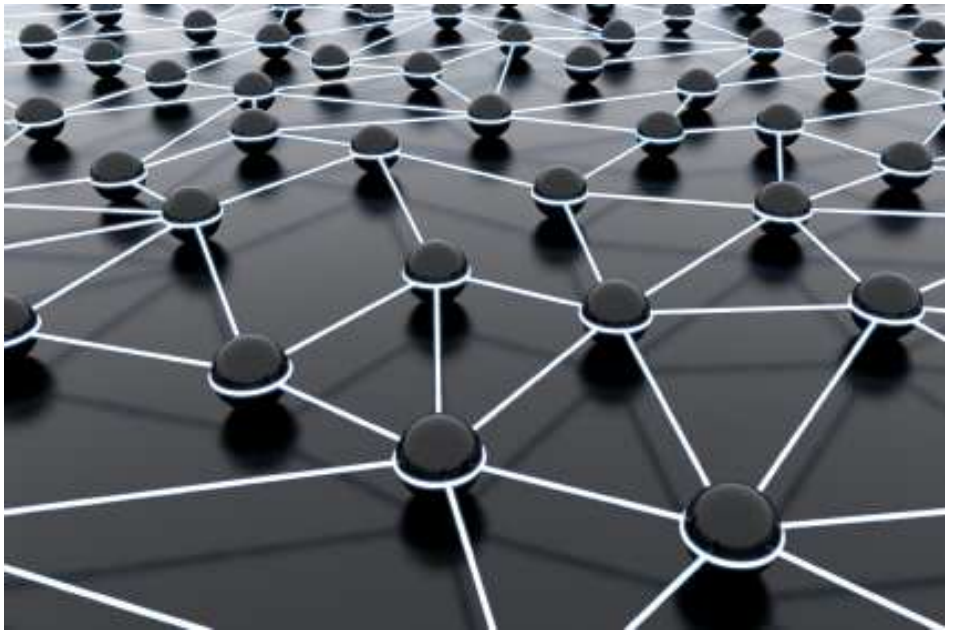
NSM2 for Telescope Test

Mikihiko Nakao (KEK-IPNS)

mikihiko.nakao@kek.jp



April 10, 2013

現場 (Gemba) meeting

at DESY

Goal

- **“Master Control” to control everybody**
  - TTD, COPPER, EVB and HLT (NSM2 native speakers)
  - PXD, SVD (EPICS speakers)
  - EUDET (foreign language)
- **Control means**
  - Set up and start subsystem in the correct sequence
  - Actions to be made by the Master Control is high-level ones only
  - Correct all the log messages with proper time stamp (msec unit) to understand the sequence and origin of a problem
- **“Logging Database”**
  - Events initiated by the Master
  - Log messages from subsystems
- **Beta-testing for the Belle II control system**

NSM2 status

- **NSM2 alpha release** (2013.3)
  - Low-level technology choices were made and implemented (corelib)
  - Very preliminary Belle II dedicated library is made (b2lib)
- **NSM2 beta release plan**
  - Database interface for log messages and NSM shared memory data
  - mysql at the beginning, to be switched to what database group offers
  - Belle II dedicated library for state handling
  - First target system: CDC/Belle2link/COPPER system at KEK
  - Currently lower priority w.r.t. FTSW firmware development

Running EUDET Run Control

- Get the zip file and unzip (I had a problem in getting from SVN)
- `cd main` and `make` (to compile GUI, go to `gui` and make)
- Open four terminals, and `cd bin`
- Make an empty configuration file: `touch ../conf/default.conf`
- Run `./TestRunControl.exe`, `./TestLogCollector.exe`, `./TestDataCollector.exe`, `./TestProducer.exe` (in this order)
- In run control window, type `c` to config, then `b` to begin run
- In producer window, type `r` to generate one event

GUI version (run control and log collector) looks similar

TestRunControl.exe

```
donau:p6
nakao@donau(31)% ./TestRunControl.exe
DEBUG: listenaddress=tcp://localhost:44000
--- Commands ---
p       Print connections
l [msg] Send log message
c [cnf] Configure clients (with configuration 'cnf')
r       Reset
s       Status
b [msg] Begin Run (with run comment 'msg')
e       End Run
x       Terminate clients
q       Quit
?       Display this help
---------------
Connect:    (waiting) (127.0.0.1:52397)
LogServer responded: tcp://localhost:44002
Connect:    LogCollector (127.0.0.1:52397)
Receive:    OK from LogCollector (127.0.0.1:52397)
Connect:    (waiting) (127.0.0.1:52400)
DataServer responded: RunNumber = 2
DataServer responded: Server = 'tcp://localhost:44001'
Connect:    DataCollector (127.0.0.1:52400)
Receive:    OK from LogCollector (127.0.0.1:52397)
Receive:    OK from DataCollector (127.0.0.1:52400)
Receive:    OK from DataCollector (127.0.0.1:52400)
Connect:    (waiting) (127.0.0.1:52402)
Connect:    Producer.Test (127.0.0.1:52402)
Receive:    OK from Producer.Test (127.0.0.1:52402)
Receive:    OK from Producer.Test (127.0.0.1:52402)
c
Receive:    NONE: Wait from LogCollector (127.0.0.1:52397)
Receive:    NONE: Wait from DataCollector (127.0.0.1:52400)
Receive:    NONE: Wait from Producer.Test (127.0.0.1:52402)
Receive:    NONE: Wait from LogCollector (127.0.0.1:52397)
Receive:    OK: Configured () from DataCollector (127.0.0.1:52400)
Receive:    OK: Configured () from Producer.Test (127.0.0.1:52402)
b
Receive:    NONE: Wait from DataCollector (127.0.0.1:52400)
Receive:    NONE: Wait from Producer.Test (127.0.0.1:52402)
Receive:    NONE: Wait from LogCollector (127.0.0.1:52397)
Receive:    OK from DataCollector (127.0.0.1:52400)
Receive:    NONE: Wait from LogCollector (127.0.0.1:52397)
Receive:    OK from Producer.Test (127.0.0.1:52402)
```

TestLogCollector.exe

```
donau:p12
nakao@donau(21)% ./TestLogCollector.exe
###### listenaddress=tcp://localhost:44002
       logfile=../logs/2013-04-10.log
Connect:    RunControl (127.0.0.1:44465)
Connect:    LogCollector (127.0.0.1:44466)
Connect:    DataCollector (127.0.0.1:44468)
Connect:    Producer.Test (127.0.0.1:44470)
INFO: Connection from Producer.Test (127.0.0.1:35857) 2013-04-10 11:16:41.681 Da
taCollector
INFO: Configuring () 2013-04-10 11:16:47.097 RunControl
Config:
Name =

[LogCollector]


INFO: Configured () 2013-04-10 11:16:49.598 Producer.Test
INFO: Starting Run 3:  2013-04-10 11:16:55.798 RunControl
INFO: Preparing for run 3 2013-04-10 11:16:56.799 DataCollector
Start Run: 3
```

TestProducer.exe

```
donau:p16
nakao@donau(7)% ./TestProducer.exe
--- Commands ---
s data Send StringEvent with 'data' as payload
r size Send RawDataEvent with size bytes of random data (default=1k)
l msg  Send log message
o msg  Set status=OK
w msg  Set status=WARN
e msg  Set status=ERROR
q      Quit
?
----------------
Configuring.
Start Run: 3
r
r
r
```

TestDataCollector.exe

```
donau:p13
nakao@donau(10)% ./TestDataCollector.exe
Connect:    Producer.Test (127.0.0.1:35857)
Configuring ()...
...Configured ()
Complete Event: 3.0
Complete Event: 3.1
Complete Event: 3.2
Complete Event: 3.3
```

single host, 4 processes

Controlling EUDET

- **NSM2 to control EUDET**
  - CUI version of EUDET run control is easy to understand and modify
  - EUDET can control NSM2, too, but in my point of view, EUDET is a much simpler program and there's no reason to do so
- **A quick hack**
  - CONFIG/START/STOP from NSM2 to EUDET was coded in 1 hour
  - It is not yet working in a coordinated way, e.g., run numbers or error messages, but no technical problem forseen

NSM2RunControl.exe

```
donau:p6
p       Print connections
l [msg] Send log message
c [cnf] Configure clients (with configuration 'cnf')
r       Reset
s       Status
b [msg] Begin Run (with run comment 'msg')
e       End Run
x       Terminate clients
q       Quit
?       Display this help
----------------
Connect:    (waiting) (127.0.0.1:52837)
LogServer responded: tcp://localhost:44002
Connect:    LogCollector (127.0.0.1:52837)
Receive:    OK from LogCollector (127.0.0.1:52837)
Connect:    (waiting) (127.0.0.1:52840)
Connect:    Producer.Test (127.0.0.1:52840)
Receive:    OK from Producer.Test (127.0.0.1:52840)
Connect:    (waiting) (127.0.0.1:52842)
DataServer responded: RunNumber = 5
DataServer responded: Server = 'tcp://localhost:44001'
Connect:    DataCollector (127.0.0.1:52842)
Receive:    OK from LogCollector (127.0.0.1:52837)
Receive:    OK from DataCollector (127.0.0.1:52842)
Receive:    OK from DataCollector (127.0.0.1:52842)
Receive:    OK from Producer.Test (127.0.0.1:52840)
12:32:47.999 CONFIG<=MASTER
CONFIG message received
Receive:    NONE: Wait from DataCollector (127.0.0.1:52842)
Receive:    NONE: Wait from LogCollector (127.0.0.1:52837)
Receive:    NONE: Wait from Producer.Test (127.0.0.1:52840)
12:32:48.499 OK=>MASTER (IDLE) configured
Receive:    NONE: Wait from LogCollector (127.0.0.1:52837)
Receive:    OK: Configured () from DataCollector (127.0.0.1:52842)
Receive:    OK: Configured () from Producer.Test (127.0.0.1:52840)
12:32:59.234 START<=MASTER
CONFIG message received
Receive:    NONE: Wait from LogCollector (127.0.0.1:52837)
Receive:    NONE: Wait from Producer.Test (127.0.0.1:52840)
Receive:    NONE: Wait from DataCollector (127.0.0.1:52842)
Receive:    OK from DataCollector (127.0.0.1:52842)
12:33:01.237 OK=>MASTER (RUNNING) run started
Receive:    NONE: Wait from LogCollector (127.0.0.1:52837)
Receive:    OK from DataCollector (127.0.0.1:52842)
Receive:    OK from Producer.Test (127.0.0.1:52840)
```

NSM2 master

```
donau:p14
nakao@donau(67)% ./master MASTER
nsmlib_checkif: checking interface <lo>
nsmlib_checkif: interface <lo> does not support broadcast
nsmlib_checkif: checking interface <wlan0>
shmkey=8120 size=616680

MASTER>config EUDET
ac=2 av[0]=config
MASTER>start EUDET 7
ac=3 av[0]=start
MASTER>
```

```
donau:p5
nakao@donau(66)% ./nsminfo2
NSM network started at 2013.04.10-12:25:21 (5m52s ago)
NSMD daemon started at 2013.04.10-12:25:21 (5m52s ago)
MASTER(1) at 172.31.0.116, DEPUTY(-1) is missing, I'm the MASTER nsmd
NSMD shmkey = 8120, ip = 172.31.0.116, pid = 12015
     created 5m52s ago, last updated 0s ago (up-to-date)
CONN 2   nid=  1 pid=12046 sock= 6 rtim=unknow nodeid=1
CONN 3   nid=  0 pid=12048 sock= 5 rtim=unknow nodeid=0
NODE 0   MASTER                          pid/uid=12048/21022 @172.31.0.116 -1
NODE 1   EUDET                           pid/uid=12046/21022 @172.31.0.116 -1
REQ  0   OK                              code=1000 hash=1801(262184,1000)
REQ  1   ERROR                           code=1001 hash=1810(262220,1001)
REQ  2   CONFIG                          code=1002 hash=173(262256,1002)
REQ  3   START                           code=1003 hash=481(262292,1003)
REQ  4   STOP                            code=1004 hash=984(262328,1004)
alist = -1
nakao@donau(67)%
```

nsminfo2

```
donau:p17
12:32:59.234 DBG: tcpwriteq: write = 20/20 9013c00
12:32:59.234 DBG: writeq before f=09013be0(00000000) l=09013be0 q=09013be0
12:32:59.234 DBG: writeq after f=00000000(00000000) l=00000000 q=09013be0
12:33:01.237 DBG: tcprecv recvlen=16/16 recvp=9006610
12:33:01.237 DBG: tcprecv recvlen=28/28 recvp=9006620
12:33:01.237 DBG: tcprecv req=1000 len=20 npar=2 p=[4099,1] coni=0
12:33:01.237 DBG: command: req = 1000 len = 20 npar = 2 datap = RUNNING con = 3
12:33:01.237 DBG: tcpsend f=09014028(00000000) l=09014028 q=09014028 1
12:33:01.237 DBG: tcpwriteq req=1000 len=20 npar=2 p=[4099,1] buf=9014048 writep
=9014048
12:33:01.237 DBG: tcpwriteq: write = 44/44 9014048
12:33:01.237 DBG: writeq before f=09014028(00000000) l=09014028 q=09014028
12:33:01.237 DBG: writeq after f=00000000(00000000) l=00000000 q=09014028
```

nsmd2 daemon

7 processes (4 above +3 EUDET processes)

NSM2EUDET plan

- Proper response and error forwarding back to NSM2
- Log message forwarding to NSM2
- Configuration parameters passed from NSM2
- Rest of the world is kept as it is from NSM2 point of view, but DataCollector will be modified to connect to EB2

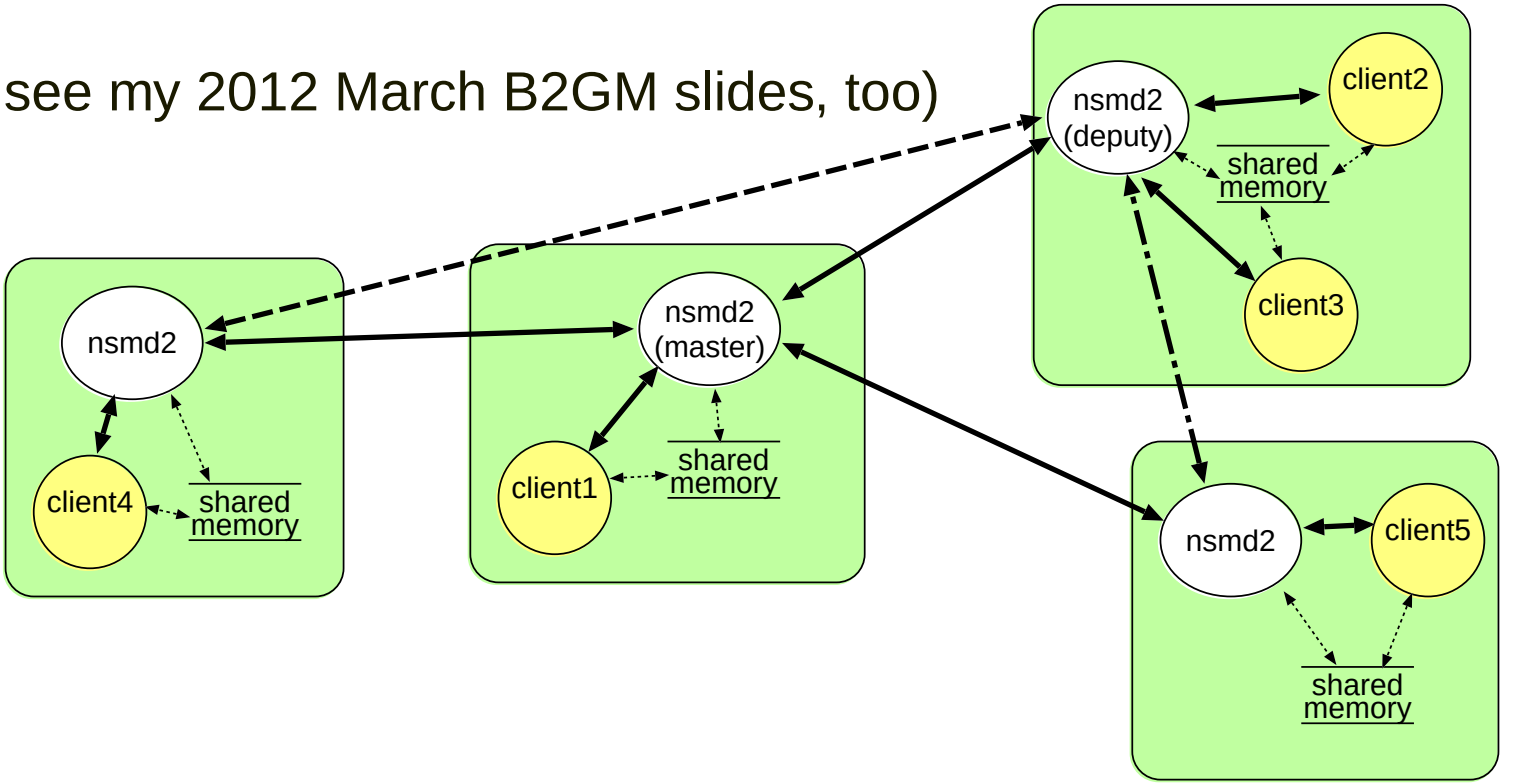
Conclusion

- **EUDAQ Run Control system is easy to understand** (to me)
- **EUDAQ Run Control can be easily controlled by NSM2, while leaving the rest of the EUDAQ world as it is**
- **Still a lot more to do on NSM2 side**

Backup

Introduction to NSM

(please see my 2012 March B2GM slides, too)



- nsmd2 daemon process is running on each host
- user (client) program communicate with other user program through nsmd via NSM API
- network shared memory is visible as POSIX shared memory

NSM2

- **Full rewriting, instead of modifying existing code**
- **Aims of changes**
  - Making the system more robust against a flood of messages
  - Reducing / removing hard coded features
  - Make the behaviour more deterministic
  - Make the allowed node-name longer, memory size bigger, etc
  - Multi-network in one process: supporting a network bridge
  - Factorize NSM-generic code and Belle(-II) specific features
  - (A better support on database, GUI)
  - Still minimizing the look-and-feel difference from NSM1
  - Make it free from 2038 (unix-year) problem — 64-bit time_t
- **Code writing started in November/2012**
  - **Today, alpha version of NSM2 package is released.**
    `https://belle2.cc.kek.jp/~twiki/bin/view/Detector/DAQ/NSM2`

NSM2 and Belle II

- **NSM2 has two library layers: corelib and belle2lib**
  - NSM1 has some Belle-dependent features embedded in the library
- **corelib**
  - Nothing is assumed about usage, message or node type
  - All functions start with `nsmlib_`
  - Flexibility at the cost of allowing wrong combinations
- **b2lib**
  - Heavily assumes the run/slow control model of Belle II
  - All functions start with `b2nsm_`
  - All necessary functions are (re)defined to avoid direct call to corelib
  - Limited flexibility to minimize mistakes

NSM messages

- **How to represent a message**
  - A message code is a text-string, with a 16-bit hash code
    (In NSM1, it was a hard-coded symbol in an include file)
  - In NSM2, the node name also has a hash code
    (in NSM1, a linear search was used)
  - Hash is maintained in the MASTER nsmd, similarly to the nodename
- **How does it work** (implemented, slightly modified since Jan)
  - Message is define on-the-fly, by registering a callback function
  - Number of parameters is variable (before it was fixed to 2)

(master)

```
int p[2];
p[0] = runno;
p[1] = runtype;
b2nsm_sendreq("CDC","START",2,p);
```

(CDC)

```
void startfunc(NSMmsg *m,NSMcontext *c) {
   :
  b2nsm_ok(m,"RUNNING","run %d",m->pars[0]); }
   :
b2nsm_callback("START", startfunc);
```

How to wait for NSM messages

- **Signal handler (default)**
  - User program can do anything else while waiting (e.g. GUI)
  - May receive another message while processing the previous one
  - This was the only method in NSM1
- **Wait function (by explictly calling `b2nsm_wait`)**
  - Cannot do anything else until timeout
  - Message order is guaranteed
  - Has to be decided when calling `nsm_init2`
- **Writing a code to use select**
  - Socket is available in the `NSMcontext` object (returned by `nsm_init`)
  - One can wait for NSM and something else at the same time
  - Once there's something to read, one can call `b2nsm_wait`
  - Need a bit more programming skill

NSM data

- **data structure definition** (implemented)
  - In NSM1, it was a hard-coded `struct` in an include file
  - In NSM2, the include file is parsed on the fly
    (only a simple struct definition can be used)
  - NSM has to know it to convert into network-byte-order
  - Revision number to make sure the same data structure is used

```
(ttd_data.h)
const int
  ttd_data_revision = 1;
struct ttd_data {
   int32 runno;
   int32 evtno;
   double trigrate;
   byte status[128];
};
```

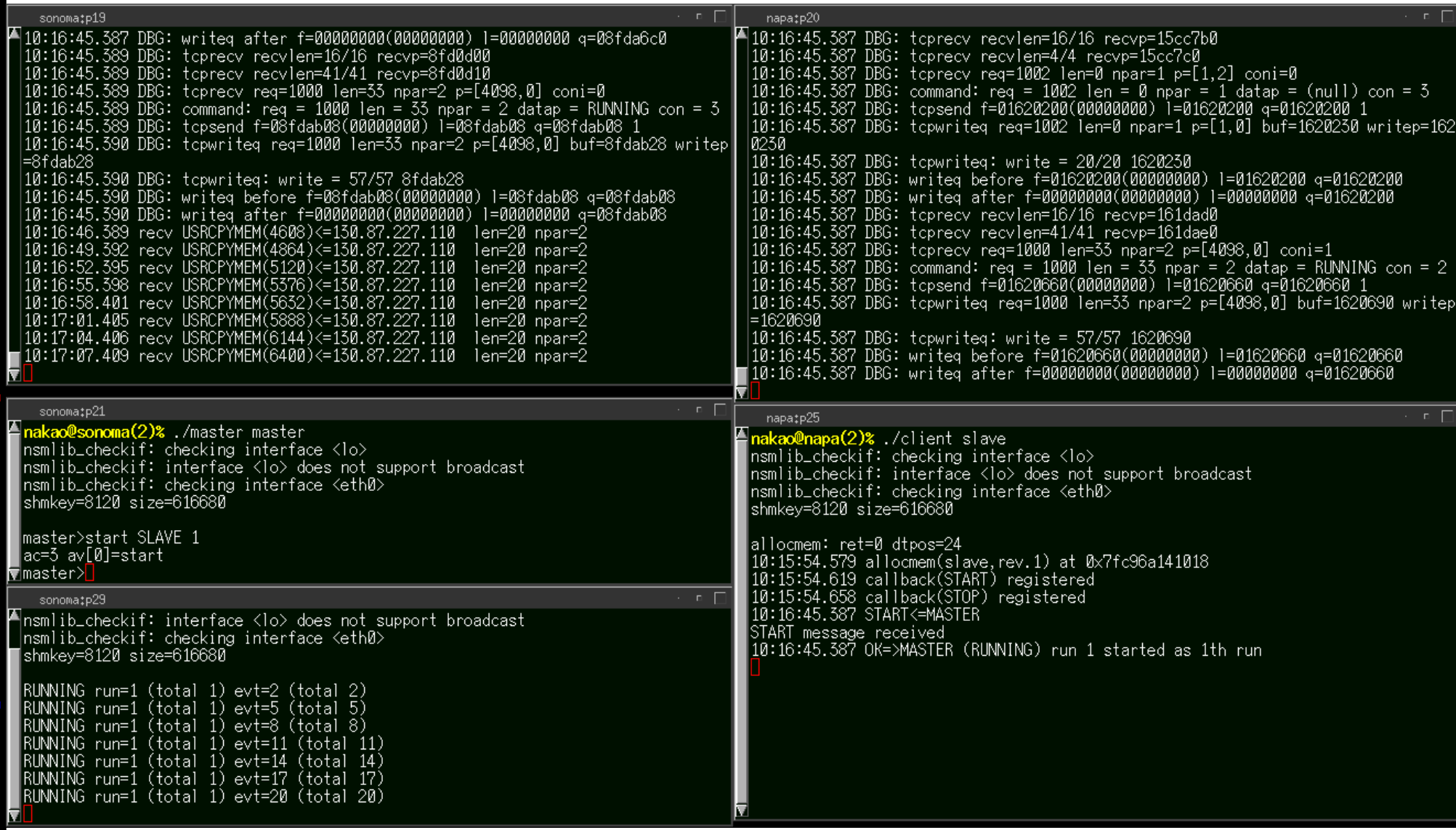
```
(main.c)
#include "ttd_data.h"
 :
float update_freq = 5; // every 2 sec
struct ttd_data *datap = allocmem("ttd_data",
        "TTD", ttd_data_revision, update_freq);
```

```
(other.c)
#include "ttd_data.h"
 :
struct ttd_data *datap
 = openmem("ttd_data", "TTD", ttd_data_revision);
```

NSM data sharing

- **UDP broadcast** (implemented)
  - Allocated data structure is shared between nsmd nodes
  - Single UDP packets if the size is below 1484 bytes,
    very minimal header, 8-byte for UDP and **only** 8-byte for NSM
    (but non-8-byte-boundary may cause a problem, probably will switch to 1480 byte)
  - Above this size, data update time may not be uniform, but time stamp difference is recorded and can be monitored
    (it was not clear in NSM1 when a particular part of data is updated)
  - Data size in 16-bit word, up to 65,296 byte (44×1484)

NSM2 screen shot

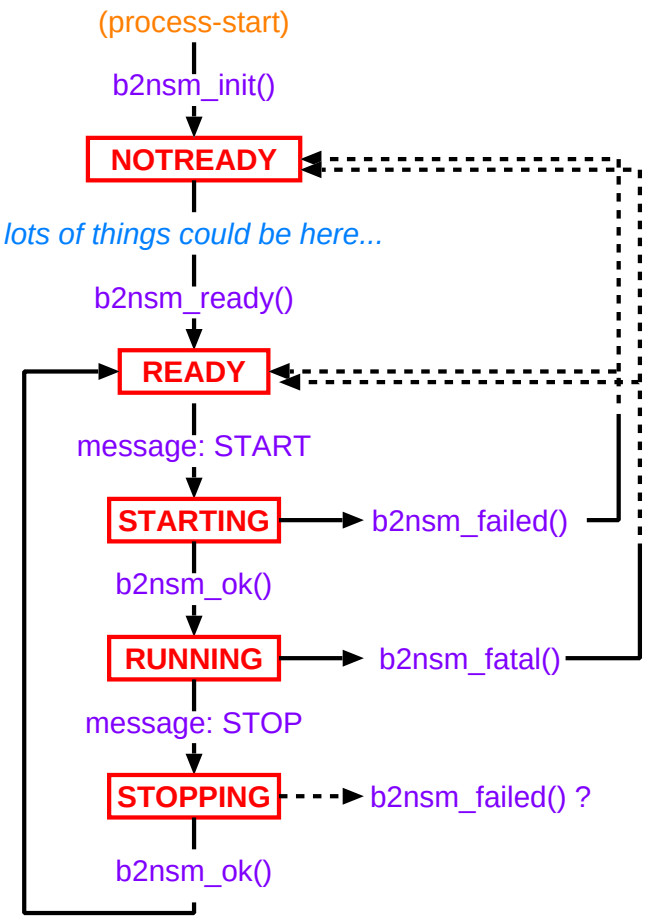


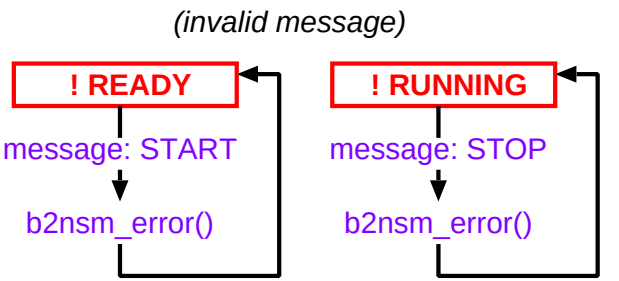
2 hosts (2 nsmd2), 3 processes (master, client, readdat)

Missing items in the alpha release

- **Missing library functions** (e.g., `nsmlib_closemem`, `...`)
- **Remote logging function** (e.g., `b2nsm_warning`, `...`)
- **Belle II run scheme design**
- **Various stress tests**
  - Many nodes (up to limit)
  - Large datasize, fast data update
  - DoS attack-like flood-of-message handling
  - Colliding two NSM networks
  - Handling incorrect network setting, disconnecting network, etc
- **Full document** (especially reference manual for the library section)

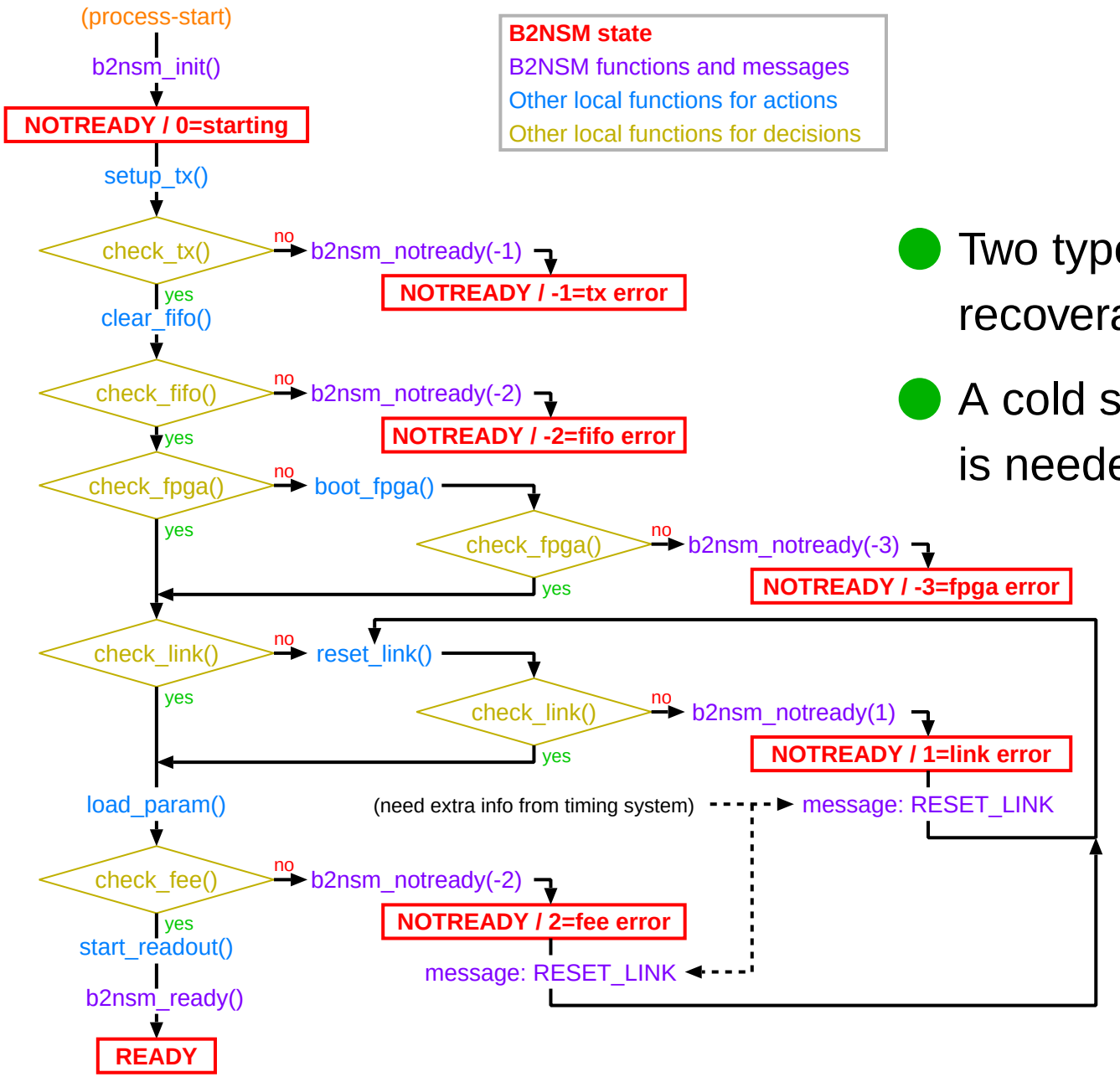
Run state model for a readout subsystem





- Acknowledge the NSM message by one of `b2nsm_ok, b2nsm_failed, b2nsm_error`
- State transitions, but states are kept inside belle2nsm library
- The system can be controlled by somebody other than MASTER
- Making a subsystem READY is the nasty part of the procedure. . .

Start-up model for COPPER



- Two types of NOTREADY: recoverable / non-recoverable
- A cold start-up procedure is needed in addition

Run control wishlist

- **Constraints**
  - Cold start, firmware programming, parameter downloading, link establishing have to be done
- **Wish**
  - Readout cycle to be all the time running, regardless the HV / accelerator condition (idle running)
  - In case of an error in one sub-system, the rest should not stop
  - Transition from idle running to real running is a quick run number and run type change in the fast TT link command
- **“Problems”**
  - Detectors are willing to initialize everything every time regardless the deadtime caused by it
  - Not easy to synchronize at the event builder
  - FPGA reprogramming has to be included as we expect SEU

Status and Plan

- **Now testing with 4 PCs on general network**

  (a few more can be quickly added if needed)

- **Many new key features are now implemented, but ~~still not~~ ready for a test use**

- **Target dates**
  - January DAQ WS for alpha release (for PocketDAQ) did not happen
  - March B2GM for ~~beta~~ alpha release
  - March B2GM for a first proposal of the state model and run start sequence or something alike for Pocket-DAQ

- **After March B2GM**
  - **Throughput measurement with a dedicated fast (GbE) network**
  - **Serious error handling tests with a large system ($\gg$ 10 nodes)**
  - ~~Master~~ **COPPER/HSLB/FTSW control for PocketDAQ**