DDS
Dynamic Deployment System

Anar Manafov, Andrey Lebedev
GSI Darmstadt
2014-11-20

Current status

- First stable release - DDS v0.4 (2014-10-24, http://dds.gsi.de/download.html),
- DDS got a 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

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,
- 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

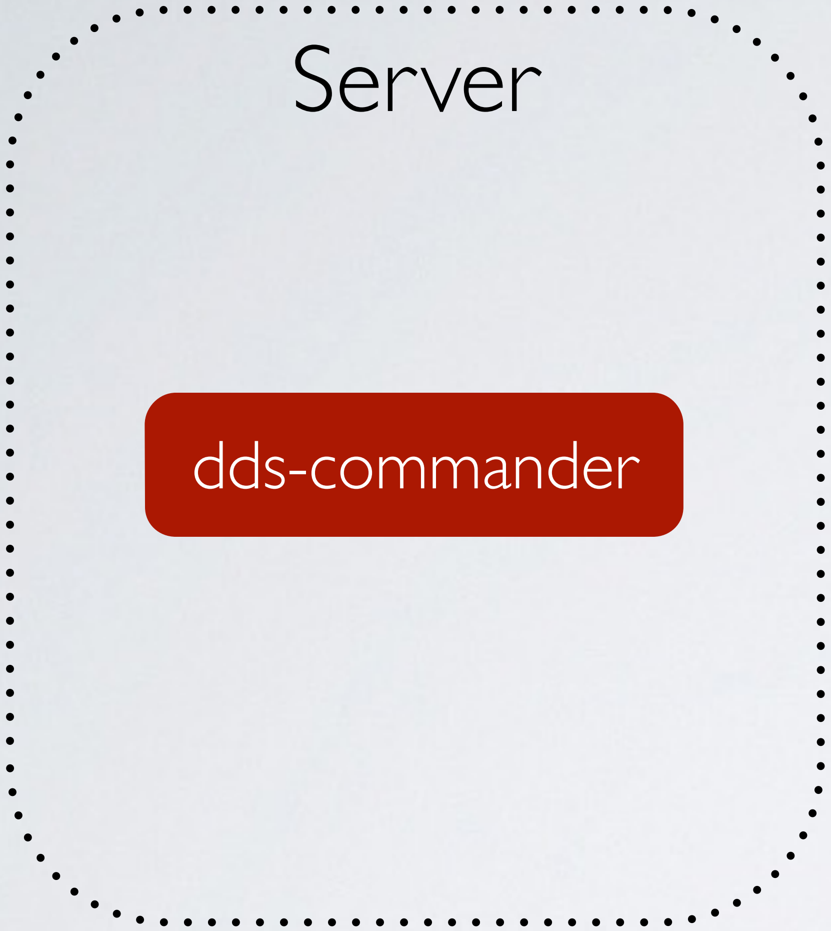
The system takes so called "topology file" (topo) as the input.

- 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.

User Workflow as of DDS v0.4

- Fire up DDS commander server,
- define a desired topology,
- choose RMS and deploy DDS agents,
- activate topology,
- stop DDS commander server.

Start DDS commander server



dds-server *start*

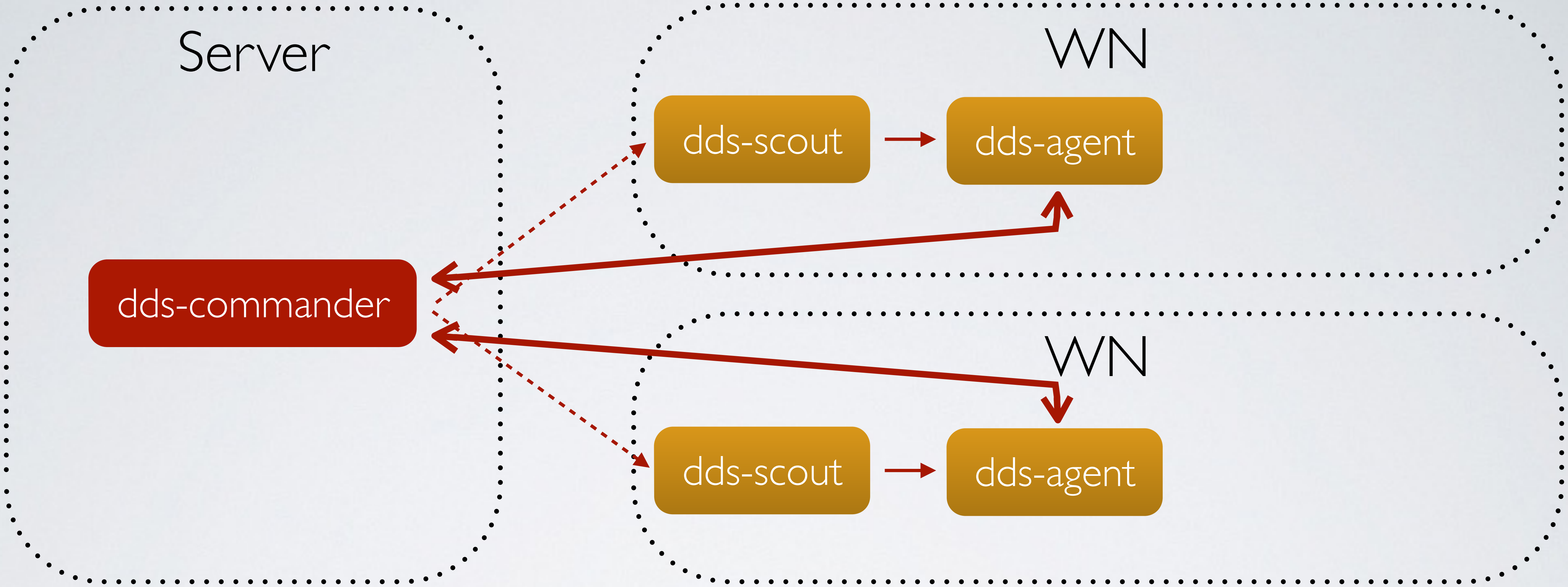
define topology

```
<topology id="myTopology">

   <decltask id="task1">
       <exe reachable="false">/Users/anar/Test1.sh -l --test</exe>
  </decltask>

  <decltask id="task2">
       <exe>/Users/anar/DDS/Test2.sh</exe>
  </decltask>

  <main id="main">
       <task>task1</task>
       <task>task2</task>
  </main>

</topology>
```

defines
weather exe is available on WNs.
If not, DDS will automatically upload
it to related WNs.

DDS learned to parse
commands with command line
arguments including quotes

Choose RMS and Deploy DDS agents



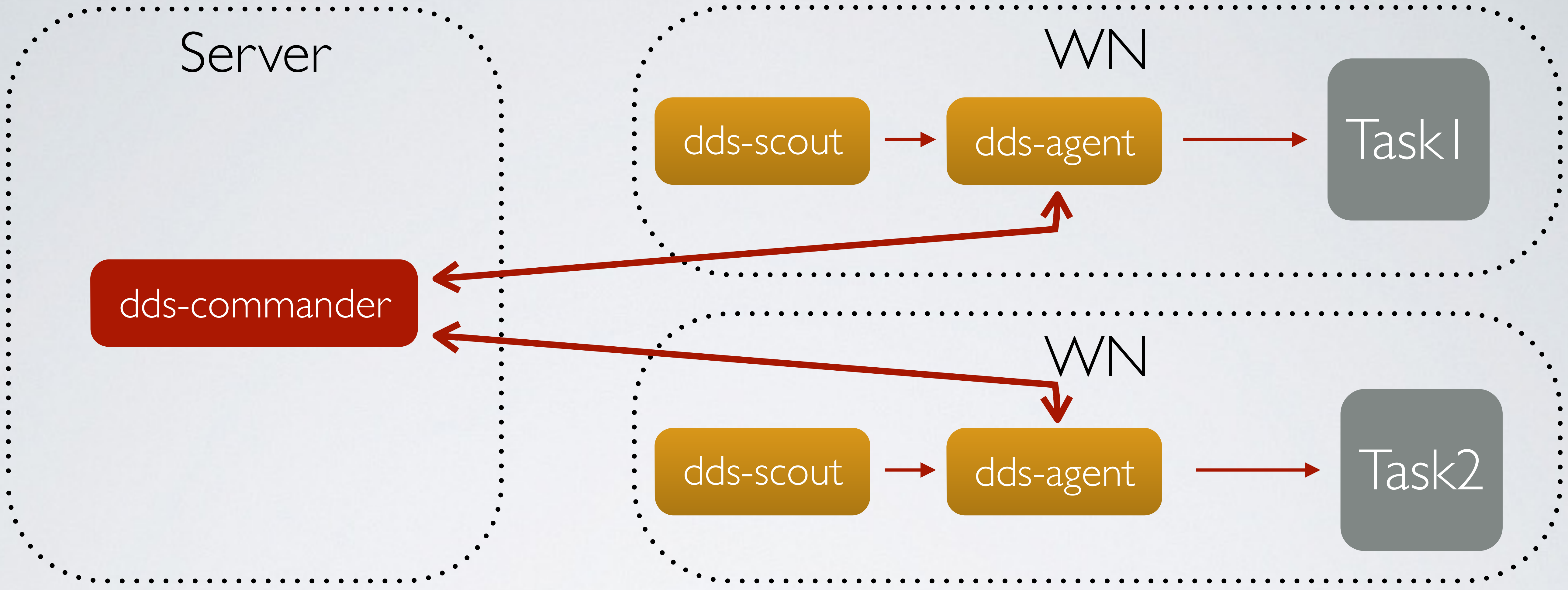
dds-server *start*

dds-submit -t *topology_test.xml* -r *ssh* --ssh-rms-cfg *ssh_hosts.cfg*

DDS's SSH plug-in configuration

```
@bash_begin@
echo "DBG: SSH ENV Script"
@bash_end@

wn1, anar@demac012.gsi.de, -p 24, /Users/anar/dds_wn_test, 1
wn2, anar@demac013.gsi.de, , /Users/anar/dds_wn_test, 4
# wn3, anar@demac013.gsi.de, , /Users/anar/dds_wn_test, 1
```

topology activation



dds-server *start*

dds-submit -t *topology_test.xml* -r *ssh* --ssh-rms-cfg *ssh_hosts.cfg*

dds-topology --*activate*

Stop DDS

Either

> **dds-server** *stop*

or

leave DDS alone and it will automatically stop if idle for a defined amount of time (configurable).

Currently available DDS commands

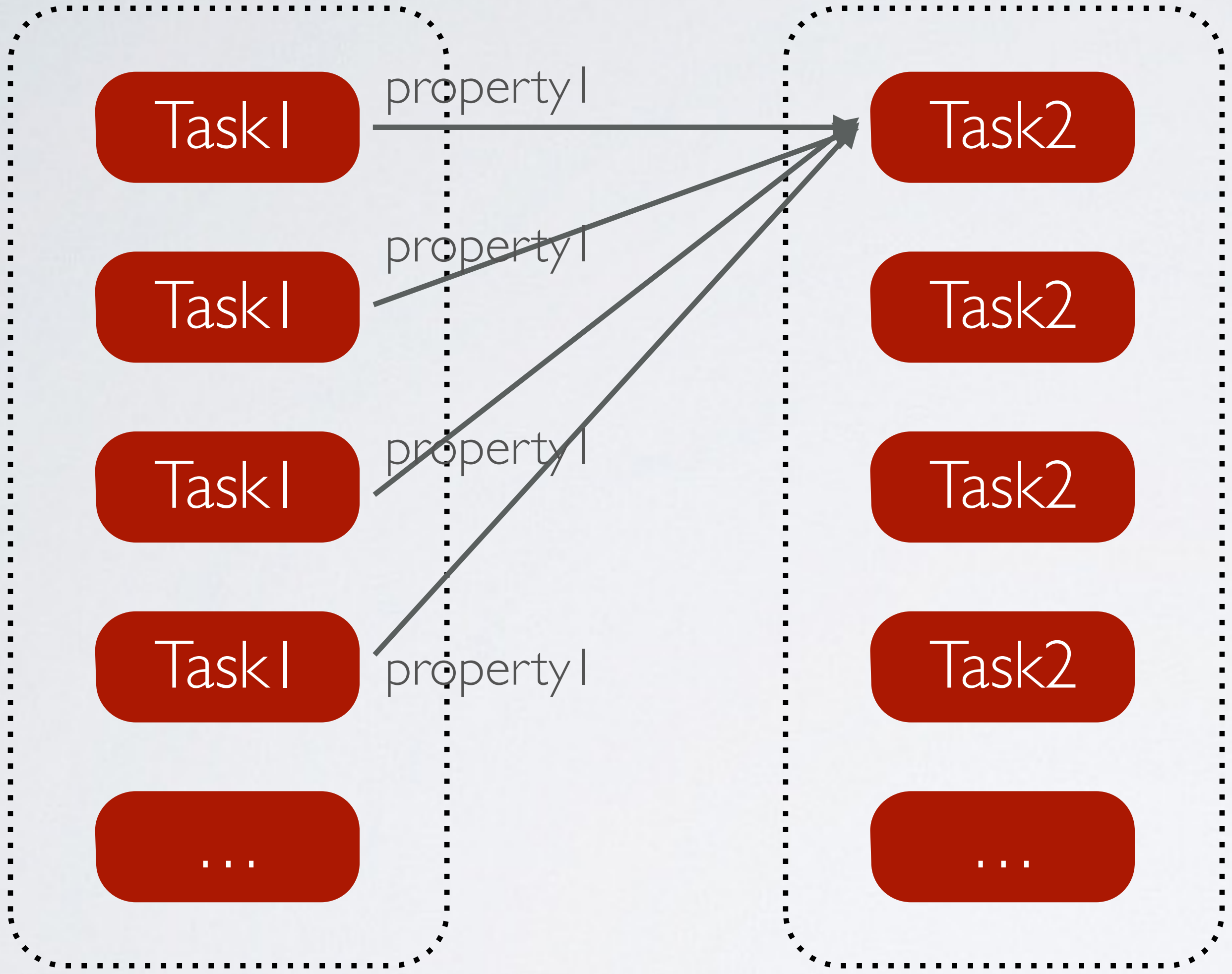
DDS command	Description
dds-commander	a main director service
dds-server	a wrapper around dds-commander
dds-daemonize	a helper program
dds-agent	DDS agent - acts as watch dog and local director on WNs
dds-prep-worker	a helper program to pack DDS WN packages
dds-info	the command to request different kinds of information from/about DDS
dds-topology	the command to validate, activate topologies as well many other topo related operations
dds-user-defaults	it can be used to access DDS settings
dds-submit	the command to submit DDS agent to RMS
dds-ssh	a DDS's SSH custom RMS. It can be used to deploy DDS agents to any machine users have an SSH access to
dds-agent-cmd	the command to send different commands to all or a set of DDS agents
dds-test	it can be used to process different DDS internal tests

Currently available DDS libraries

DDS lib	Description
libdds-topology-lib	Provides topology operations. (internal API)
libdds-protocol-lib	Implements and defines DDS internal protocol. (internal API)
libdds-user-defaults	Implements core user-defaults operations and containers. (open API)
libdds-key-value-lib	Implements key-value propagation API. (open API)
libdds_sys_files	A helper and wrapper API for different system functions. (internal API)
libpipe_log_engine	Implements a pipe based log engine. Used in DDS scripts. (internal API)

Current target

1. Deploy FLP-EPN-like setups



Missing features, currently under development:

- key-value propagation (~80% done)
- task requirements (~50% done)

Current target

2. Test DDS on the scale of about 2-5K agents
   - understand reliability and scalability of the system,
   - detect and solve possible bottlenecks.

Plans

1. Finish up key-value propagation algorithms,
2. finish up the support for tasks requirements,
3. deploy FLP-EPN-like setups on the HLT cluster,
4. improve internal tests capabilities in order to predict and solve possible scalability bottlenecks,
5. constantly adjust DDS development to follow up extensions of the HLT setup prototype.