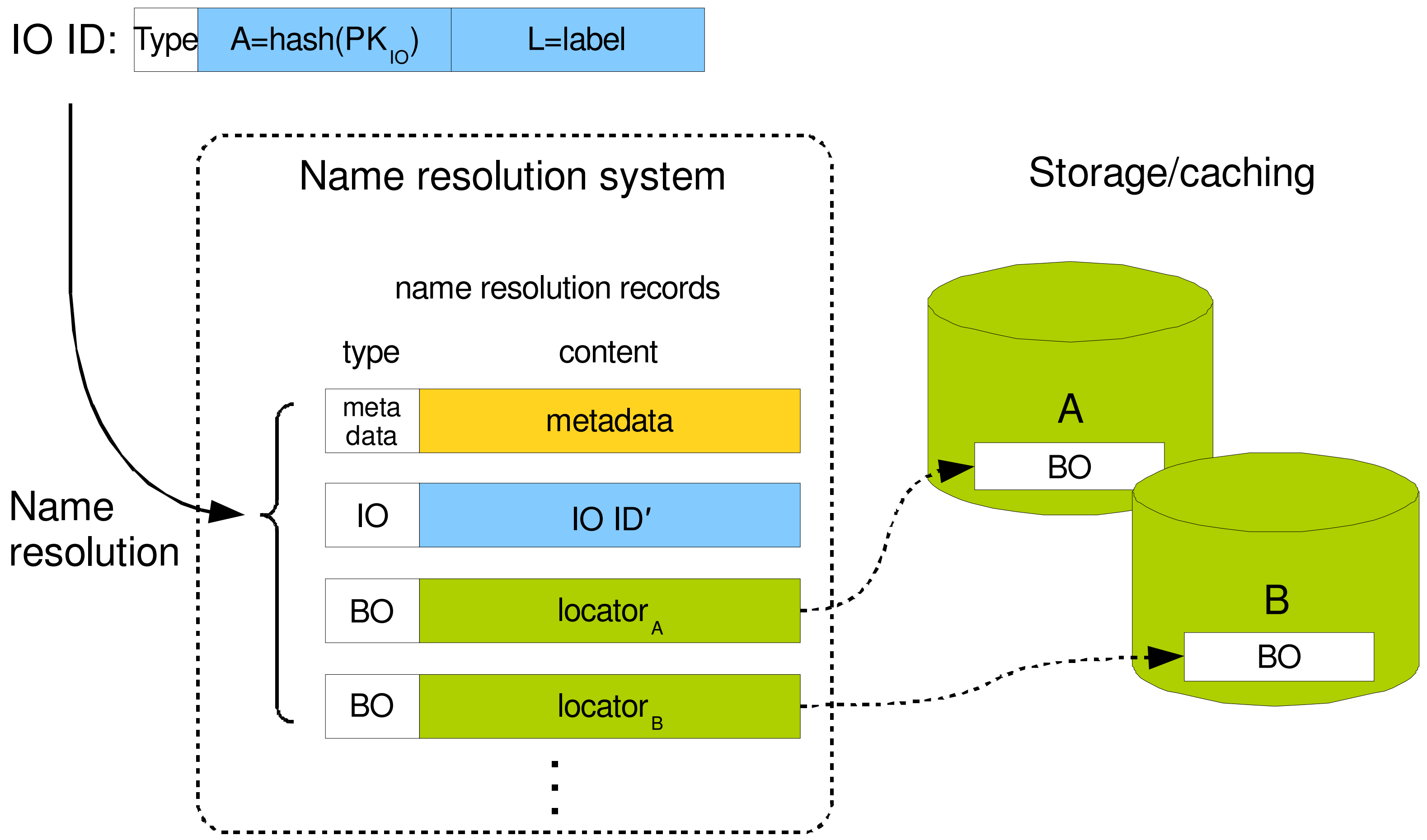
Label (L)

- Identifying individual object published by Authenticator
  - contains a number of identifier attributes associated with an IO
  - (A, L) combination needs to be globally unique

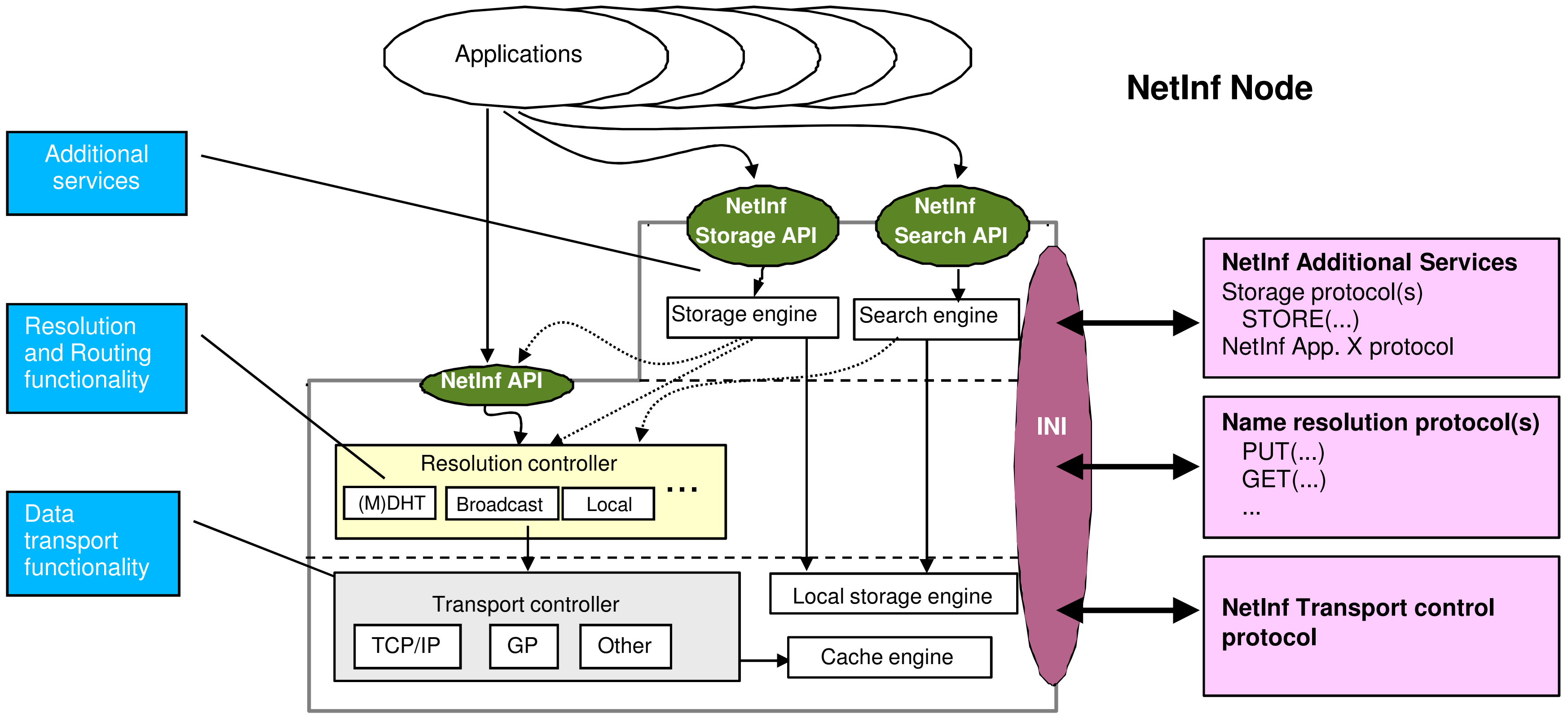
- Supporting the combination of:
  - name persistence, self-certification, owner authentication, and owner identification
  - for information objects that are static or dynamic, that change location, that change owner, and whose owner change organisational affiliation
  - without need to trust the delivering host
- Security metadata include
  - Cryptographic hash of the data
  - Certificate chain binding the IO key (PK_{IO}) to owner key (PK_{owner})
  - Signature of the self-certified data with SK_{IO} or any other authorised SK
  - All data needed for owner authentication and identification
- Separation of notion of *publisher* and *owner*, where the latter can stay anonymous



Mapping Information Model to NetInf name resolution

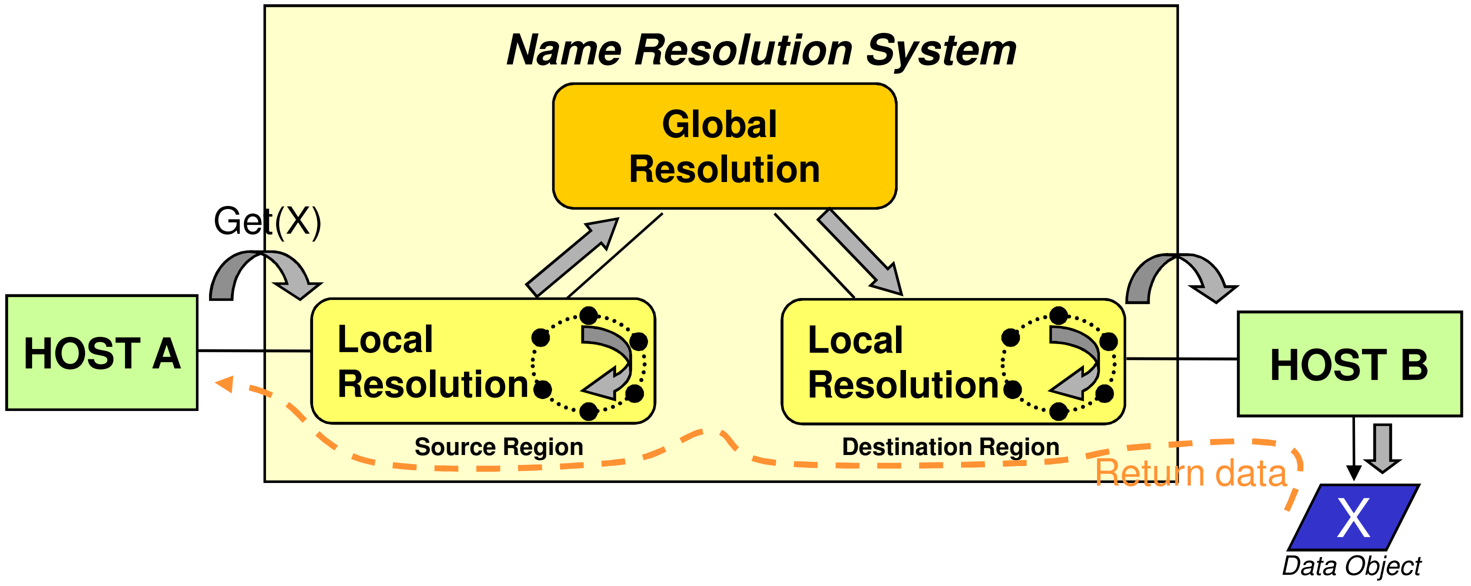


NetInf Node Architecture



Scalable Name Resolution System

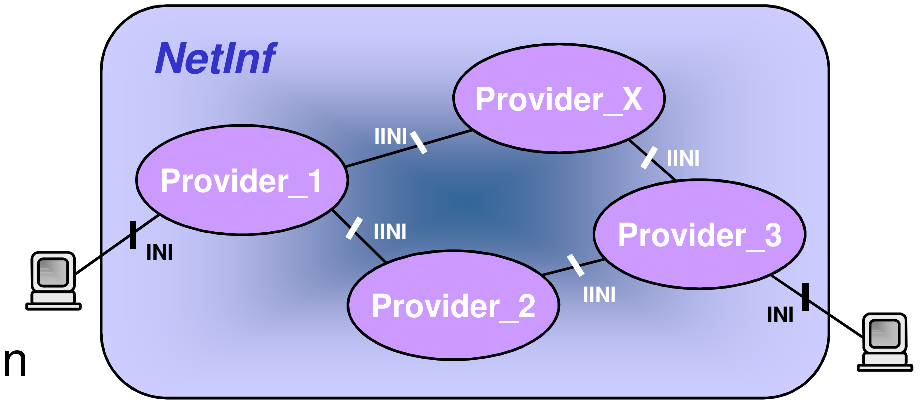
- Combination of:
  - Hierarchical DHTs (Provider-based)
  - Topological embedding of DHTs
  - Name-based routing



Standardization issues

Issues to be considered for standardization:

- ❖ NetInf interfaces
  - – User-Network Interface
  - – Inter-Provider Interface
    - Name resolution service between different domains
      - – registration/query protocols
      - – name resolution records
    - Transport and routing of messages
- ❖ Naming scheme
- ❖ NetInf APIs



INI	= Information Network Interface
IINI	= Inter-Provider INI

Summary and Conclusion

- NetInf is about the design of a *new network architecture* based on an *information-centric* paradigm
- Some problems that are addressed by the NetInf
  - Naming scheme for *naming information*
  - World-wide scalable *Name Resolution* mechanism for flat names
  - IOs as *representation of information*
  - Enable *efficient information dissemination*
    - Benefit from *available copies, anycast, caching, solve Flash-Crowd, …*
  - *Secure information-centric architecture* by embedding security into identifiers
- Some significant results up to date
  - Naming scheme with integrated security that is independent of hosts
    - Detailed definition and security analysis
  - Design of a scalable integrated name resolution and routing scheme (MDHT)
  - Scalable solution for mobility of objects, hosts and networks (LLC)
  - Overall NetInf architecture prototype
    - Demonstrated at international conferences

Thank you !

www.4ward-project.eu