TINE Release 4.0 News

(Oct 10, 2008: That was the week that was !)

“What a long, strange trip it’s been ….”

TINE Kernel 4.0.4 (RC)

- Kernel Revision now at 4.0.4, what's new (C – not java!):
  - SetPacketMTU() now accepts an MTU (maximum transport unit) up to 64 Kbytes (c only).
    - Shared memory transfer fixed at 64 kbyte 'datagrams'.
    - TCP/IP at ~ 6 Kbyte 'parcels'.
    - Default is 1472 ! (some OSes can't handle more than this!)
  - ExecLink() is now re-entrant, thread-safe.
    - Multiple 'simultaneous' synchronous calls work fine
    - But please avoid if possible!
      - Use asynchronous CM_SINGLE calls (via AttachLink())
      - Grouped with a single callback -> serializes nicely with high efficiency on both the client and server
  - "NALARMS" can now return a device-specific 'snapshot'.
  - Refactor Image source header.

TINE Kernel 4.0.4

- Systematic “CYCLER” logic can deterimine the System Data Stamp.
  - Servers : look for a server called “CYCLER” in ‘my’ context.
  - If it exists and delivers a property called “CycleNumber” then listen for global events!
  - Can supply a trigger callback!
  - The System Data Stamp is set to the incoming values of “CycleNumber” (post trigger).
  - Data requests are tagged with the System Data Stamp along with the time stamp (and user data stamp).
  - Good to ~ 10 Hz ?

TINE Kernel 4.0.4

- Local History archive can now archive data records with
  - Data time stamp
  - System Data Stamp
  - User Data Stamp
- TO DO:
  - Introduce format type CF_HISTORY
    - Archive and retrieve any data type
    - Use data format = CF_HISTORY data tag to specify the requested type if exotic otherwise:
    - CF_DBLDBLDBLDBL returns

      value + timestamp + system stamp + user stamp

TINE Kernel 4.0.4

- Bug-fixes and Embellishments:
  - Trap ‘server_redirection’ for all cases if CF_DEFAULT is requested (php).
  - Remove CA_READ access if call requests 0 data elements
  - Protect against duplicate history record indices
  - Problem when displaying large string data with debug level > 4 fixed (D. Franke).

TINE Kernel 4.0.4

- Java
  - Major refactoring in order to handle ‘extreme’ cases (1000s of links : more later).
  - Cycle trigger logic.
  - Debug output more readable.
  - Check alarm data array boundaries
  - Synchronize the device array list where ever it’s used.
  - “unknown address” run-time exception now in the TLink() constructor
    - Was in the execute() and attach() methods.
    - Else -> if unhandled the link.close() method might not be called and the link table fills up!

TINE Kernel 4.0.4
(last week's java bug …)

```
}
private long systemSrvCycle()
{
  long cycletime = System.currentTimeMillis();
  long msecToSleep = 10;
  if (!isInitialized) return STD_CYCLE_INTERVAL*200;
  if (gLinkTablesAccessed) return STD_CYCLE_INTERVAL;
  if (isInsideCycle) return STD_CYCLE_INTERVAL;
  isInsideCycle = true;
  try
  {
    if (debugLevel > 25) DbgLog.log("systemSrvCycle","SystemCycle called at " + new Date().to
    // basic server service engine (called at the polling rate)
    // call each module's update routine ...
    for (int i=0; i<numEqmTableEntries; i++) eqmTable[i].update();
    // process all external requests ...
    doScheduler(); // check contract list ...
    if (registrationPending && cycletime - registrationTime > 1000)
    { // anything still need to be sent out ?
      int doneCounter=0;
      for (int i=0; i<numEqmTableEntries; i++)
      {
        if (eqmTable[i] == null) continue;
        if (!eqmTable[i].isRegistered) SendRegisteredExportToENS(eqmTable[i]);
        else if (!eqmTable[i].grpRegistered) joinEnsGroup(eqmTable[i]);
        else doneCounter++;
      }
      if (doneCounter == numEqmTableEntries) registrationPending = false;
      registrationTime = cycletime;
    }
  }
  catch (Exception e)
  {
    TFecLog.log("systemSrvCycle", "unhandled exception "+e.getMessage());
  }
  if (gCycleDelay > 10) msecToSleep = gCycleDelay;
  isInsideCycle = false;
  return msecToSleep;
}
```

Bug-fixes, Embellishments, Works in progress …

- “Keyword” stock property:
  - <property>.KEY parsed from description
    - e.g. "[0:1000 V][0:500 ms]<TRC>Transient Recorder voltage vs time“ -> returns “TRC”.
  - tget in scripts.
    - Unix world launches a tineRepeater to handle the requests.
      - Communicates via pipe
      - Starts a listener for read data sets
      - Avoids bothersome queries to ENS as well as synchronous polling of the target server.
    - Nothing there for windows yet!

Extreme Scenarios

- *Case 1:* 1000s of links
  - 1400 individual links @ 2.5 Hz
  - Was working in java rather poorly
    - Frequent timeouts (client reconnects)
    - ~ 300 ms to traverse and send out the contract list per client -> 2nd client = ~600 ms > 400 ms!
    - Subscriptions handled individually one after the other with synchronization around the connection table object! -> hard to get the request in !
    - You would timeout, too, wouldn't you?
  - Use ArrayLists rather than LinkedLists
  - Travers only what is necessary in the connection table
    - Break the connection table up into a working table and an 'appendToList' table.
    - ~ 70-90 ms to traverse the list per client
  - Synchronize on a finer scale
    - Incoming requests not competing with the same synchronization object as the connection table!
  - Now works efficiently for several clients !

Extreme Scenarios

- But … there are still limits!
- Tricks and Tips:
  - Use Structures to serialize the data
    - Maybe 1400 calls can become 10 ?
  - Use Bitfields !
    - 32 bits to an integer -> 50 calls ?
    - Bitfields let you name and address each bit if you want
    - CM_DATACHANGE will save lots of network traffic (most of the 90 ms is in the delivery of the data)
      - Maybe 1 single ‘watchdog’ link at CM_TIMER
  - Lots of clients : Use the CM_NETWORK flag

Extreme Scenarios

- **Case 2:** Lots of Megabytes / sec
  - UDP Datagrams -> SetPacketMTU(64000)
  - SystemAssignBufferSpace(large !)
  - SetClientRecvQueueDepth(0)
  - SetRequireAcknowlegments(0)
  - Good Hardware!

  - Lots of clients -> use CM_NETWORK!

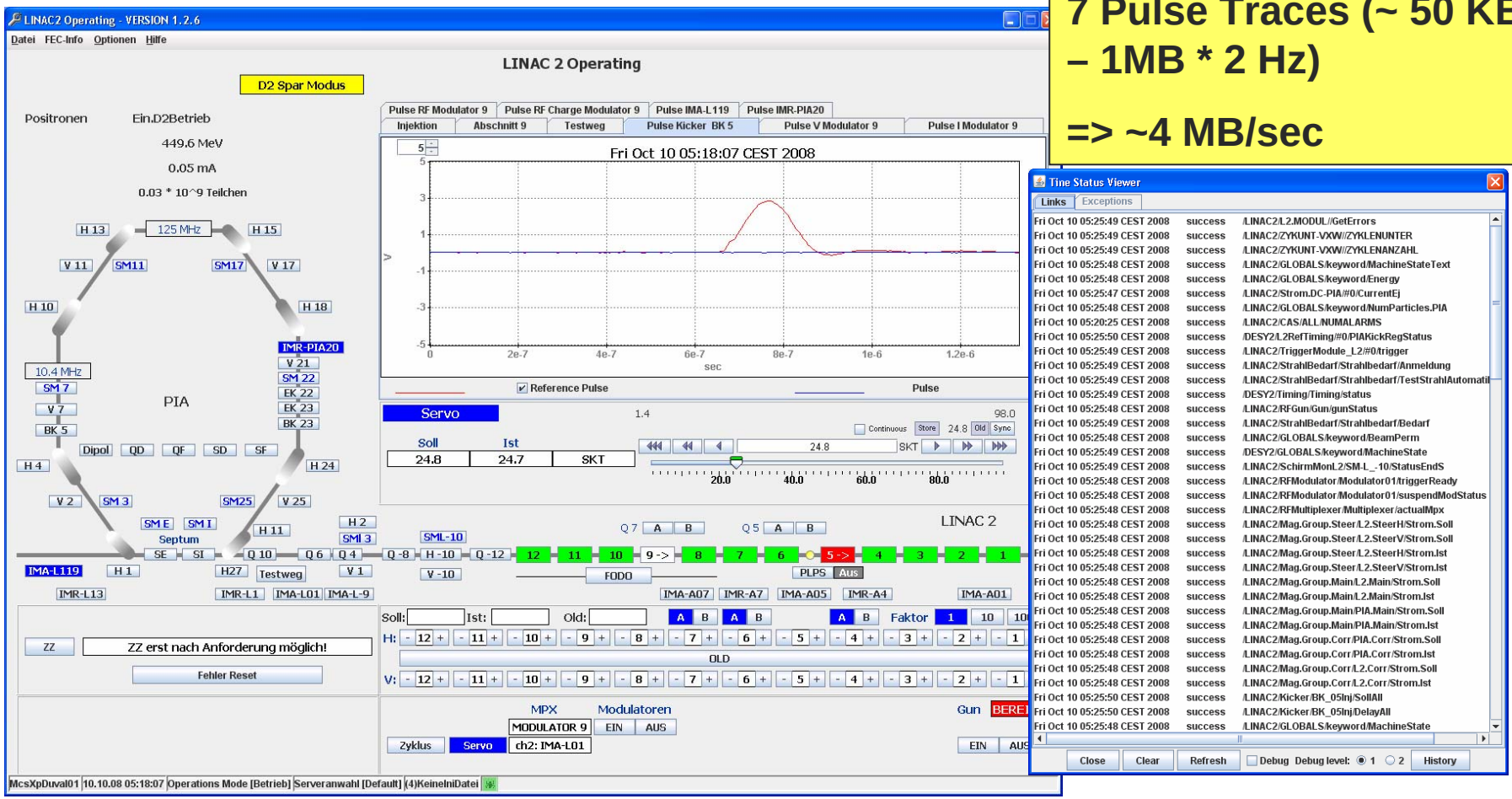
Extreme Scenarios

Case 3: Lots of links + Lots of data

> 200 Links

7 Pulse Traces (~ 50 KB – 1MB * 2 Hz)

=> ~4 MB/sec



Extreme Scenarios

- Set the socket receive buffer size (java):
  - TInitializerFactory.*getInstance*().getInitializer().setClnRcvBufferSize(256000);
  - -Dcln.RECV_BUFFER_SIZE=256000
  - etc.
- C:
  - int SystemAssignBufferSpace(UINT32 rcvBufferSpace,UINT32 sndBufferSpace);