# Entity Identifiers for Lineage Preservation

Julien Gaugaz and **Gianluca Demartini**

L3S Research Center

Leibniz Universität Hannover

# Outline

- Project context: OKKAM
- Setting: Revision of Entity Identity Decisions
- Problem Statement
- Example Scenario
- Prime Number Labeling Scheme for DAGs
- Lineage Preserving Entity ID
- Discussion & Future Work

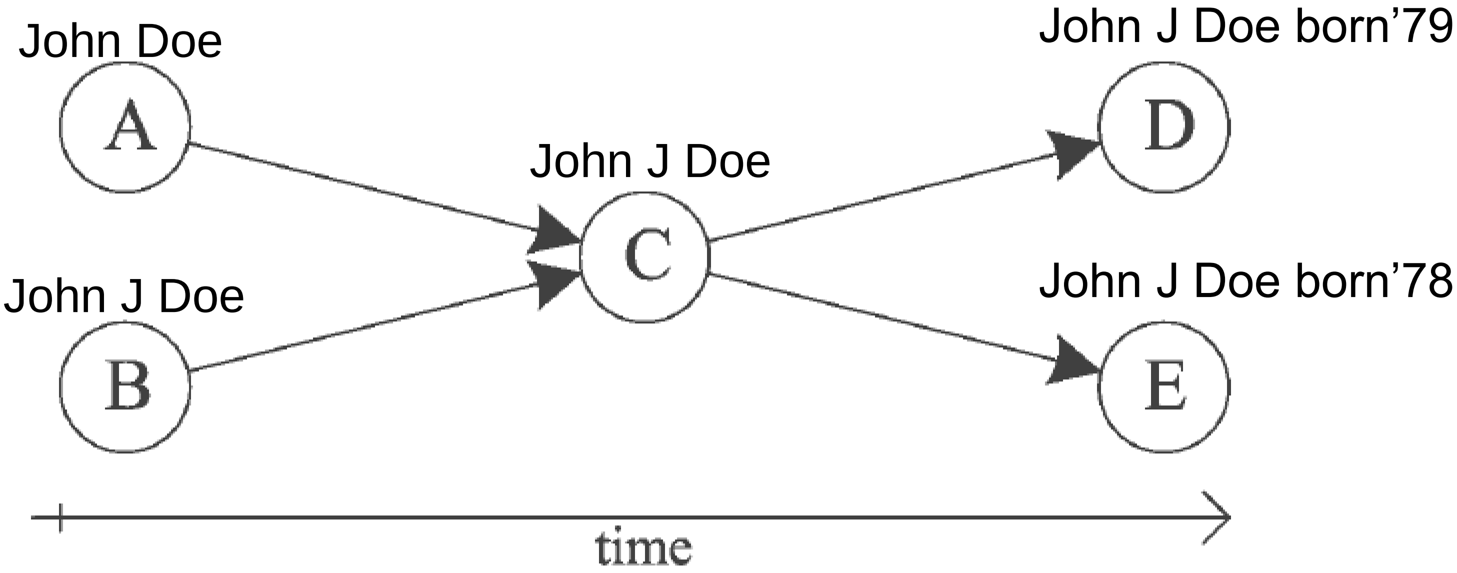
# OKKAM

- OKKAM Goal:
  - Foster the re-use of entity indentifiers to ease information integration
  - To create and manage a large collection of entity identifiers (EID)
  - Not to create a complete knowledge base
    - Only discriminative information is stored

# Settings

- Operations during the Entity Lifecicle:
  - Creation (ID Issuing)
  - Split
  - Merge

As a result of revising entity identity decisons



John Doe

A

John J Doe

B

John J Doe

C

John J Doe born’79

D

John J Doe born’78

E

time

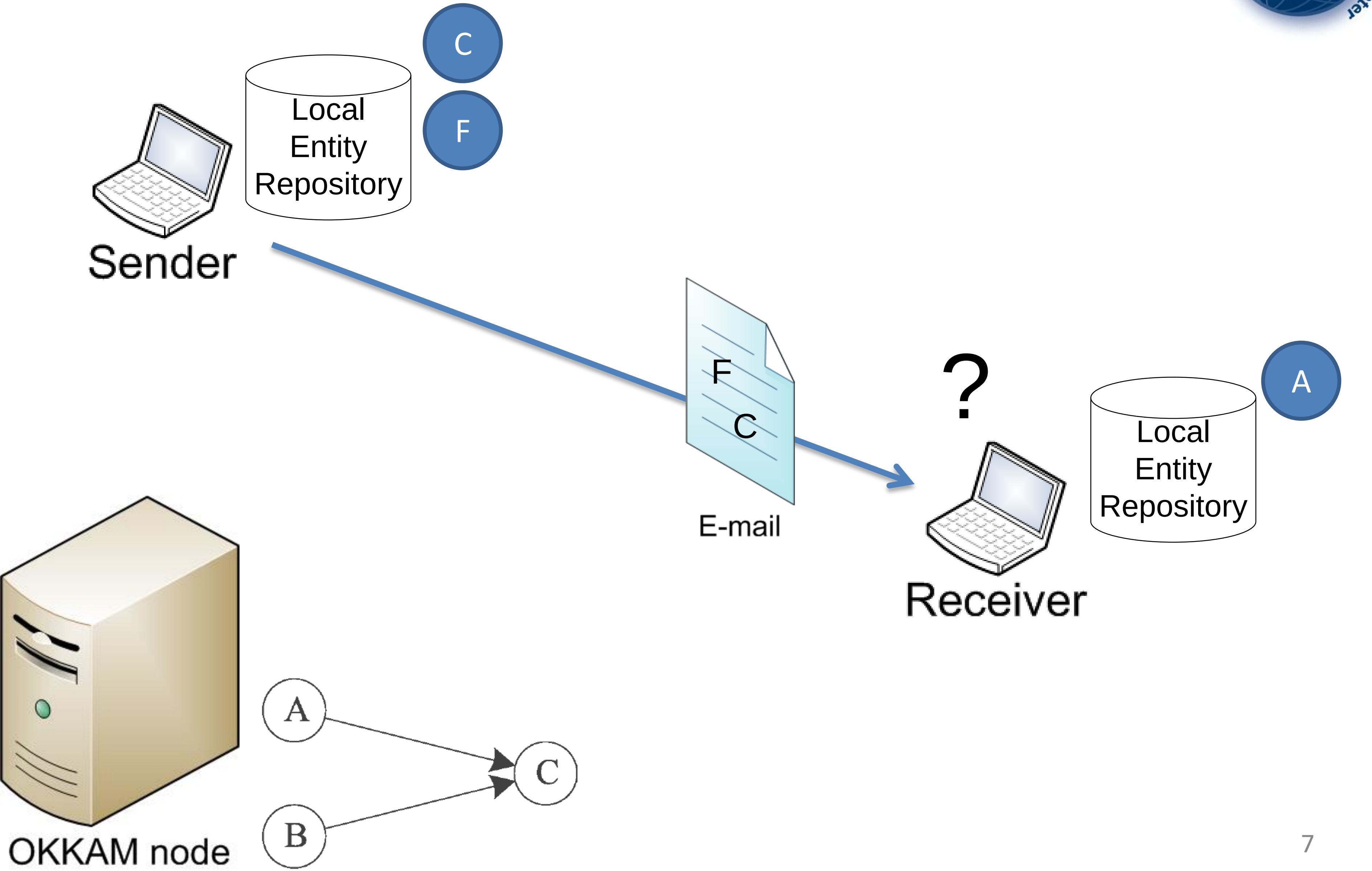
# Problem Definition

- Goal: Resolve locally the lineage of entities
- By providing Lineage Preserving EIDs
- Need for supporting the following operations:
  - Creation of EIDs
  - Resolve whether a given EID A is an ancestor of an EID B
  - Retrieve the list of all the ancestors of an EID
  - Retrieve the list of all the descendants of an EID

# Contribution

- EIDs that include its history
  - Content changes, history doesn’t!
  - Lineage Preservig EIDs allow to detect deprecated EIDs **locally**
  - No need for querying the OKKAM node
  - Definitive advantage in a fast evolving enviroment and as long-term solution

# Scenario


C
F
Local Entity Repository
Sender
F
C
E-mail
?
A
Local Entity Repository
Receiver
A
C
B
OKKAM node

# Outline

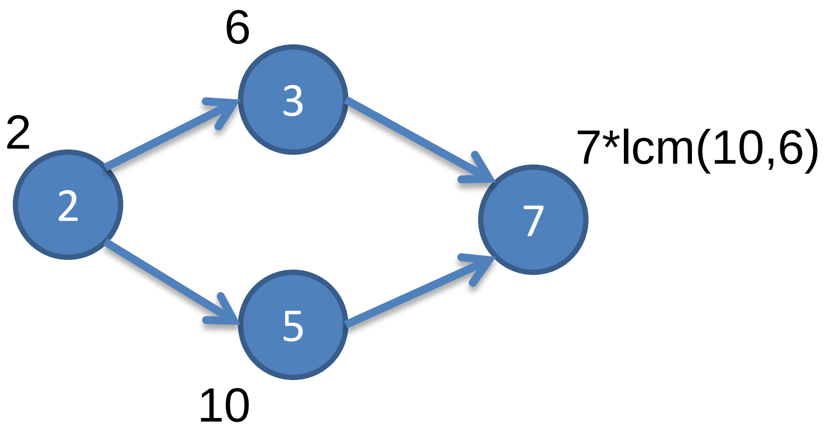
- – Project context: OKKAM
- – Setting: Revision of Entity Identity Decisions
- – Problem Statement
- – Example Scenario
- – **Prime Number Labeling Scheme for DAGs**
- – Lineage Preserving Entity ID
- – Discussion & Future Work

# Prime Numbers Labelling Scheme for DAGs

– DAG: G(V,E)

– Algorithm:

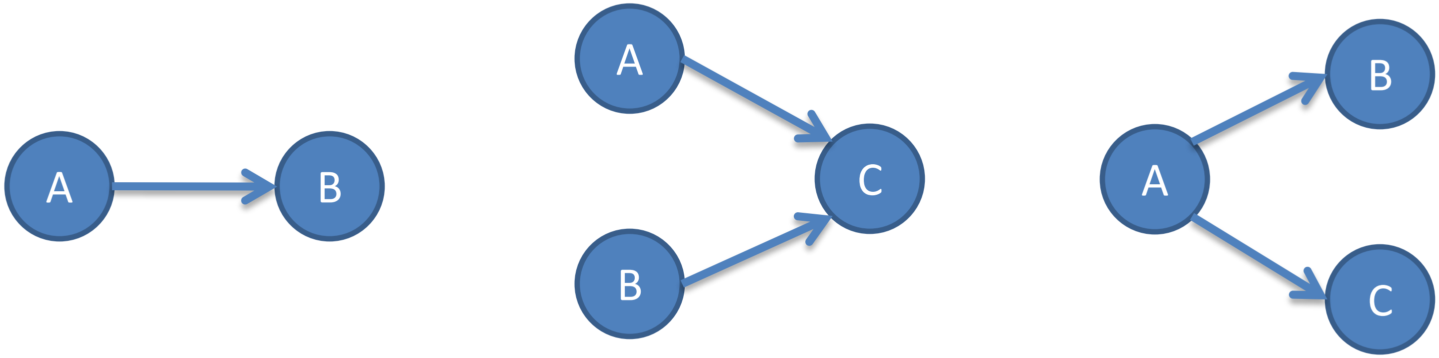
- Assign a unique prime number *p* to each *v* in *V* **self-label**
- Label each *v* with ($p$* the least common multiplier of its ancestors’ label) **ancestor-label**



**Adapting Prime Number Labeling Scheme for Directed Acyclic Graphs**
G Wu, K Zhang, C Liu, J Li - Database Systems for Advanced Applications, 2006 Springer

# LPID

- The history of an entity can be represented as a DAG
- Prime number labeling can be used as a basis for creating the entity Ids



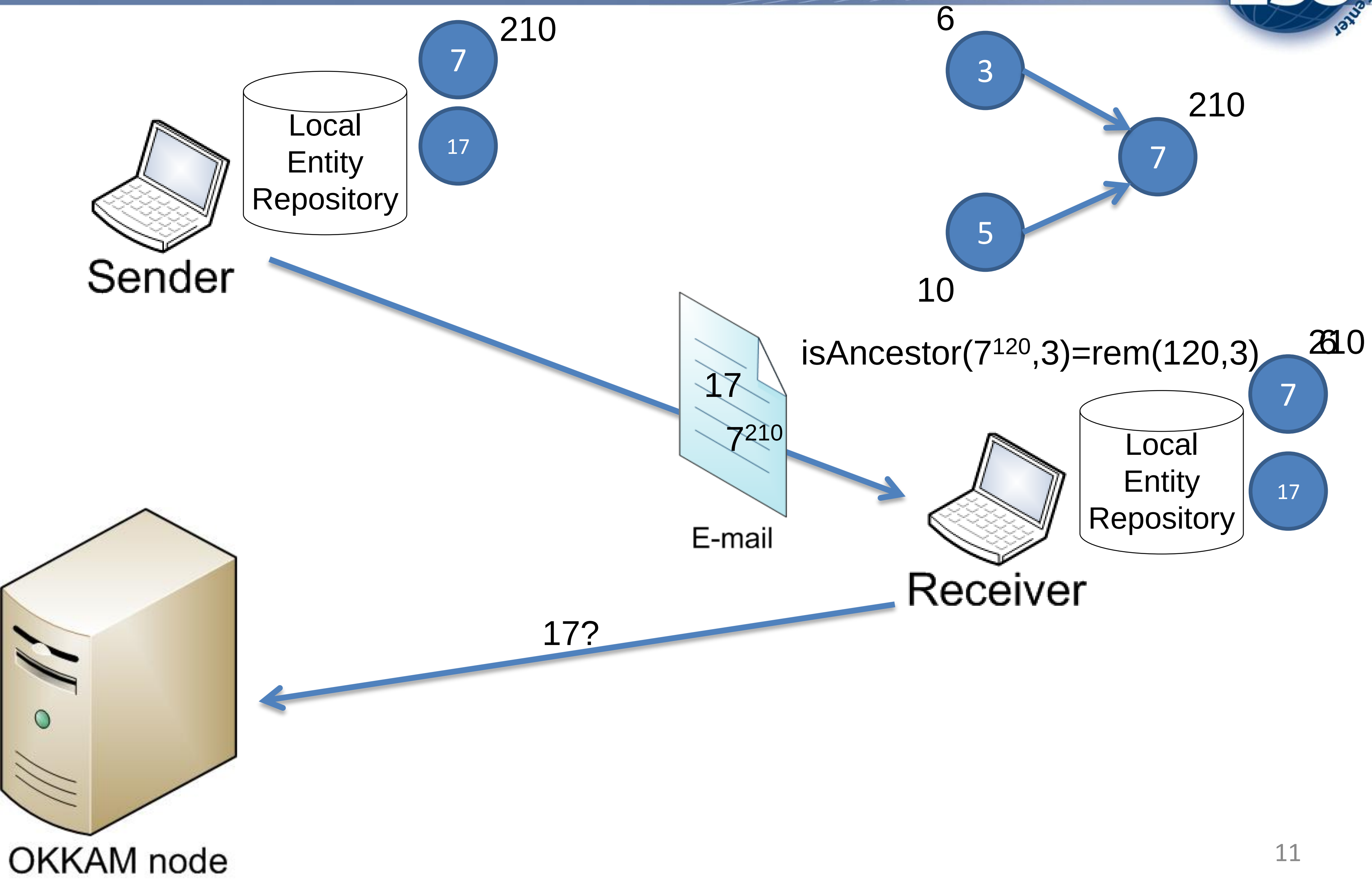
Deprecation

Merge

Split

- For each entity *e* we assign $i=(e_{self}, e_{ancestor})$

# Application of LPID



210

7

17

Local Entity Repository

Sender

6

3

210

7

5

10

E-mail

17

$7^{210}$

isAncestor($7^{120}$,3)=rem(120,3)

210

6

7

17

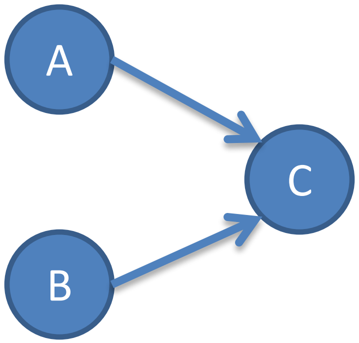
Local Entity Repository

Receiver

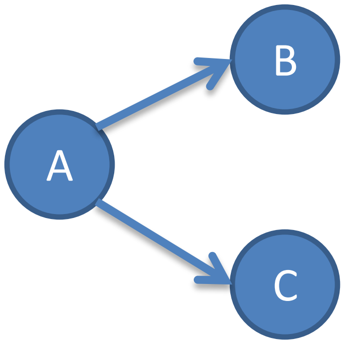
17?

OKKAM node

# Merge vs Split



Merge



Split

- If the system knows A and receives C
- It can replace A with C

- If the system knows A and receives C
- It only knows that A is deprecated

# Discussion

- Being able to identify locally deprecated EIDs allows to:
  - Reduce the number of requests to OKKAM nodes
  - Assure that an entity is represented by only one EID in the local repository
- EIDs based on prime numbers are not intuitive for end-users

# User Friendly EIDs

– Using DNS for resolving EIDs
  - We can encode LPID in a DNS node for associating user-friendly name to entities
  - The max number of unique ancestors we can encode is, at least, 483 (estimation)

  [RFC1035 (DNS standard)]

# Future Work

- In-depth analysis of space/time requirements
- Comparison with other approaches
- Simulation for studing the growth of LPID size with the number of entities and operations considered
- Simulation for studing the space limitation while using DNS

# The End

Thanks