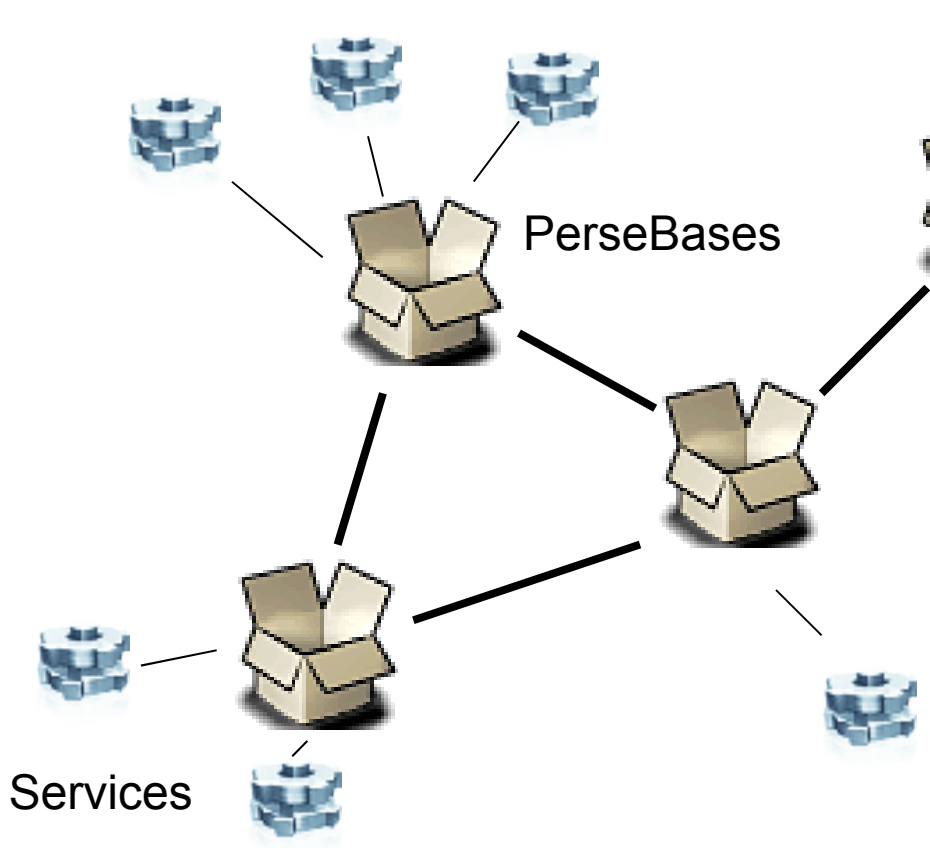
Distribution and Management of Service Descriptions in Pervasive Service Environments

A way to enable PerSE for service adaptation and improved p-graph calculation

PerSE architecture (ServiceNet)



- PerSE Entities
  - Services
  - PerseBases
    - Managing services
    - Calculating p-graphs
    - Managing Connections

Motivation 1

- Each service has description (known by parent PerseBase)
- To use the ServiceNet (to calculate a ServiceGraph), the PerseBase must have an initial knowledge about the available resources in the network
  - The usable services and their description
  - The corresponding PerseBases
  - The connections between the PerseBases
  - Some knowledge about other graphs in the same context

Motivation 2

- When an environment of a service changes, it adapt (itself)
  - Change of the service's description
- Method needed to distribute the services' (entities') descriptions over the ServiceNet
  - Actions needed:
    - Bases discovery
    - Service discovery
    - Knowledge discovery

Requests 1

- Distributed knowledge
  - Two types of knowledge :
    - Description knowledge
    - Usage history knowledge
  - no central instance
    - Avoids bottlenecks, speed up calculation, fault-tolerance
- “Sufficient” information distribution
  - Every PerseBase should know all the information it’s interested in
- Fast adaptation of the global meta-knowledge
  - The services environment is fast changing

Requests 2

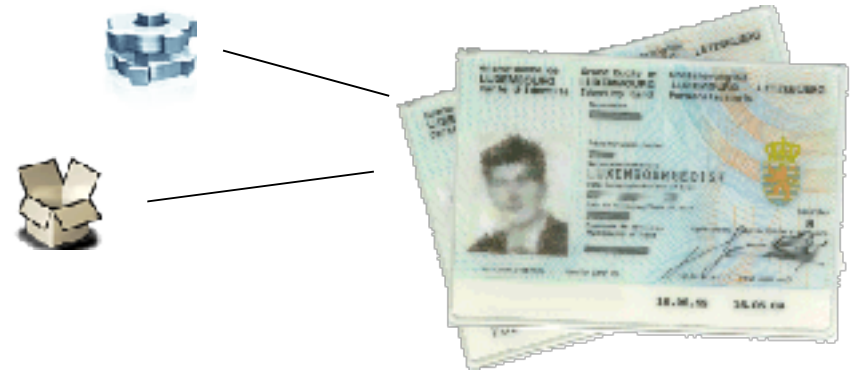
- Minimize amount of meta-information transport
  - Small bandwidth of connections in some configurations
- Minimize calculation time for description management
  - Especially for services implemented on smart (narrow-chested) devices
- Fast access on interesting current service description
  - Calculate a p-Graph on any PerseBase in time.
- Description-Format itself should be easily (and compatible) extensible
  - To hold specific information of new services

Related Work

- Web Services
  - Service Registry
  - UDDI
- Computing Grids
  - OGSA-DAI
  - DAISGR

Describing an entity

- Each entity owns an InformationCard
  - Entity types:
    - Service
    - PerSE Base



- InformationCard:
  - Entity-Id (EID, CLSID)
  - Description (static information; owned by service)
  - Context (dynamic information; owned by parent base)

InformationCard content

- Entity-ID (EID)
- Entity-Type (Service, PerseBase)
- Card-Timestamp
- Validity Time
- Description and Context:

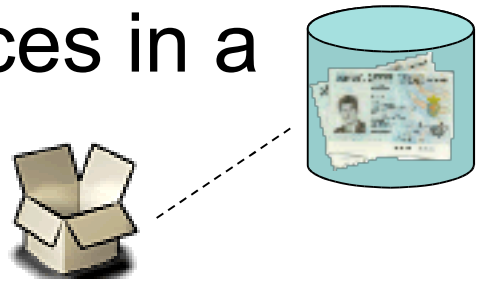
Connection:
- speed
- failure rate
- connected PerseBases
- …

PerseBase:
- Location
- …

Service:
- type of offered service
- ServiceHome
  - (corresponding PerseBase)
- connection speed to base
- minimal response time
- behavior (SDL-description)
- …

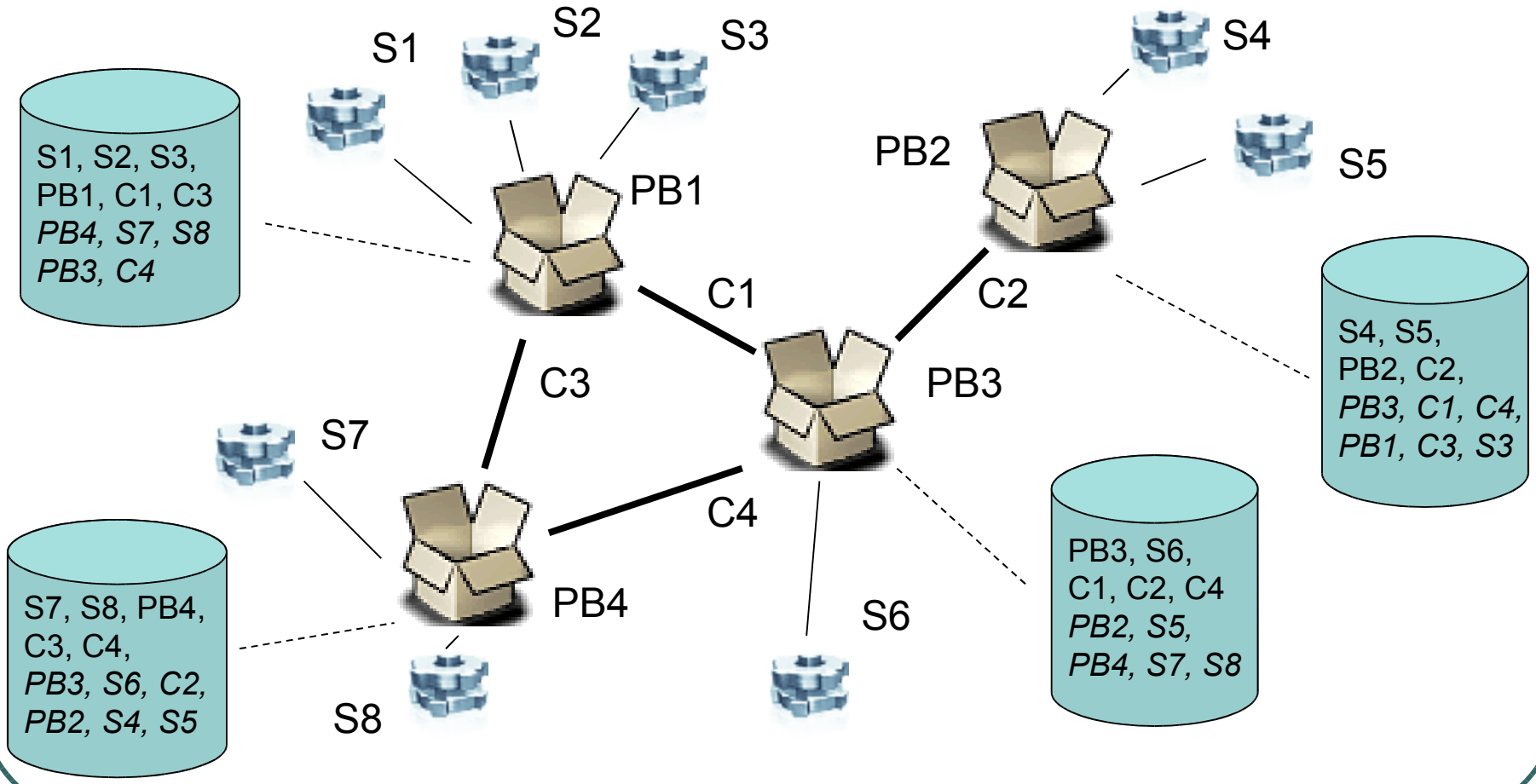
Information filing 1

- Each PerseBase stores the InformationCards of “it’s” services in a database (*InformationDeck)*



- In addition, the InformationDeck contains the InformationCard of all other services and entities, the PerseBase is interested in to calculate a p-graph

Information filing 2



Information Distribution 1

- Two modes of description distribution
  (Description Exchange Message, DXM)
  - Full DXM
    - Contains the complete InformationCard as stored in the emitting PerseBase's InformationDeck
  - Partial DXM
    - Contains the EID, the Card timstamp and the records of the InformationCard that have changed
    - Contains a "Previous timestamp" field to assure data integrity

Information Distribution 2

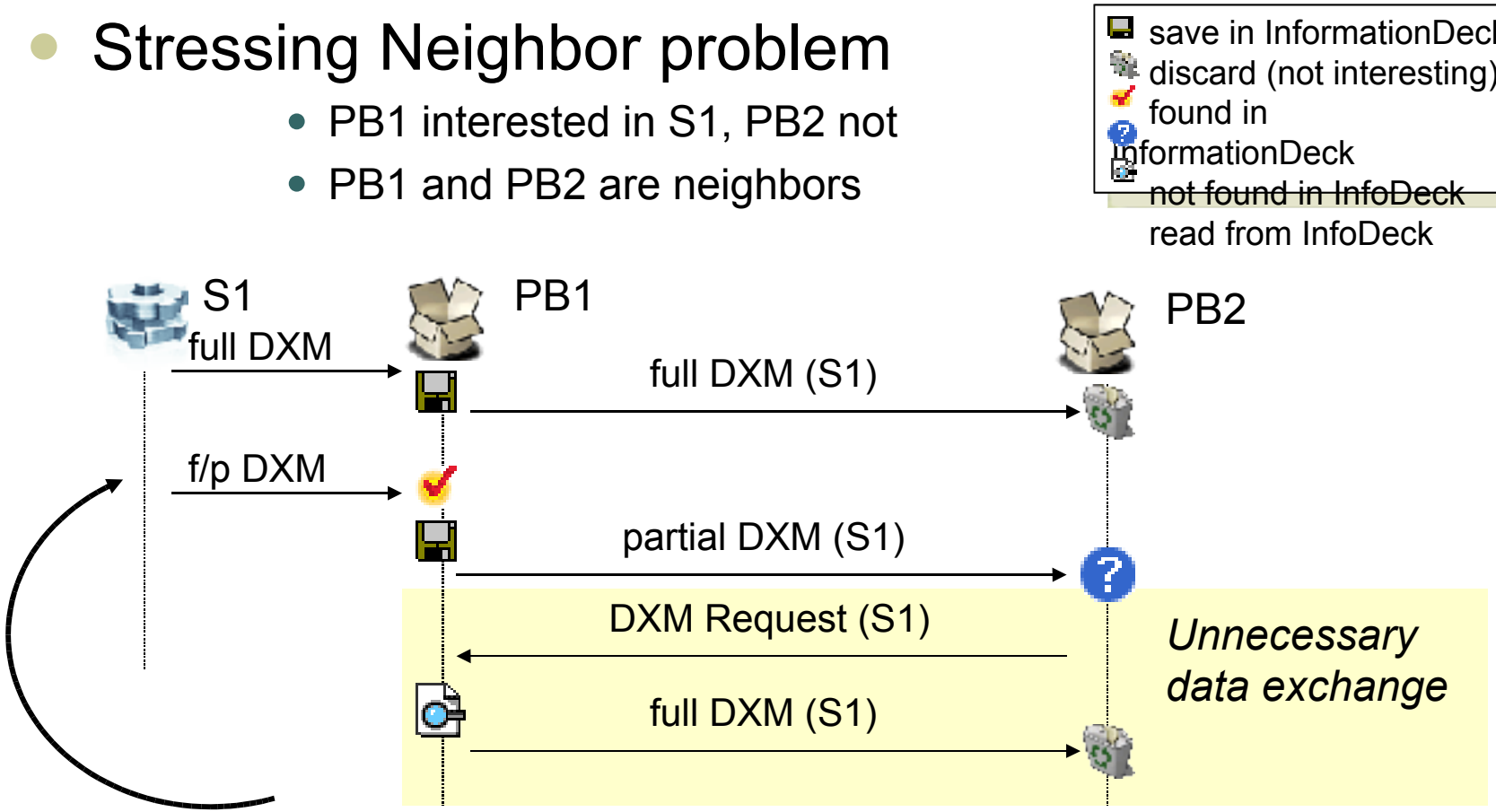
- Standard Propagation: Sparse flooding
  - Every PerseBase floods information, that has been changed in their InformationDeck to all other known PerseBases
  - Received information that is not “interesting” will not be stored (and therefore not further flooded)
    - Information too old
    - Service too far away
    - Service two slow
    - Service not matching quality criteria…
    - Already enough similar (and better) services in the database
  - EID + Card TS form a unique ID for each DXM

Information distribution 3

- DXM Request
  - A PerseBase can request full DXMs for given EIDs or all known entities from its neighbors
  - Use cases
    - Initial InformationDeck creation
    - Get full InformationCard for a received partial DXM with unknown EID
    - Information about HomeBase, intermediate PBs and connections between them for a new interesting service
- Each PerseBase has a sparse knowledge of the complete ServiceNet

Information Distribution 4

- Stressing Neighbor problem
  - PB1 interested in S1, PB2 not
  - PB1 and PB2 are neighbors



Information Distribution 5

- Solutions for the Stressing Neighbor Problem
  - Ignore it
  - Remember, why suppressed the description of which entity and just request full DXM when a record changed that justified this decision
    - But: You store a lot of rubbish, you're not interested in, so management needed to maintain this mountain of garbage (LFU strategies, validity time…)
  - Tell neighbors which entities I'm not interested in until which field change (Deny message)
    - Just move the problem mentioned above to the neighbors, more difficult to "change your mind".

Future Challenges 1

- There are a lot of open questions:
  - Which description records on the InfoCard?
  - How to get the record values (measurement, keep-alive-tests, …)
  - How to interpret these values, how to compare services?
  - How to decide, which entity is “interesting” and which not?

Future Challenges 2

- Structure the description of an entity on an InformationCard just in one level or hierarchically?
- Introduce generic “InformationCard templates” to reduce transmitted data?
- Physical limits (Memory for InformationDeck exceeds)?
- Security?
  - How to assure integrity of DXMs?
  - How to assure authentication of DXMs? Signing?
  - Different “Level of trust” based on PB’s comput. power?

Future Challenges 3

- How to structure and manage the InformationDeck of each PerseBase?
  - Easy to update and maintain
  - Small (limited resources for PerseBases on pervasive devices)
  - Prepared for easy service graph calculation