DDS
Dynamic Deployment System

Anar Manafov, Andrey Lebedev
GSI Darmstadt
2016-04-01

Basic concepts

DDS:

- implements a single-responsibility-principle command line tool-set and APIs,
- treats users' tasks as black boxes,
- doesn't depend on RMS (provides deployment via SSH, when no RMS is present),
- supports workers behind FireWalls with only outgoing connection,
- doesn't require pre-installation on WNs,
- deploys private facilities on demand with isolated sandboxes,
- provides a key-value properties propagation service for tasks,
- provides a rules based execution of tasks.

The Contract

The system takes so called "topology file" (topo) as the input.

- Users describe desired tasks and their dependencies using topology files,
- users are provided with a WEB GUI to create topos. Can be created manually as well.

Current status, updates since last meeting

1. Revised from scratch RMS plug-in architecture,
2. new:
   SSH plug-in, **localhost** plug-in, **SLURM** plug-in (beta),
3. new **MESOS** plug-in (alpha) - work of Giulio and Kevin,
4. new DDS intercom API (incl.: key-value, custom cmd, RMS plug-in protocol),
5. multiple DDS protocol improvements:
   - ability to use arrays in messages,
   - improved way of using strings in messages,
   - data consistency check in compile time,
6. core code improvements targeting safer concurrent development,
7. improved Users' manual and API documentation,

and many more other internal improvements and bug fixes.

more details here: https://github.com/FairRootGroup/DDS/blob/master/ReleaseNotes.md

New RMS plug-in architecture
Motivation

Give external devs. a possibility to create DDS plug-ins - to cover different RMS.

Isolated and safe execution. A plug-in must be a standalone processes - if segfaults, won't effect DDS.

Use DDS protocol for communication between plug-in and commander server - speak the same language as DDS.

New RMS plug-in architecture
What plug-in supposed to do?

DDS RMS plug-in tasks:

#1: contact DDS commander server asking for submissions details,

#2: deploy DDSScout fat script on target machines

#3: execute DDSScout on target machines.

That is basically it. The rest DDS will do on its own.
Once DDSScout is executed it will check environment and execute DDS agents, which will contact back to DDS commander server and ask for tasks.

DDS supports dynamic workers, so users can send more agents or stop some. DDS will dynamically add and remove them from active list.

New RMS plug-in architecture
API

```
CRMSPluginProtocol prot("plugin-id");
```

(1)

```
prot.onSubmit([](const SSubmit& _submit) {
       // Implement submit related functionality here.

       // After submit has completed call stop() function.
       prot.stop();
});

prot.onMessage([](const SMessage& _message) {
       // Message from commander received.
       // Implement related functionality here.
});

prot.onRequirement([](const SRequirement& _requirement) {
       // Implement functionality related to requirements here.
});
```

(2)

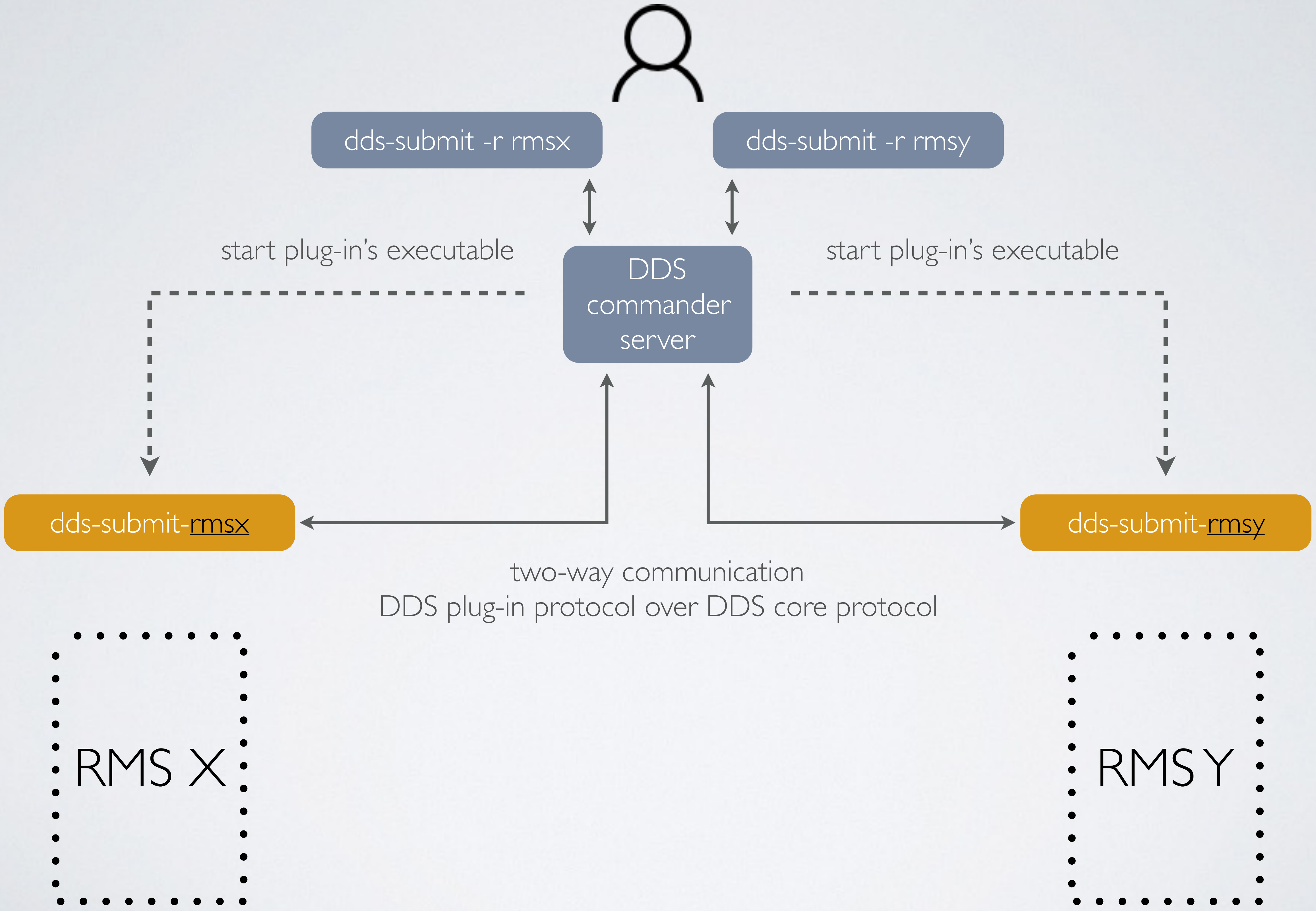
```
// Let DDS commander know that we are online and start listen for messages.
prot.start(bool _block = true);
```

(3)

```
// Report error to DDS commander
proto.sendMessage(dds::EMsgSeverity::error, “error message here”);

// or send an info message
proto.sendMessage(dds::EMsgSeverity::info, “info message here”);
```

New RMS plug-in architecture



New RMS plug-in architecture

```
$ dds-submit -r localhost -n 10



dds-submit: Contacting DDS commander on pb-d-128-141-130-162.cern.ch:20001 ...
dds-submit: Connection established.
dds-submit: Requesting server to process job submission...
dds-submit: Server reports: Creating new worker package...
dds-submit: Server reports: RMS plug-in: /Users/anar/DDS/1.1.52.gfb2d346/plugins/dds-submit-
localhost/dds-submit-localhost
dds-submit: Server reports: Initializing RMS plug-in...
dds-submit: Server reports: RMS plug-in is online. Startup time: 17ms.
dds-submit: Server reports: Plug-in: Will use the local host to deploy 10 agents
dds-submit: Server reports: Plug-in: Using '/var/folders/ng/vl4ktqmx3y93fq9kmtwktpb40000gn/
T/dds_2016-03-31-15-33-32-090' to spawn agents
dds-submit: Server reports: Plug-in: Starting DDSScout in '/var/folders/ng/
vl4ktqmx3y93fq9kmtwktpb40000gn/T/dds_2016-03-31-15-33-32-090/wn'
dds-submit: Server reports: Plug-in: DDS agents have been submitted
dds-submit: Server reports: Plug-in: Checking status of agents...
dds-submit: Server reports: Plug-in: All agents have been started successfully
```

Two ways to activate a topology

```
dds-submit -r RMS -n 100
dds-topology --set <topology_file #1>
dds-topology --activate
dds-topology --stop
dds-topology --set <topology_file #2>
dds-topology --activate
```

(1)

Reserve resources first, then deploy different topologies on it.

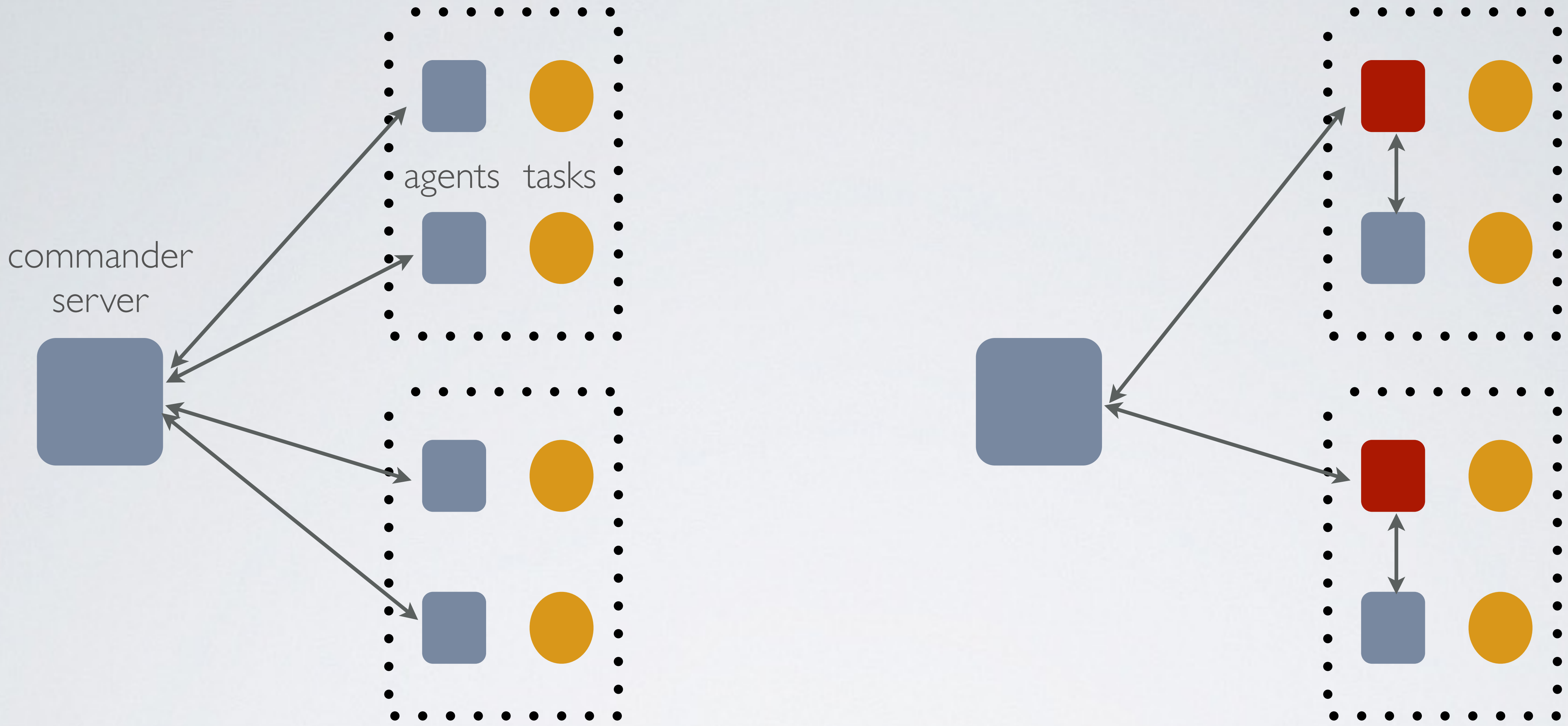
```
dds-submit -r RMS --topo <topology_file>
```

(2)

Reserve resources according to requirements of a given topology.

We aim to delegate the complex work of requirements analysis and corresponding resource allocation to RMS.

Lobby based deployment



1. DDS Commander will have one connection per host (lobby),
2. lobby host agents (master agents) will act as dummy proxy services, no special logic will be put on them except key-value propagation inside collections,
3. key-value will be either global or local for a collection

- Releases - DDS v1.2 (in beta)

  (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