EUROPEAN MIDDLEWARE INITIATIVE

Catalogue synchronization (Update)

Fabrizio Furano (CERN IT-GT-DMS)

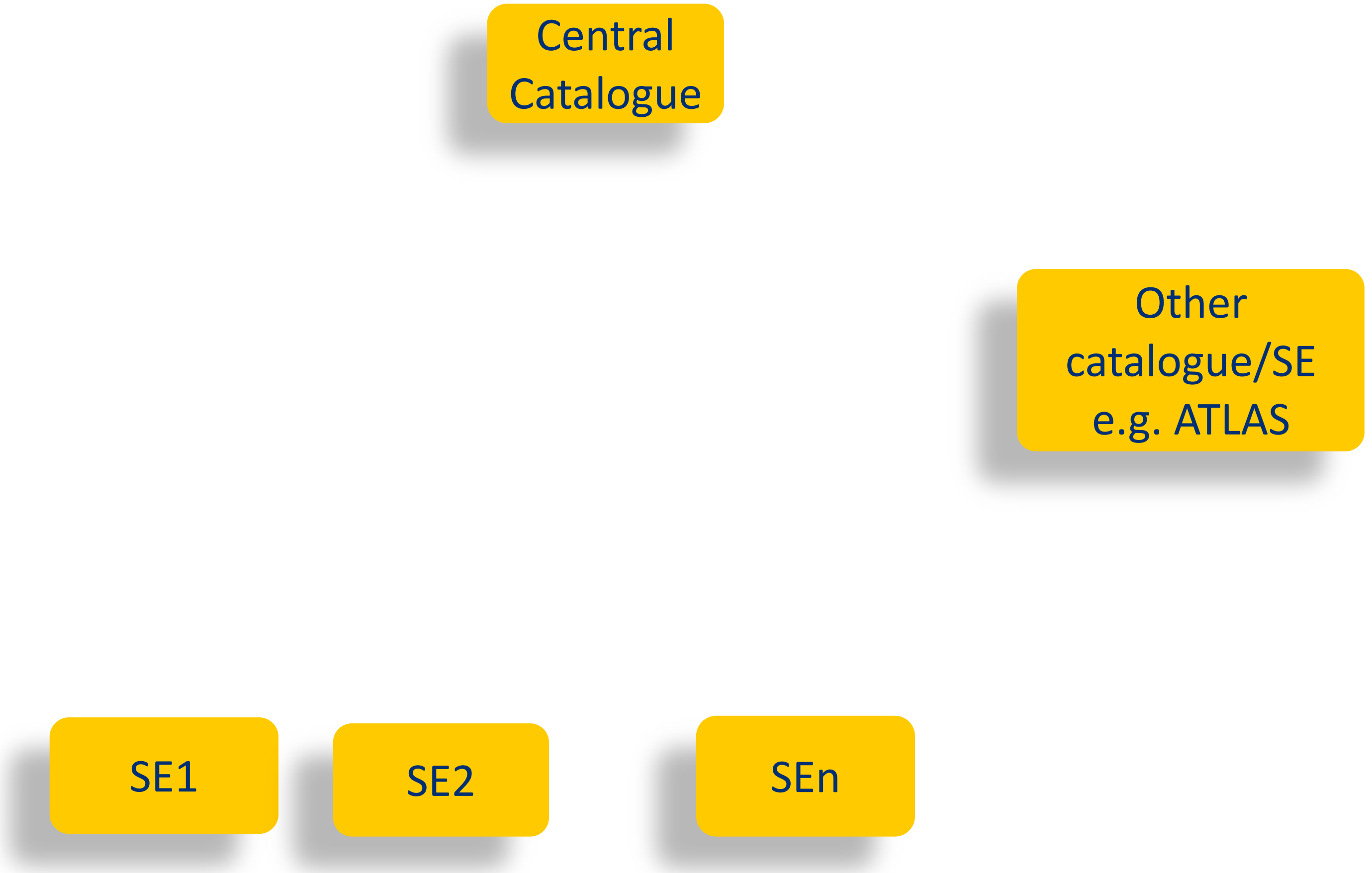
The problem

- Various catalogues keep information that is related
  - E.g. LFC keeps info about the content of remote Storage Elements, each one with its own catalogue
    - A change in the permissions of a file in LFC is not automatically reflected by the peripheric catalogue
    - If a SE looses a file, the LFC does not know
    - If a new file is not correctly registered -> dark data
- Keeping them in sync is a very hard problem
- Namespace scanning for diffs is an expensive workaround

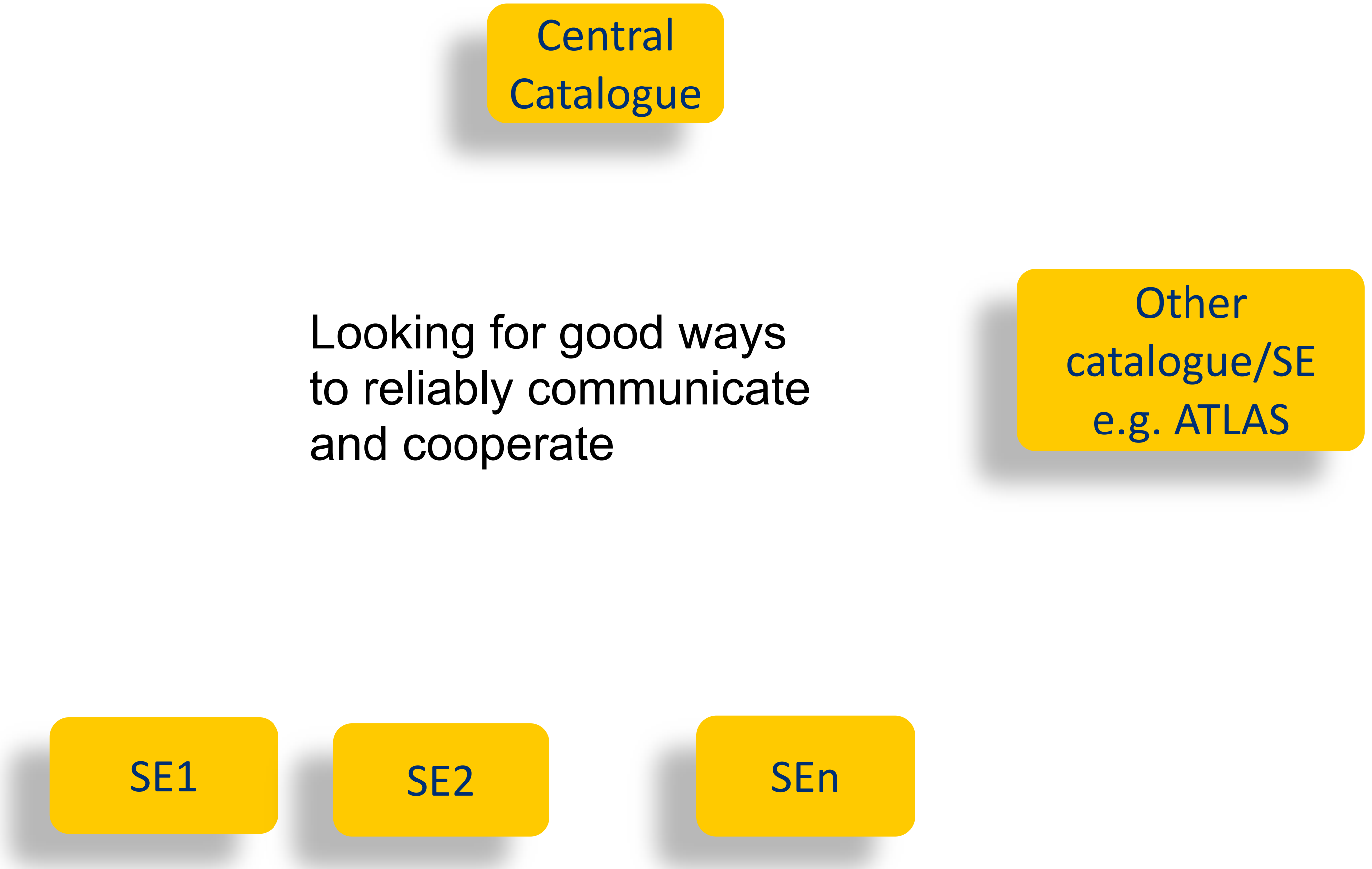
The idea

- Make the various catalogues/SE able to talk to each other
  - In order to exchange messages that keep them synchronized in realtime
  - 2 directions:
    - Central Catalogue->SE (downstream)
      - e.g. to propagate changes in the permissions
    - SE->Central Catalogue (upstream)
      - e.g. to propagate info about lost and missing files

Communication

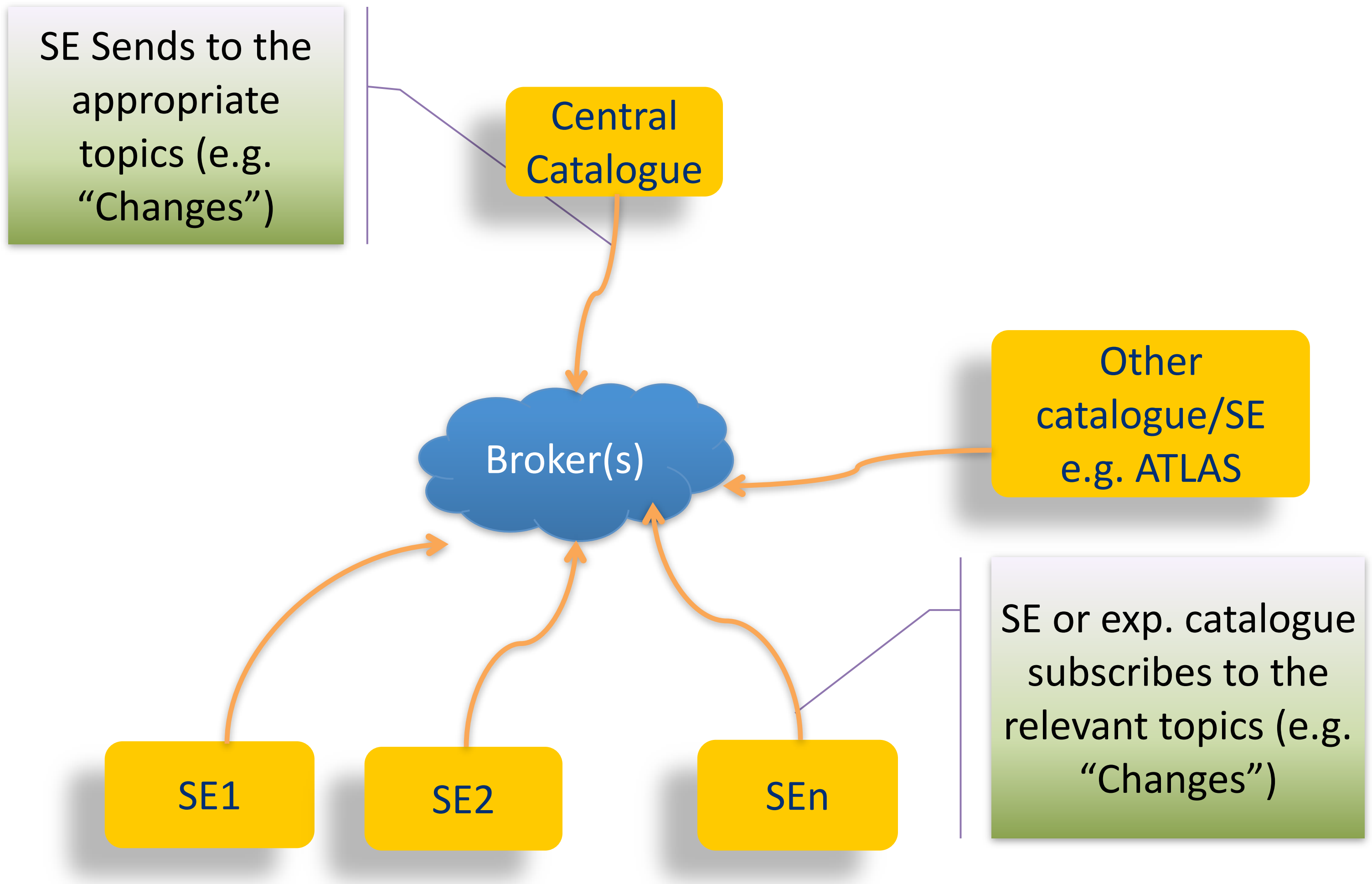


Communication

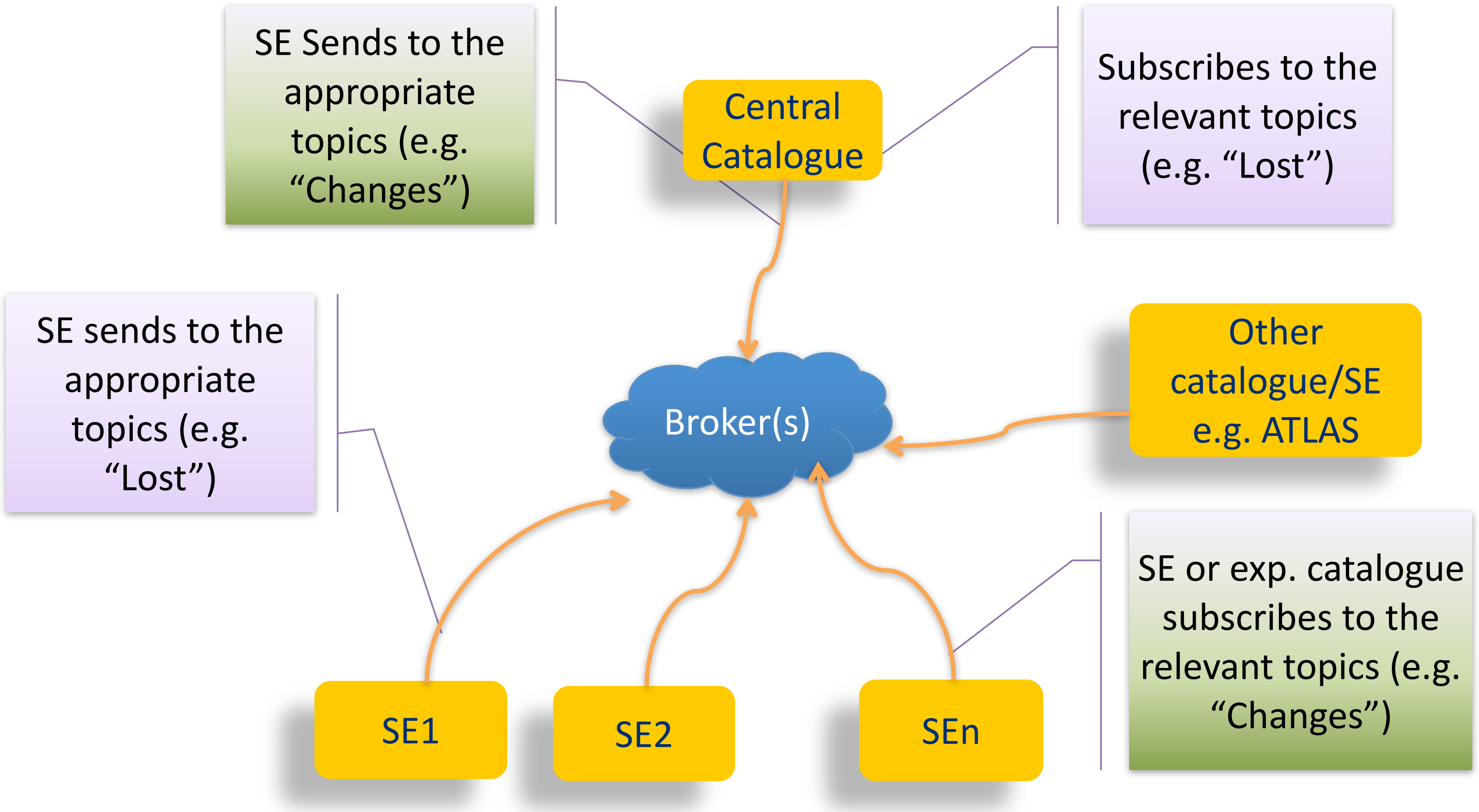


Looking for good ways to reliably communicate and cooperate

Communication

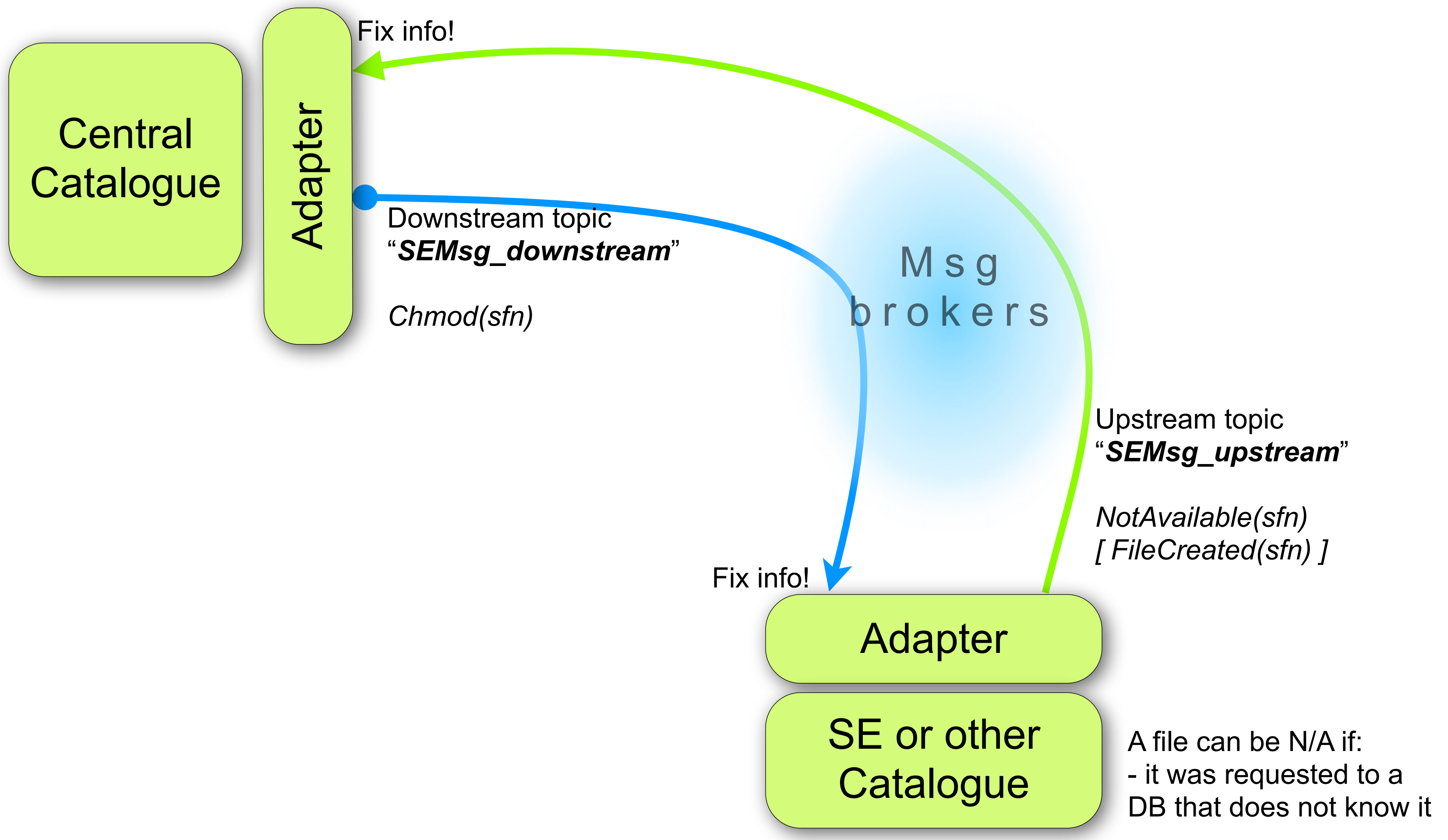

SE Sends to the appropriate topics (e.g. "Changes")
Central Catalogue
Broker(s)
Other catalogue/SE e.g. ATLAS
SE or exp. catalogue subscribes to the relevant topics (e.g. "Changes")
SE1
SE2
SEn

Communication



SE Sends to the appropriate topics (e.g. "Changes")
Central Catalogue
Subscribes to the relevant topics (e.g. "Lost")
SE sends to the appropriate topics (e.g. "Lost")
Broker(s)
Other catalogue/SE e.g. ATLAS
SE or exp. catalogue subscribes to the relevant topics (e.g. "Changes")
SE1
SE2
SEn

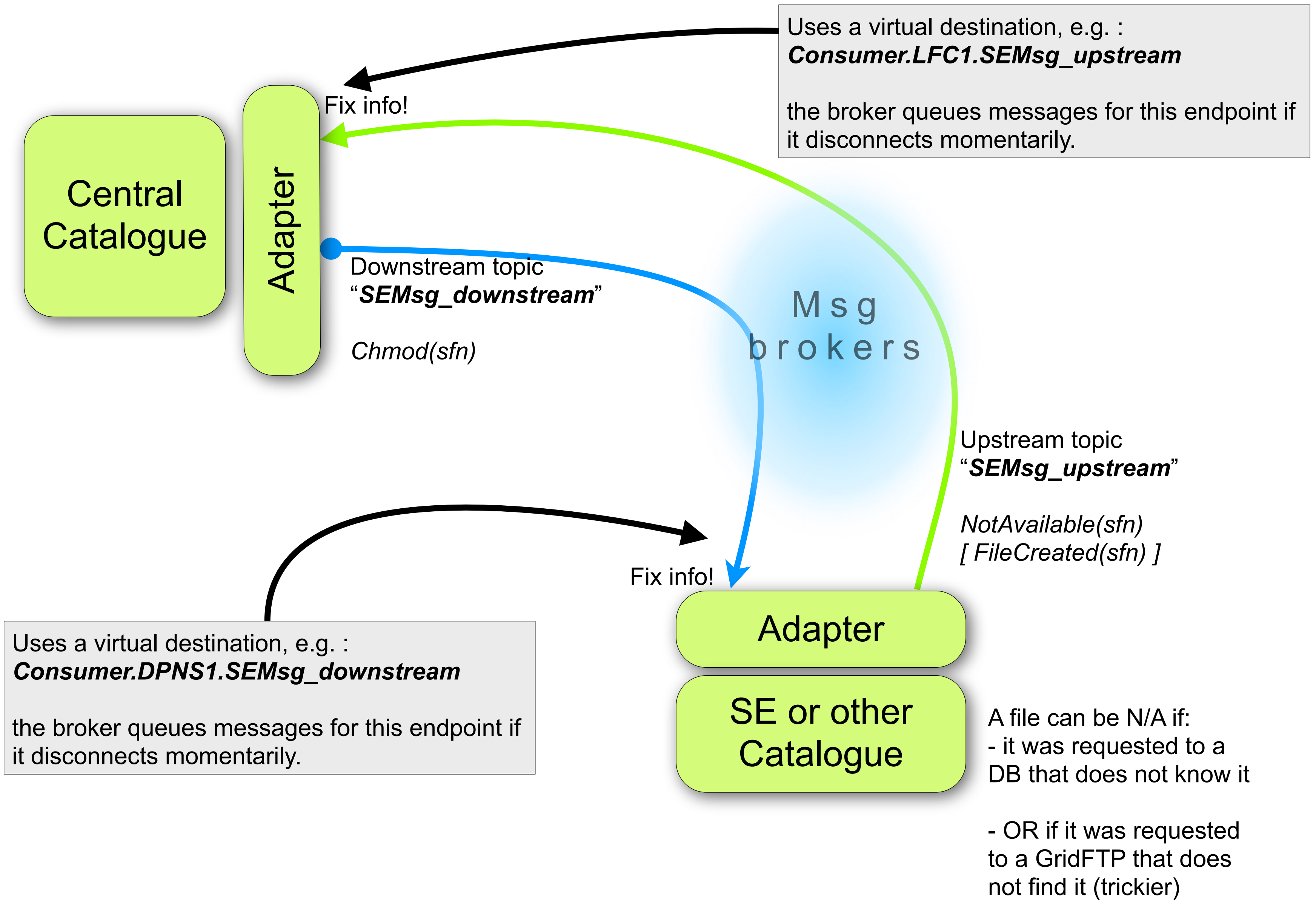
Simplified architecture



A file can be N/A if:
- it was requested to a DB that does not know it

- OR if it was requested to a GridFTP that does not find it (trickier)

Simplified architecture



Uses a virtual destination, e.g. :
Consumer.LFC1.SEMsg_upstream

the broker queues messages for this endpoint if it disconnects momentarily.

Uses a virtual destination, e.g. :
Consumer.DPNS1.SEMsg_downstream

the broker queues messages for this endpoint if it disconnects momentarily.

A file can be N/A if:
- it was requested to a DB that does not know it

- OR if it was requested to a GridFTP that does not find it (trickier)

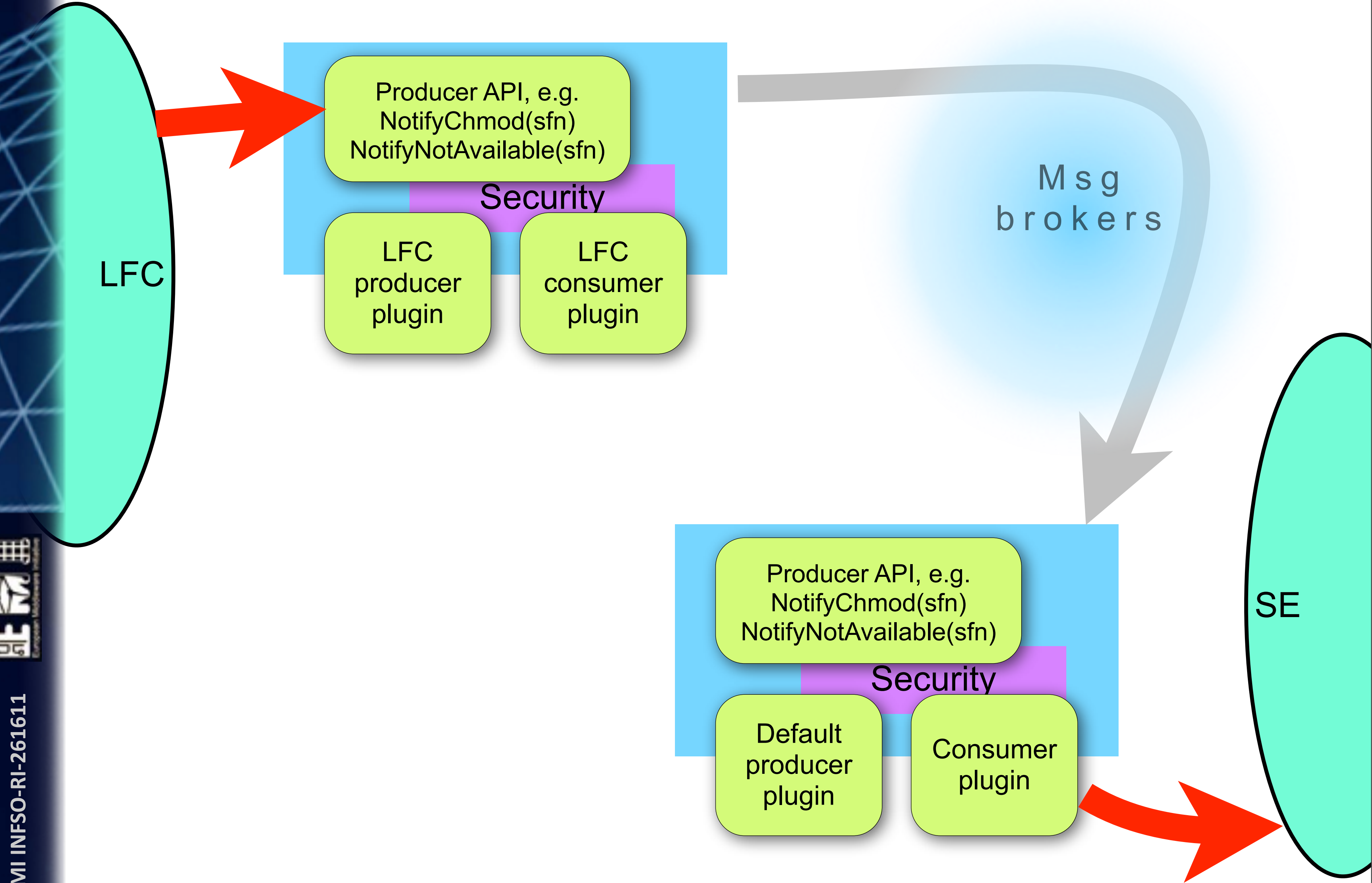
Syncat protocol -> SEMsg

- Full specification of the messaging-based protocol, for others to do the same
- The current implementation is a component called SEMsg
  - Native bulk operations
  - Built to be robust, efficient and easy to integrate
  - Plugin-based (ev. with “null” plugins), loaded at runtime (dlopen)
    - A plugin that performs actions (in the catalogue) when a message comes
    - A plugin that performs SE(Catalogue)-specific actions when a message has to be sent through the API
    - 5 plugins available by now: LFC producer+consumer, DPM producer+consumer, Python consumer
  - Provides a configurable daemon that consumes and dispatches the notifications
  - Provides various kinds of API
    - a CLI to manually send notifications
    - a simple C/C++ API to be used in external systems
    - a very simple and efficient Python consumer API
  - **hides the technicalities of message composition and of the security implementation**
  - ***crafted to avoid bringing in dependencies***
  - The same tools are used for the LFC and DPM prototype, loading different sets of plugins
    - Hence, more sets of plugins can be added, to talk to other systems
  - Easier task for the developer of the integration, no knowledge about messaging/security/ X509 crazy details required
    - Everything is documented, though

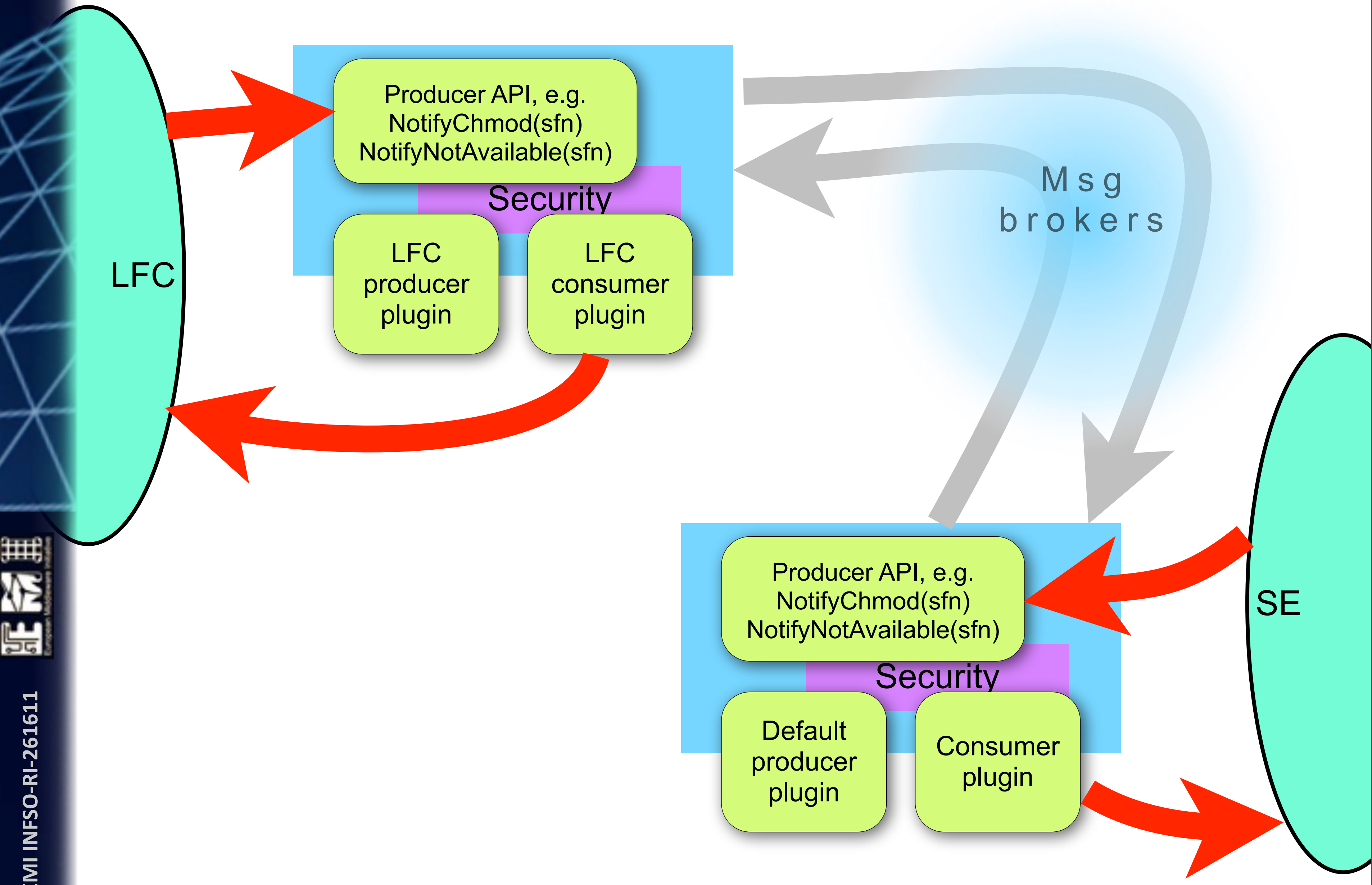
Detail - SEMsg plugins

Detail - SEMsg plugins

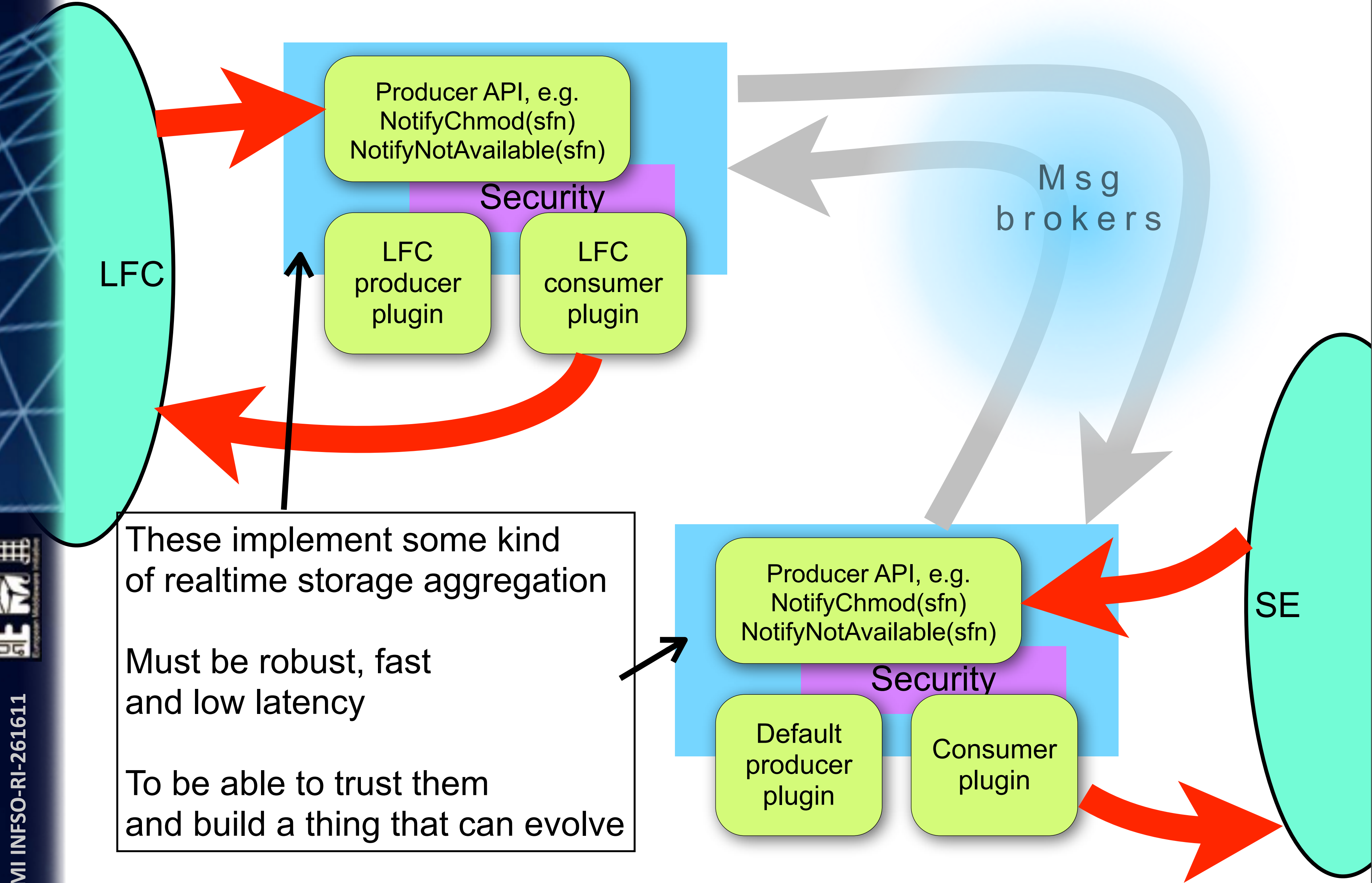
Detail - SEMsg plugins



Detail - SEMsg plugins



Detail - SEMsg plugins



These implement some kind of realtime storage aggregation

Must be robust, fast and low latency

To be able to trust them and build a thing that can evolve

The Python plugin

- One more consumer plugin, in the SEMsg distribution, with the DPM and LFC ones
- Associates python funcs to SEMsg notifications
  - Fully configurable in the SEMsg config file and generic
    - e.g. The notification FileNotAvailable invokes the function 'func1' from the module 'module1' passing its content as individual simple parameters
  - Fast: invokes natively the Python C API, no python spawns are performed
  - Benefits from the SEMsg structure, e.g. the security and the bulk messages
    - ***The Python script only deals with the action to be performed, not with the tech details of the messaging and of the X509 security***
- Tested with Python 2.4, 2.5, 2.6

Dev update: v1.2.0 --> v1.3.0

- Work in progress (90%) about:
  - Whitelisting (an endpoint may choose the endpoints it trusts) based on the existing X509 implementation (since 1.2.0)
  - SEMsg can securely propagate the DN of the requestor of the action that triggered the notification
  - The receiving endpoint can apply its authorization mechanism to this info
  - DPM/LFC supporting this
  - Update of the docs
  - Credits to Gergely Debreczeni (testing and suggestions)

What's next

- Sync with the other SE developers (STORM, dCache)
  - Update on the preferred way to integrate with it
  - Previously agreed schedule:
    - Storm/dCache as producers of notifications:
      - Toy prototype/Proof of concept (these days)
        - **Action: Storm/dCache teams**
        - Install SEMsg from https://svnweb.cern.ch/trac/lcgdm/wiki/Dpm/Dev/Components
        - Point it to valid host/svc certificates
        - Find WHERE in the code to send the notf
        - Invoke the CLI from there
      - Development of a Java API for SEMsg (After the toy prototype)
        - **Action: GT**
      - Utilization of the new Java API inside Storm/dCache
        - Instead of the CLI invokations
        - **Action: Storm/dCache teams**
    - Storm/dCache as consumers of notifications: apparently postponed
      - My advice is to use the available Python plugin for a simple prototype

Conclusions

- Making catalogues and SEs interact seems a good way to attack the consistency problem
  - It's a form of realtime interaction between SEs and catalogues
  - By definition, it won't mathematically kill the inconsistencies, but will help making a better system
  - Will be interesting to explore the possibilities of the technology and of the implementation. Many interesting ideas are being proposed.
- SEMsg is available as a pre-release until EMI-2
- Protocol and SEMsg documentation in the Wiki
    - https://twiki.cern.ch/twiki/bin/view/EMI/EmiJra1Syncat
- Packages here:
    - https://svnweb.cern.ch/trac/lcgdm/wiki/Dpm/Dev/Components
- Integration with the other SE developers is starting
- Feedbacks are welcome
- The messaging (test) infrastructure and the tools seem really OK

EUROPEAN MIDDLEWARE INITIATIVE

Thank you

EMI is partially funded by the European Commission under Grant Agreement INFSO-RI-261611