

## Tango release 7.1.1 (Nov 2009)

- Mainly a bug fixes release after major release 7
- Unix domain socket for communication between devices on the same host (Unix like host)
- Events multi TANGO\_HOST
  - Attribute name given to the callback method is the FQDN

## Tango box (Dec 2009)


- Virtual computer (VMWare) running Ubuntu 9.10
- Tango control system installed
  - Release 7.1.1
  - PyTango 7.1.0 rc
  - Archiving system
    - HDB / TDB / Snapshot and their GUI's
  - Jddd
  - QTango 3 (QtDesigner)
  - ATK 4.0.6 (Netbeans)
  - Alarm system

## Pogo (Not ready yet)

- Re-done using Xtext / Xpand (oaw) technologies
  - Do not need Eclipse to run
- A .xmi file to store model
- Support simple Tango class inheritance
- GUI modified as well
- In testing and finalization phase at the ESRF (for Cpp)

TANGO Code Generator - Release 7.0.0\_a - Thu Apr 08 14:34:03 CEST 201

File Edit Help

Palette: 

Holec ACps PowerSupply

**Holec**

- Class Properties
  - Name
- Device Properties
  - MaxCurrent
  - MinCurrent
- Commands
  - State
  - Status
  - On
  - Off
  - Remote
  - Reset
- Scalar Attributes
  - Current
  - Voltage
- Spectrum Attributes
- Image Attributes
- States
  - ON
  - OFF
  - FAULT

**Tango DeviceImpl**

- + State
- + Status
- + ---

↑

**PowerSupply**

- + State
- + Status
- + ---

↑

**ACps**

- + State
- + Status
- + ---

↑

**Holec**

- + State
- + Status
- + ---

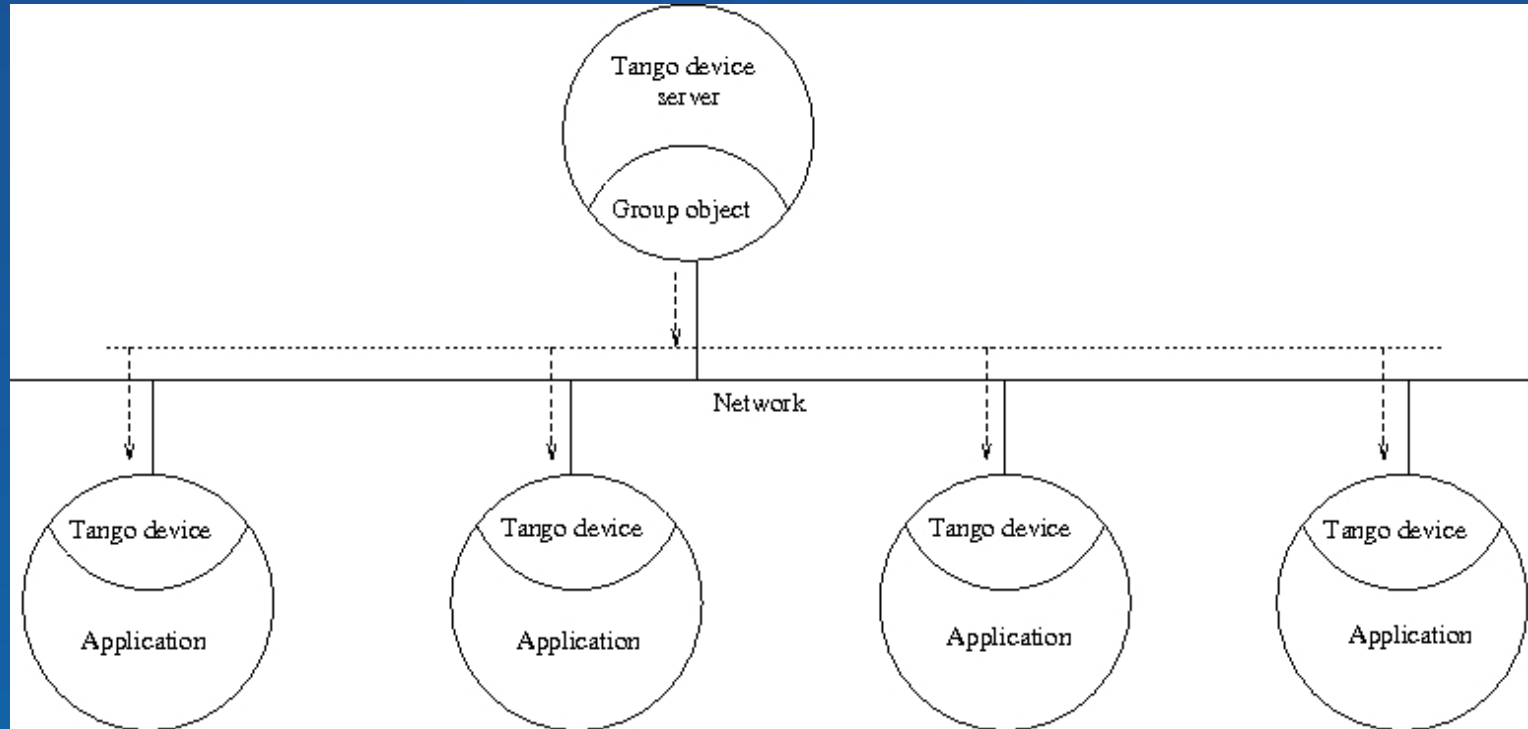
## Next releases

- Tango 7.2 (Mid/End Summer 2010)
  - DeviceProxy class thread safe
  - Added Attribute class methods to inform a client that a device is able to send data ready event
  - Subscribe several times to the same event
  - Several minor changes
  - Bug fixes
- Tango 7.3
  - Forwarded attributes
  - ...
  - Bug fixes

# Event System Replacement Goals

- Improve performances
- If possible
  - no additional process
  - use reliable multicasting
- 3 possible systems
  - Tango group (100 % Tango based solution)
  - DDS
  - Zero MQ

# Event system using Tango group



## Event system using Tango group

- A Tango class inheriting from Device\_Impl and has its own network calls
  - 5 network calls
- One transient device in the database for each application listening on events
- New EventDeviceProxy and EventGroup classes
- A garbage collection thread in the db server to delete transient device entry for crashed applications
- No additional process
  - Only Tango clients and servers

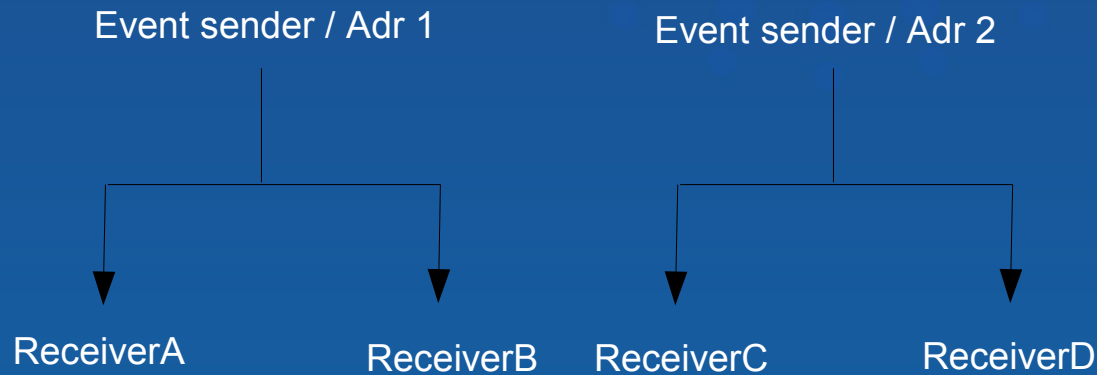
# Event system using Tango group

- Pub: Intel P4 / 2.4 Ghz / 1.5 Gbyte / Ubuntu 9.04
- Clnt: Intel Core2 Duo / 2.6 Ghz / 4 GB / Ubuntu 9.04
- Network: 100 Mbit/sec

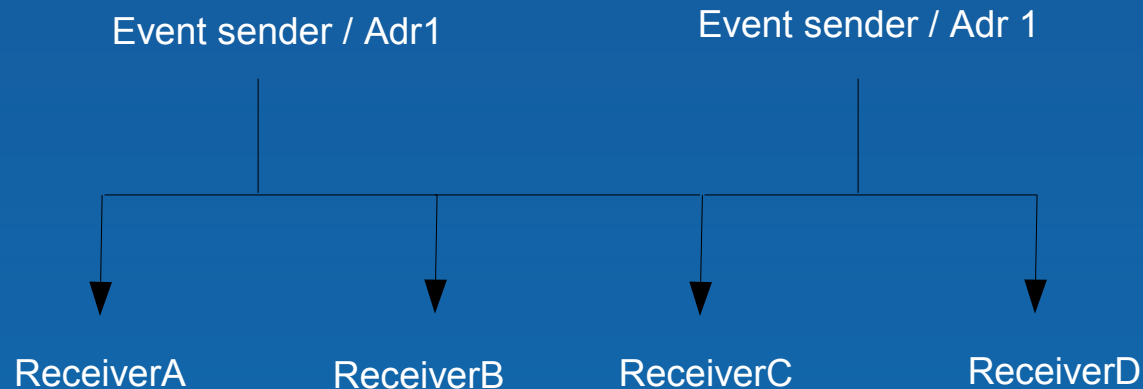
Clnt	1 DevLong		1024 DevLong		262144 DevLong	
	T. V7	Group	T. V7	Group	T. V7	Group
1	770	<b>1330</b>	650	<b>800</b>	10.7	<b>8.5</b>
2	770	<b>1320</b>	460	<b>570</b>	5	<b>4.3</b>
5	400	<b>870</b>	200	<b>280</b>	2.3	<b>1.7</b>
10	220	<b>450</b>	100	<b>150</b>	0.9	<b>0.8</b>

# Multicasting

- Multicast groups without “host pollution”



- Ideal case



- In reality

# Multicasting

- Ideally one multicast group per event !
- How to best spread the events on the available multicast group?
  - Event sender (Tango device server) moving from one host to another
  - Event available on every control room workstation
- Is it possible to have random port number when using multicasting?
  - How is it possible to be sure that a random port is free on all hosts member of the multicast group?
  - Even more difficult problem for late joiner host!

# OpenSplice – Good points

- CORBA ORB / DDS cohabitation
  - We want to be able to share data between the ORB and DDS
    - OpenSplice IDL compiler do not code a IDL sequence the same way than omniORB does
  - Follow the recipe given in the OpenSplice documentation
    - Not very simple but it works.
- DDS Liveliness QoS
  - In the so-called “Automatic mode”, could replace the Tango heartbeat system

# OpenSplice – Good points

- Performance

- Pub: P4 / 2.4 Ghz / 1.5 Gbyte / Ubuntu 9.04
- Sub: Core 2 Duo / 2.6 Ghz / 4 Gbytes / Ubuntu 9.04 (64 bits)
- 100 Mbit network

Sub	1 int (32 bits)			1024 int			1 MByte		
	Tango	Group	<b>DDS</b>	Tango	Group	<b>DDS</b>	Tango	Group	<b>DDS</b>
1	770	1330	<b>12500</b>	650	800	<b>1850</b>	10.7	8.5	<b>7.9</b>
2	770	1320	<b>10500</b>	460	570	<b>1800</b>	5	4.3	<b>7.9</b>
5	400	870	<b>7900</b>	200	280	<b>1800</b>	2.3	1.7	<b>8</b>
10	220	450	<b>6500</b>	100	150	<b>1700</b>	0.9	0.9	<b>8.1</b>

## OpenSplice – Good points

- Small example in Java works fine
- Host on several sub-net
  - Simple test works fine once the network team has allowed some multicast address routing
- The tuner tool (not included in the community edition)
  - But only able to deal with DDS entities running on the local host
  - Need another PrismTech service if you want to see DDS entities running on remote host (not included in the compact edition!)
- SIMD but it's not the standard

# OpenSplice – Annoying points

- XML configuration file
  - One file for all hosts in the control system
  - Make this file available for all control system hosts
- Network failure
  - Long delay (1 min) before the subscriber is informed
    - Need further investigation
  - No error reported to the Data Writer write() method while the network is down.
- DDSi is there but it's not their flagship protocol
- Compatibility between releases

## OpenSplice – Bad points

- Error management (DDS - OMG)
  - No detailed info for calls returning pointers!!
- The `xx_TYPE_NATIVE` in the DDS PSM IDL (DDS - OMG)
  - Compatibility problem
- User process core dump
  - All OS signals with OpenSplice handler to clean-up the DDS entities
  - No info for process debugging
- “kill -9” forbidden
  - *If an application is killed in this manner, the shared memory regions of OpenSplice will become inconsistent and no recovery will then be possible other than re-starting OpenSplice and all applications on the node*

## OpenSplice – Bad points

- Data partitioning
  - One XML file with all partitions defined → Not possible to dynamically connect a host to a partition. Connection at OpenSplice startup
    - Each CTRM workstation will receive all the events !!
  - One algorithm for data partitioning?
- Heavy system
  - 3 processes – Shared memory segment on each host
- Not pure Java – It's a JNI layer
- Configuration
  - Which shared memory segment size?
  - Communication buffer size ?

## ZMQ studies

- <http://www.zeromq.org> and <http://www.imatix.com>
- Comes from the AMQP world
  - Actually completely decoupled from AMQP
- Communication layer above TCP/IP stack
  - Provides a socket like interface
    - You create a ZMQ socket, you bind it, you write to the socket....
  - Provides several messaging patterns
    - Request – Reply
    - Publisher – Subscriber
      - Support a multicasting protocol (OpenPGM / google code)
  - The transported data is a message
    - Do not provide coding / de-coding support
- LGPL

## ZMQ – Testing (Release 2.0.6)

- We have used the CORBA marshaling / un-marshaling system to encode / decode the data in the ZMQ message
- Publisher / Subscribers using classical TCP communication
  - No multicasting
- Performances
  - Same hosts / network than DDS test

Sub	1 int (32 bits)				1024 int				1 MByte			
	T V7	Grp	DDS	ZMQ	T V7	Grp	DDS	ZMQ	T V7	Grp	DDS	ZMQ
1	770	1330	12500	<b>45000</b>	650	800	1850	<b>2400</b>	10.7	8.5	7.9	<b>10.7</b>
2	770	1320	10500	<b>27000</b>	460	570	1800	<b>1200</b>	5	4.3	7.9	<b>4.7</b>
5	400	870	7900	<b>14000</b>	200	280	1800	<b>500</b>	2.3	1.7	8	<b>1.8</b>
10	220	450	6500	<b>7300</b>	100	150	1700	<b>230</b>	0.9	0.8	8.1	<b>0.8</b>

## ZMQ – Good points

- Light weight system
  - No extra processes, only Tango clients and servers
  - Very simple wire protocol
- Performance
  - Previous slide
- Easy to learn (socket like API)
- Moving from TCP pub/sub to Multicasting pub/sub is only changing a string
  - Could be a control system configuration parameter

## ZMQ – Bad points

- Multicasting does not work
  - Some re-send packets never sent
- Java portability
  - The available Java binding is a JNI layer above the C library
- Do not help in amount of code to be written / supported
- Young product

# Summary

	Advantages	Drawbacks
Group	Fully Tango based Only client/server No dependency	Performances Code to be written
DDS	Lot of features Performances	Kill -9 / Core files Heavy Partitioning Java portability Configuration Compatibility between releases
ZMQ	Performances Only clients/servers	Multicasting Code to be written Java portability Young product

## Conclusion

- Let's start with ZMQ?
  - Using CORBA CDR to encode / decode data
  - Keep an eye on ZMQ/PGM and collaborate with them to get it running
    - Should not be a big deal
  - If everything goes smoothly
    - Implement a ZMQ transport for GIOP in omniORB / JacORB to replace IIOp
- Java portability is an issue (platform dependency) !!