DDS

LOBBY BASED DEPLOYMENT

Andrey Lebedev (GSI)
Anar Manafov (GSI)

2017-11-10
Alice Offline Week @ CERN

Highlights

- Token based authentication
- Lobby based deployment

Token based authentication

Assumptions & Prerequisites

- Meet minimum security requirements, such as:
  - prevent erroneous connection attempts from DDS agents of expired sessions or agents of other users,
  - prevent connection attempts on DDS UI channels from DDS commands and user apps which don’t use privileges of the current DDS user.

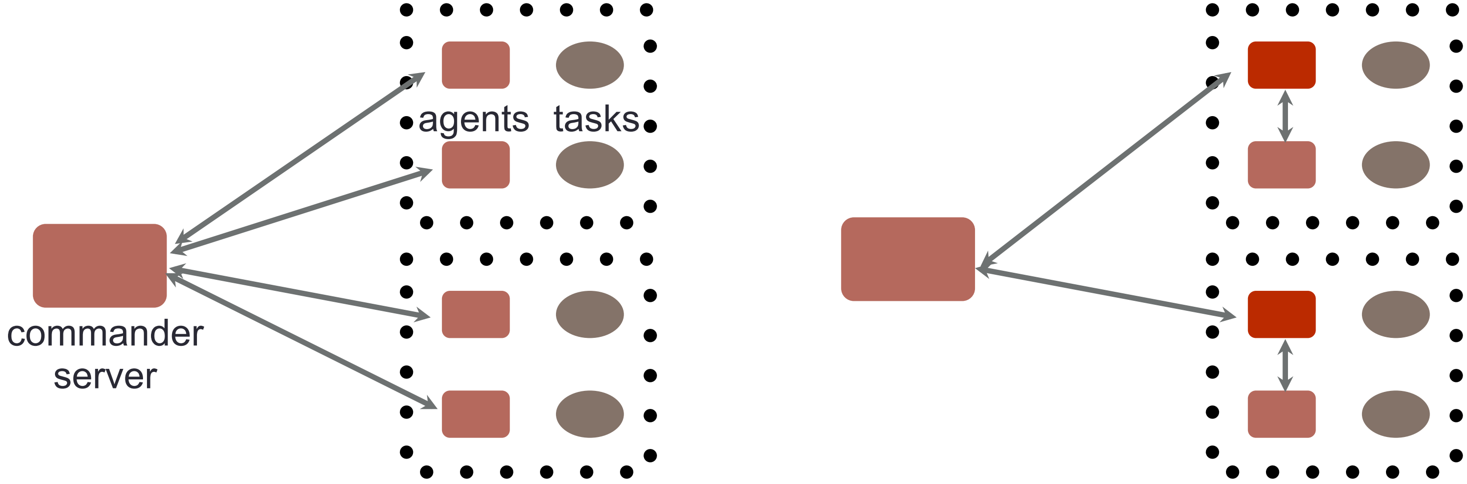
Token based authentication

Current solution

- DDS commander generates a session ID (SID).
  - SID is represented by a GUID (globally unique identifier).
- A file with SID is packed as a part of DDS WN packages.
  - Agents spawned from a given package can connected back only using that SID.
- SID is stored on the commander's host in the "~/.DDS" directory. This helps DDS commands and user apps to connect to DDS commander.
- RMS plug-ins receive that SID in a command line when get spawned by the commander.

Lobby based deployment

- DDS Agents of a given user on one host is a **lobby**.
  - **Lobby Leader** is an agent who has a direct network connection to commander;
  - All other agent are **lobby members** communicating with the commander via the leader.
- Several prototypes have been developed
  - Finally we came up with the current **solid** design



Shared memory communication

- Shared memory channel
  - Exactly the same event-based API as DDS network channel;
  - Duplex and many-to-many communication;
  - Asynchronous read and write operations;
  - dds-protocol;
  - Efficient message forwarding.
- Implementation
  - **boost::message_queue**: message transport via shared memory;
  - **dds-protocol**: message definition, encoding, and decoding;
  - **boost::asio**: the proactor design pattern and an efficient thread pool.

Communication channels

- Network and shared memory channels;
- Unified event-based API for application and protocol events;
- Compile time check for errors where possible;

Subscribe to messages

```
client->registerHandler<cmdUPDATE_TOPOLOGY>(
    [](const SSenderInfo& _sender,
       SCommandAttachmentImpl<cmdUPDATE_TOPOLOGY>::ptr_t _attachment) {
       // User’s code
});
```

Subscribe to channel events

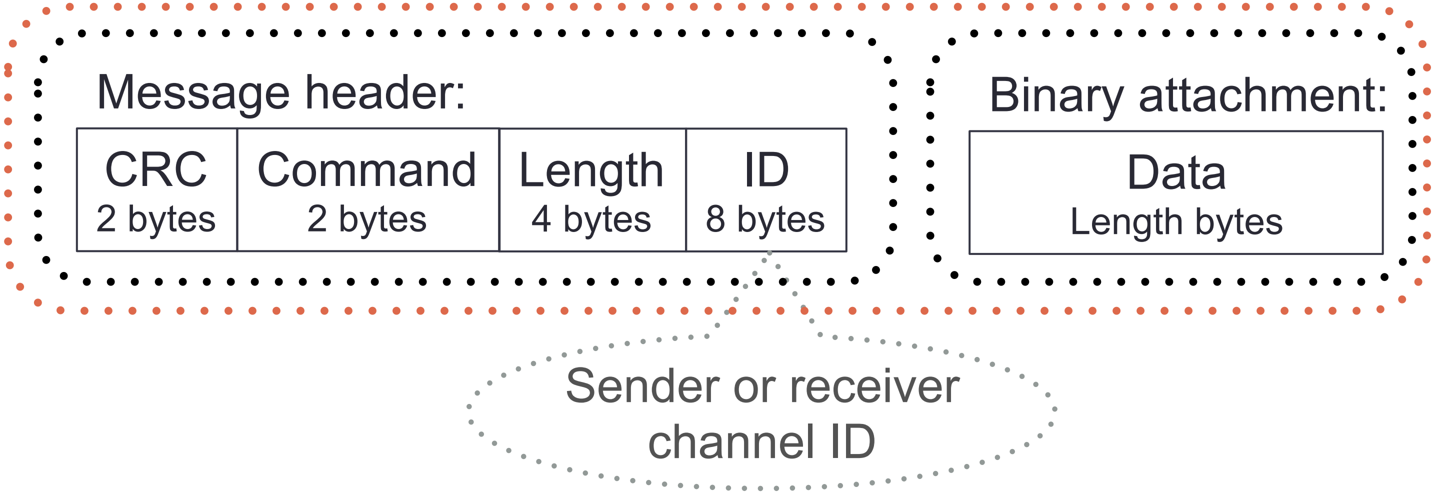
```
client->registerHandler<EChannelEvents::OnConnected>(
    [](const SSenderInfo& _sender) {
       // User’s code
});
```

Subscribe to messages

```
BEGIN_MSG_MAP(CInfoChannel)
    MESSAGE_HANDLER(cmdREPLY_PID, on_cmdREPLY_PID)
    MESSAGE_HANDLER(cmdREPLY_AGENTS_INFO, on_cmdREPLY_AGENTS_INFO)
END_MSG_MAP()
```

dds-protocol news

- Message header:
  - Add protocol header ID (PHID) containing either sender or receiver channel ID



Message header:

CRC 2 bytes	Command 2 bytes	Length 4 bytes	ID 8 bytes

Binary attachment:

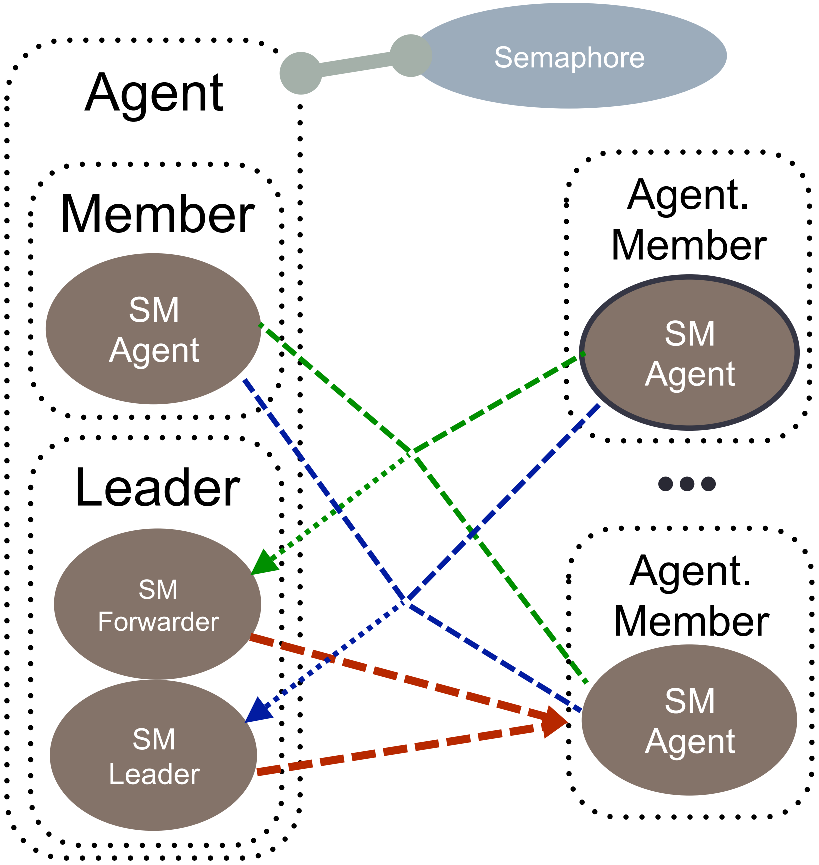
Data Length bytes

Sender or receiver channel ID

- One of the use cases for PHID is **message forwarding**:
  - A forwarding channel receives raw messages and pushes them further. For instance, from network to shared memory;
  - No extra memory allocations, encoding and decoding of messages are performed;

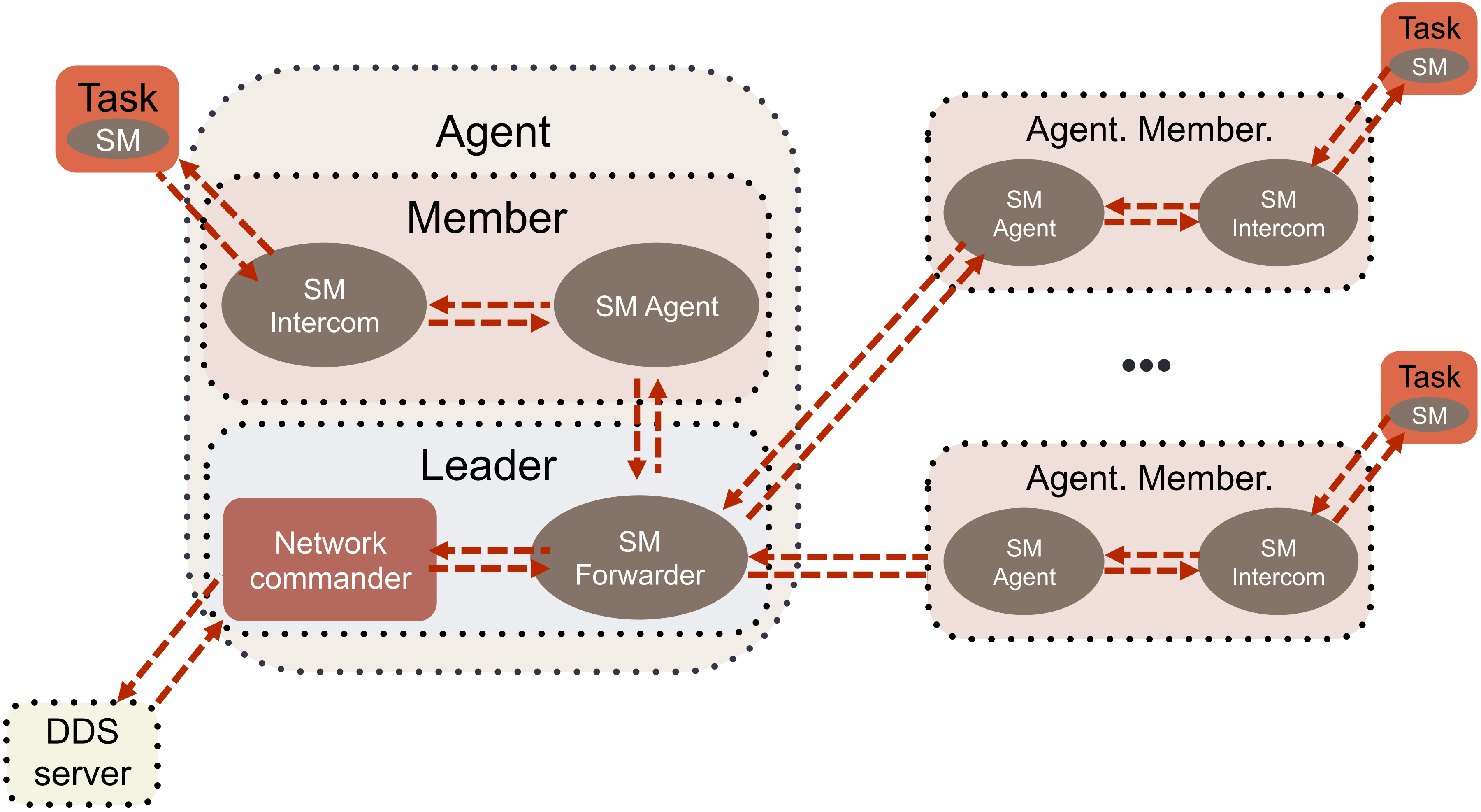
Lobby leader election

A lobby leader election: “First in takes all”.

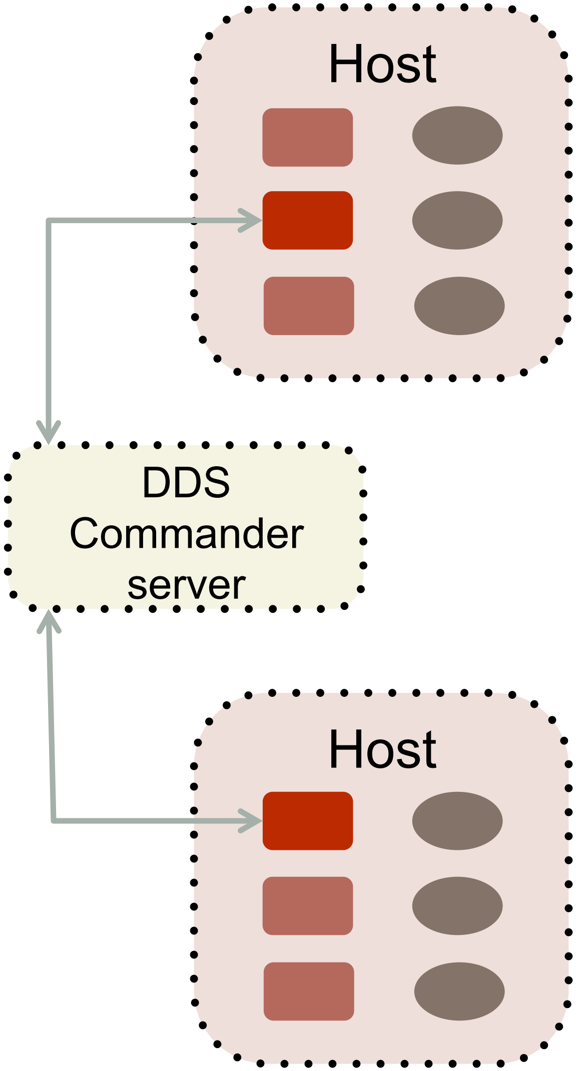


- A leader is the one who first owns a SID semaphore;
- Each lobby member sends a special message to the leader with its connection information;
- The leader opens a channel and sends back a confirmation;
- Than a member sends a “lobby member handshake” message to Commander via SM Forwarder channel of the leader;
- Commander adds the agent to the list of approved agents;
- The communication is established.

A lobby


Task
SM
Agent
Member
SM Intercom
SM Agent
Leader
Network commander
SM Forwarder
DDS server
Task
SM
Agent. Member.
SM Agent
SM Intercom
•••
Task
SM
Agent. Member.
SM Agent
SM Intercom

Lobby based deployment



- One network connection per host;
- Local communications only via DDS shared memory channels;
- Unified agents and an unified lobby leader election;
- Efficient message forwarding;
- dds-protocol via network and shared memory channels;
- Handshake- and token-based authentication;

Roadmap

- **v1.8** – developer release (**ready**)
  - Lobby based deployment and all DDS features are stable;
  - Working on tests for edge cases;

- **v2.0** - stable release
  - A public release of the new architecture (lobby based deployment);
  - Here's hopping that architecture of DDS v2.x branch remains a mainstream until final production, unless requirements change.

DDS v1.8 (developer release)

- Releases - DDS v1.8
  - http://dds.gsi.de/download.html
- DDS Home site:
  - http://dds.gsi.de
- User's Manual:
  - http://dds.gsi.de/documentation.html
- Continues integration:
  - http://demac012.gsi.de:22001/waterfall
- Source Code:
  - https://github.com/FairRootGroup/DDS
  - https://github.com/FairRootGroup/DDS-user-manual
  - https://github.com/FairRootGroup/DDS-web-site
  - https://github.com/FairRootGroup/DDS-topology-editor