KATCP & MeerKAT Overview

SKA LMC Common Framework workshop
Trieste, 25-27 Mar 2015

Lize van den Heever

Technical Lead: MeerKAT Control-And-Monitoring

KATCP

- KATCP = Karoo-Array-Telescope Communications Protocol
- Developed by SKA South Africa in the CASPER collaboration that developed the Reconfigurable Open Architecture Computing Hardware (ROACH) board.boards
- Adopted by a few other projects using ROACH boards
- KATCP Protocol libraries for C, Python, Ruby (no Java)
- TCP/IP based communications
- Text-based, human-readable protocol; encoding of binary data is possible
- KATCP specifies a standardised protocol as well as behaviour (timeouts, replies, logging)
- Synchronous (blocking) and asynchronous supported

- Control through KATCP request-reply pairs, sometimes request-informs-reply
- Supplemented by asynchronous informs for events
- Monitoring and reporting through KATCP sensors
- Logging and log-levels through KATCP informs

- Some standard messages and sensors defined in KATCP protocol specification
- Each project can defines appropriate common requests, informs and sensors as required (e.g. alarms/events for SKA project)

- KATCP carries M&C data only, science data (bulk data) uses SPEAD

KATCP Examples

- Standard requests: `client-list, sensor-list, version-list, help, halt, restart, watchdog, log-level, sensor-sampling, sensor-value`
- Standard informs: `sensor-status, log, client-connected, disconnect, version-connect`
- Standard MeerKAT sensors: `device-status, heartbeat, sensors.ok, comms.ok`
- Sensors: updates comprise: `timestamp, sensor status, sensor value`
- Standard requests, informs and sensors, specified in KATCP protocol, is built into KATCP protocol library
- KATCP request/reply for control commands:

```
?set-x-limit 100
!set-x-limit ok

?log-level debug
!log-level ok debug

?sensor-value x-limit
#sensor-value 1427043978.954988 1 x-limit nominal 100
!sensor-value ok 1
```

- Asynchronous informs:

```
#log info 1427043900.4532 rscadmin.libkatcp rxl:rx\_detected\_and\_on-line
#mode-changed STOP

#sensor-status 1427043968.954988 1 wind-speed nominal 6.563
#sensor-status 1427043978.394981 1 wind-speed nominal 6.140
#sensor-status 1427043988.941945 1 wind-speed nominal 5.966
```

KATCP Examples cont.

Sensors for reporting and monitoring:

- Sensor update comprise: `timestamp, sensor status, sensor value`
- Sensor type :
  `integer, boolean, float, discrete, string, timestamp, address`
- Sensor status:
  `unknown, nominal, warn, error, failure, unreachable, inactive`
- Sampling strategies :
  `auto, none, period, event, differential, event-rate, differential-rate`
- Sensor examples

```
?sensor-list /wind/
#sensor-list wind-direction Wind\_direction\_angle deg float 0 360
#sensor-list wind-speed Wind\_speed m/s float 0 70
!sensor-list ok 2

?sensor-value wind-speed
#sensor-value 1427043978.954988 1 wind-speed nominal 12.5
!sensor-value ok 1

?sensor-sampling wind-speed period 1
#sensor-status 1427101978.170049 1 wind-speed nominal 9
!sensor-sampling ok wind-speed period 1
#sensor-status 1427101988.701059 1 wind-speed nominal 9.1
#sensor-status 1427101998.134524 1 wind-speed nominal 9.65
...

?sensor-sampling wind-speed none
!sensor-sampling ok wind-speed none
```

KATCP Examples cont.

Supports introspection - `?help` for requests/commands and `?sensor-list` for sensors

- **$telnet 10.8.67.53 4005**

```
Connected to 10.8.67.53.
#version-connect katcp-protocol 5.0-IM
#version-connect katcp-library katcp-python-0.6.0a0
#version-connect katcp-device katsim-weather-test-0.1 katsim-weather-test-0.1
```

- **?sensor-list** (or **?sensor-list /wind/** for filtering)

```
#sensor-list device-status Overall\_status\_of\_the\_weather\_system \@ discrete ok degraded fai
#sensor-list fmeca-FD0001-weather-acromag-ok Connection\_to\_weather\_acromag\_ok \@ boolean
#sensor-list input-comms-ok Input\_module\_comms\_ok \@ boolean
#sensor-list pressure Barometic\_pressure mbar float 600 1100
#sensor-list rainfall Rainfall mm float 0 500
#sensor-list relative-humidity Air\_humidity percent float 0 100
#sensor-list temperature Air\_temperature degC float -10 50
#sensor-list wind-direction Wind\_direction\_angle deg float 0 360
#sensor-list wind-speed Wind\_speed m/s float 0 70
!sensor-list ok 9
```

- **?help** (or **?help simulate-value** for filtering)

```
#help simulate-value Set the parameters for simulating the value of a sensor, and immediately
   set the sensor to the average value.
   The value wobbles randomly in the range (value - value_bound, value + value_bound).
   Each time the value changes it moves no more than fluc_bound from the current value.
   Parameters
   ----------
   sensorname : string
        Name of the sensor to set the simulated value of.
   avg_value : float
        Average of the simulated value.
   fluc_bound : float
        Maximum distance value may move from its current value at each simulation step.
   value_bound : float
        Maximum distance value may be from the average value.  ...
   Examples: ?simulate-value air.pressure 900.0 1.0 50.0
```

KAT Comms Libraries

In MeerKAT the KATCP protocol is supported by KAT Comms Python libraries (katcorelib and katuilib):

- Provide the interface and communication infrastructure necessary for binding components of the system together, based on the configuration
- Components in system defined in templated configuration, supports any combination of real and simulated devices
- Extend KATCP interrogation, support dynamic discovery and manage connections, including auto-recovery (re-syncing introspected sensors and commands on re-connection).
- It manages connectivity, reconnection, introspection – and presents a user-friendly container with comms status, requests and sensors - auto-completion based on introspected sensor and request descriptions
- Provide a container object, as per the configuration, containing each component, with its requests and sensors as introspected on the line
- Provide usability layer, like filtering, formatting, colour coding, utility functions
- Provide aggregated control over groups of devices e.g. groups of antennas commanded together
- Provide immediate low-level/manual control of any M&C interface based on introspection of requests
- Provide detailed monitoring of any M&C interface based on introspection of sensors
- All CAM components uses the katcorelib comms library for internal communication, monitoring and control

KAT Comms Libraries examples

KAT container object built from configuration and introspection of KATCP interfaces

```
import katuilib; configure(); kat.status()
Client                    IP            Port          Status       # Commands   # Sensors     Last Connected
=============== =============== ====== =============== ============ ============ ====================
anc             10.8.67.51    4001    synced              11           620          Mon Mar 23 09:06:23 mkat-ancillary-proxy
data_1          10.8.67.51    4003    synced              11           37           Mon Mar 23 09:06:24 mkat-data-proxy
data_2          10.8.67.51    4004    synced              11           37           Mon Mar 23 09:06:24 mkat-data-proxy
data_3          10.8.67.51    4005    synced              11           37           Mon Mar 23 09:06:23 mkat-data-proxy
data_4          10.8.67.51    4006    synced              11           37           Mon Mar 23 09:06:24 mkat-data-proxy
exe             10.8.67.52    2050    synced              8            6            Mon Mar 23 09:06:24 kat-executor
kataware        10.8.67.50    2110    synced              12           81           Mon Mar 23 09:06:24 kat-aware
katlogserver    10.8.67.50    8110    synced              11           6            Mon Mar 23 09:06:24 kat-log-server
katpool         10.8.67.50    2026    synced              13           25           Mon Mar 23 09:06:23 kat-pool
katstore        10.8.67.50    2090    synced              13           6            Mon Mar 23 09:06:24 kat-store
katstore_ar     10.8.67.54    2094    synced              8            7            Mon Mar 23 09:06:2  kat-store-archive
katstore_query   10.8.67.54             2095    synced              10           6              Mon Mar 23 09:06:24 kat-store-query
m000            10.8.67.51    4007    synced              11           606          Mon Mar 23 09:06:23  mkat-receptor-proxy
m001            10.8.67.51    4008    synced              11           606          Mon Mar 23 09:06:23  mkat-receptor-proxy
m062            10.8.67.51    4009    synced              11           606          Mon Mar 23 09:06:23  mkat-receptor-proxy
m063            10.8.67.51    4010    disconnected        11           606          Mon Mar 23 09:06:23  mkat-receptor-proxy
mon_obs         10.8.67.52    2080    synced              12           18           Mon Mar 23 09:06:23  kat-monitor
mon_proxy       10.8.67.51    2080    synced              12           138          Mon Mar 23 09:06:24  kat-monitor
mon_store       10.8.67.54    2080    synced              12           21           Mon Mar 23 09:06:24  kat-monitor
nm_flap         10.8.67.55    2000    synced              10           28           Mon Mar 23 09:06:24  node-manager-2.0
nm_monctl       10.8.67.50    2000    synced              10           172          Mon Mar 23 09:06:24  node-manager-2.0
nm_obs          10.8.67.52    2000    synced              10           36           Mon Mar 23 09:06:24  node-manager-2.0
nm_portal       10.8.67.56    2000    synced              10           44           Mon Mar 23 09:06:23  node-manager-2.0
nm_proxy        10.8.67.51    2000    synced              10           140          Mon Mar 23 09:06:24  node-manager-2.0
nm_sim          10.8.67.53    2000    synced              10           188          Mon Mar 23 09:06:23  node-manager-2.0
nm_store        10.8.67.54    2000    synced              10           52           Mon Mar 23 09:06:24  node-manager-2.0
sched           10.8.67.50    2060    synced              8            30           Mon Mar 23 09:06:24  katscheduler-1.0
```

Blue indicates a controlled component, **Green** a monitor-only component

KAT Comms Libraries cont.

- Detailed monitoring of all components in the container based on the sensor introspection

```
In [10]: kat.m001.sensors.ap_<tab>
  Display all 152 possibilities? (y or n)
 kat.m000.sensor.ap_actual_azim            kat.m000.sensor.ap_actual_azim_rate      kat.m000.sensor.ap_actual_elev
 kat.m000.sensor.ap_actual_elev_rate            kat.m000.sensor.ap_requested_azim              kat.m000.sensor.ap_requ
 kat.m000.sensor.ap_build_state            kat.m000.sensor.ap_local_time_synced     kat.m000.sensor.ap_tiltmeter_read_e
 kat.m000.sensor.ap_control                     kat.m000.sensor.ap_mode                  kat.m000.sensor.ap_on_source_t
 kat.m000.sensor.ap_on_target              kat.m000.sensor.ap_ped_door_open         ...

In [5]: kat.m063.print_sensors("mode|control", "period", 1.0)
 Print filtered sensors on m063 : mode|control : once :   @  09:07:01  |
 Page 1 of 1  <B>ack  <N>ext    Items:10   Per page:14 (+/-)            Q to quit
 ------------------------------ Once off - values not updating ----------------------------------
 Name                                        Unit         Status      Value time              Update time

 ap.azim-aux1-mode-selected                                nominal   19:06:18.73   09:07:00.95             0
 ap.azim-aux2-mode-selected                                nominal   19:06:18.73   09:07:00.95             0
 ap.cb-sdc-recv-controller-closed                          nominal   19:06:18.73   09:07:00.96             1
 ap.control                                               nominal   19:06:18.73    09:07:00.96            remote
 ap.mode                                                  nominal   22:04:45.37    09:07:00.96            stop
 mode                                                     nominal   22:04:45.39    09:07:00.97            STOP
 rsc.rxl.rfe2.temp-control.enabled             boolean     nominal   06:55:15.41   09:07:00.97             0
 rsc.rxs.rfe2.temp-control.enabled             boolean     nominal   19:06:28.74   09:07:00.97             0
 rsc.rxu.rfe2.temp-control.enabled             boolean     nominal   19:06:29.50   09:07:00.98             0
 rsc.rxx.rfe2.temp-control.enabled             boolean     nominal   19:06:27.78   09:07:00.98             0

In [11]: kat.kataware.print_sensors("Wind")
 Print filtered sensors on kataware : Wind : once :   @  09:08:33  |
  Page 1 of 1  <B>ack  <N>ext    Items:3   Per page:3 (+/-) Q to quit
  ------------------------------ Once off - values not updating ----------------------------------
  Name                              Unit        Status      Value time              Update time             Value

  alarm.ANC_Wind_Gust                         nominal   21:56:44.10        09:08:33.28             nominal,new,anc_gust_wi
value = '1...
  alarm.ANC_Wind_Reporting_Failure             nominal   19:08:04.10   09:08:33.28
           nominal,cleared,agg anc wind reporting ok ...
```

KAT Comms Libraries cont.

- Aggregated control over groups of devices e.g.

```
kat.ants.req.mode("POINT")
equivalent to serially commanding the antennas with
kat.ant1.req.mode("POINT"), kat.ant2.req.mode("POINT"), ...
```

- Low-level / manual control of any M&C interface based on introspection of requests

```
In [09]: import katuilib; configure();
In [10]: kat.m001.req.ap_<tab>
cam.m001.req.ap_clear_track_stack       cam.m001.req.ap_set_on_source_threshold
cam.m001.req.ap_star_track                      cam.m001.req.ap_enable_point_error_refracti
cam.m001.req.ap_maintenance             cam.m001.req.ap_set_stow_time_period
cam.m001.req.ap_stop
          cam.m001.req.ap_enable_point_error_systematic  cam.m001.req.ap_rate
          cam.m001.req.ap_set_weather_data
cam.m001.req.ap_stow
          cam.m001.req.ap_enable_point_error_tiltmeter   cam.m001.req.ap_reset_failures
          cam.m001.req.ap_set_x_band_focus
cam.m001.req.ap_track                                      cam.m001.req.ap_enable_warning_ho
cam.m001.req.ap_set_indexer_position    cam.m001.req.ap_slew
cam.m001.req.ap_track_az_el  …

In [11]: kat.m001.req.ap_track_az_el?
 Type:       KATRequest
 String Form:<katcorelib.katcp_client.KATRequest object at 0x7f53fc36ea50>
 File:       /usr/local/lib/python2.7/dist-packages/katcorelib/katcp_client.py
 Definition: cam.m001.req.ap_track_az_el(self, *args, **kwargs)
 Docstring: Request the AP to set the antenna at the position specified by the
 azimuth and elevation parameters at a specified time.
 Parameters:
 timestamp : KATCP Timestamp, The time when the position coordinates should be applied
 azim : float, Azimuth coordinate (degrees)
 elev : float, Elevation coordinate (degrees)
 Returns:
```

MeerKAT and KATCP

- MeerKAT uses KATCP protocol and KAT comms libraries (katcorelib and katuilib) as the communications middleware, also between components within MeerKAT Control-And-Monitoring (CAM) subsystem
- MeerKAT CAM architecture evolved through XDM, Fringe-Finder, KAT-7 to MeerKAT
- Design decisions were made for SKA-Phase 1, with scalability to SKA-Phase 2 always in mind
- MeerKAT CAM is fully configuration driven, including telescope specification and CAM deployment
- MeerKAT CAM is designed to completely adapt to requests and sensors discovered through introspection. Rolled-up sensors, aggregate sensor rules and alarm rules, GUI status displays automatically include sensors discovered during introspection
- KAT container provides low-level control of all discovered requests
- MeerKAT CAM design is heavily based on sensor monitoring - setting sensor sampling strategies and reacting to the updates, then commanding via KATCP requests to take action

Terminology:

- MeerKAT CAM (Control-And-Monitoring) is the equivalent of SKA TM
- MeerKAT "devices" are equivalent of instances of Element LMCs
- MeerKAT CAM components are equivalent of instances of TM subelements

MeerKAT M&C "Framework"

(All these and more are described in par. *4. CAM concepts and design decisions* of shared MeerKAT CAM Design Description rev 2)

- **KATCP device translators** for all hardware devices/subsystems that are not delivered with a KATCP interface (modbus, OPC/UA, SNMP, web-services, Ganglia metrics)
- **Proxy layer** (Leaf nodes in SKA-speak) for common control and monitoring of all devices
- **Hierarchical and distributed monitoring** - Rolled-up sensors implemented on proxies and nodemanagers, aggregate sensors per rules in configuration, distributed katmonitors to gather and archive monitoring data, central kataware component implements alarms management and actions as per alarms definition in configuration
- **Standardised central logging** - Device logging over KATCP. Proxies and Monitors components, gather and store all KATCP logs centrally. Level of logging over the KATCP interface is configurable via KATCP request. This provides a consistent mechanism and formatting for system-wide logs and a central store of system logs to support fault finding and engineering tests. The CAM subsystem provides a web interface for viewing on-line system logs, which allows the log sources to be filtered and ordered on user request.
- **Homogeneous node (virtualised container) management**, configuration driven deployment
- CAM development with **fully simulated system** - through KATCP simulators, simulating the KATCP interface and device behaviour. CAM system can be functionally exercised and qualified in a fully simulated environment.

MeerKAT Hierarchical & Distributed Monitoring

- Hierarchical monitoring:
  - by defining on rolled-up sensors in proxies and nodemanagers
  - averaging sensors for mean, max, min values across multiple similar sensors (e.g. for mean wind speed, max wind gust), maths based, defined in configuration, implemented by proxies
  - aggregate sensors are rule based across multiple child sensors, defined in configuration
  - aggregate sensors implemented by katmonitors
  - health & status displays, and alarms, mostly based on rule-based aggregate sensors
- Distributed monitoring through katmonitors
  - a katmonitor on each node, gathers and archives all sensors exposed on that node
  - any sensor introspected will automatically be gathered and archived and included in rules to which it matches and in rolled-up sensors
  - new nodes are simply added to the configuration and then are automatically included in the system monitoring and archiving by defining an instance of katmonitor for that node
- Telescope level monitoring and alarms management
  - implemented by central kataware component
  - rule-based alarm definitions (mostly on aggregate sensors) including alarm actions and notifications, in configuration
  - health & status displays, mostly based on rule-based aggregate sensors
- Central logging
  - device logs gathered via KATP interface and exposed by proxies for central logging

MeerKAT Device M&C

MeerKAT proxy layer

- KATCP device translators implemented for converting all protocols to KATCP (modbus, OPC/UA, SNMP)
- MeerKAT proxies (leaf nodes in SKA.TM-speak) provide consistent layer for monitoring and control of all device

  Proxies protect access to lower-level devices/ element LMCs / hardware. All engineering/support/system components/tools connect via the proxy layer and not directly to hardware devices/subsystems.
- Proxies implement rolled-up sensors for all devices it manages, e.g. `sensors.ok`, `comms.ok`, with sensor status (`nominal, warn, error`, etc), and rolled-up `device-status` (`ok, degraded, fail`), indicating the worst status of the aggregation
- Proxies facilitate central logging of all device logs received on KATCP interface

MeerKAT Telescope Aggregate sensors

Aggregate sensor rules examples:

```
{% for id in ants %}
[${id}:agg_${id}_no_wind_stow]
description = True if ${id} is not stowing
${id}_windstow_active = value == False
{% end %}

[sys:agg_sys_nodemanagers_ok]
description = True if all the node managers in the system are happy
{% for n in system_nodes %}sys.monitor.nm_${n} = value == True
{% end %}

[anc:agg_system_cooling_ok]
description = True if the KAPB cooling is OK
anc_bms_imminent_cooling_failure = value == False

[anc:agg_system_intrusion_ok]
description = True if there is no intrusion in the system
anc_bms_kapb_rfi_door_open = value == False
{% for id in ants %}
${id}_ap_ped_door_open = value == False
{% end %}
```

MeerKAT Telescope Monitoring and Alarms

Alarm rules examples:

```
{% for n in system_nodes %}
[${n} Disk Use]
numeric = True
sensor = anc.ganglia.${n}.kat.disk_part_max_used
strategy = period 60
warn = None, 75, 1
error = None, 85, 1
{% end %}

[System Cooling Failure]
numeric = False
sensor = agg_system_cooling_ok
strategy = period 1
delay = 60
critical = value == '0'
action = notify_sys
action_params = mkat-shutdown-kapb

[ANC Wind Speed]
numeric = True
#ANC implements this sensor giving 10-min mean wind speed over all wind sensors
sensor = anc.mean_wind_speed
strategy = period 1
#40 km/h = 11.1 m/s
critical = None, 11.1, 1.5
action = notify_sys
action_params = automated-windstow-antennas
```

MeerKAT CAM Deployment

(The term “CAM node” is used for a virtualised container running any set of CAM processes.)

- Configuration driven deployment
  - specify VMs/containers (called nodes) and their allocation to host servers
  - specify processes to launch per node
  - configuration centrally served from a head node to all nodes as required
  - very easy to scale as the project grows and deploys more elements and/or functionality: adding additional host servers, nodes and/or processes only needs an update of the relevant configuration

- Deploy generic nodes
  - each node has all CAM software deployed and could run any of the CAM components
  - each node initially runs only a single nodemanager service
  - nodemanager waits for instructions from central system controller to register and launch processes to run on that node
  - central system controller coordinates launching of groups of registered processes across nodemanagers, to facilitate sequenced start up and controlled shut down

- Nodemanagers:
  - provides generic process monitoring `(pids, running, mem and cpu usage, etc)`
  - provides generic process management `(register/deregister, restart, start/stop, kill)`
  - launches registered processes per groups as coordinated by system controller
  - manages process output into a common file storage on each node

MeerKAT and KATCP - Pros & Cons

Pros:

- KATCP - protocol itself is light-weight, human-readable
- Extremely simple to use and lean
- Puts very little requirements on devices (TCP/IP stack) and protocol wrappers/drivers
- KATP - uses TCP/IP as the field bus
- KATCP library – provides server and client implementations, configurable multiple connections and configurations, implements connection management and introspection
- KATCP simulator framework exists, easy to build simulators for any KATCP interface
- Simple and lean implementation, nothing more and nothing less than what is required, short learning curve, easy to use, adapt and expand
- katcorelib and katuilib comms libraries provides middle ware and rich usability layer
- MeerKAT CAM design is fully adaptive to introspection
- Modern technologies, implementation and software developement approaches
- MeerKAT architectural decisions were made with SKA phase 1, with potential scalability for SKA phase 2, in mind

Cons:

- Not a framework per se: MeerKAT "Framework" = KATCP protocol + KAT comms libraries + selected generic and central components
- Protocol does not implement security or authentication on the line – relies on underlying network design
- Provides Python, C and Ruby protocol libraries, no JAVA, KAT comms libraries in Python only

MeerKAT and KATCP - Conclusion

(FWP = Framework Protocol)

- It's a different decision to make, not a framework per se,
  but if you wanted to maximise precursor re-use for risk reduction or schedule pressure

To leverage the precursor environment and support early integration, AIV and element LMC interface validation:

- Create a KATCP <-> FWP translator (for general use and plug-and-play e.g. to wrap simulators or existing MeerKAT components like weather stations)
- Adapt Proxy base (leaf nodes) to "talk" FWP
- Adapt the KAT comms libraries to "talk" FWP which will present the user with an SKA container, as it currently presents KAT container

Benefits:

- This immediately provides an engineering interface to Element LMCs
- As well as a potential for a fully simulated SKA system based on Element ICDs
- It allows for any SKA Element to be integrated into MeerKAT system for early integration and testing within the precursor system
- This can be done on a very short timescale (2-3 months)

Useful links

- CASPER - https://casper.berkeley.edu/wiki/KATCP
- KATCP-Python - https://pypi.python.org/pypi/katcp
- MeerKAT CAM (Control-And-Monitoring) Design Documentation Pack, Dated: 12 August 2013 https://drive.google.com/drive/u/0/#folders/0B8fhAW5QnZQWNUVfa0l6SG5SOXVMVXhRS3JsVXEwQQ/0B8fhAW5QnZQWdzdYdmlNTkhValE/0B8fhAW5QnZQWTmI5UWdRdkpoTWs
- KATCP Documents: KATCP Protocol Specification, Guidelines for Communication with Devices Rev 5 and KATCP_Device_Simulator_Requirement_Specification_Rev1. All of these in LMC Shared google drive folder: https://drive.google.com/open?id=0B8fhAW5QnZQWRzV0VjBLNUcwekk&authuser=0

SKA AFRICA
SQUARE KILOMETRE ARRAY

Questions ?