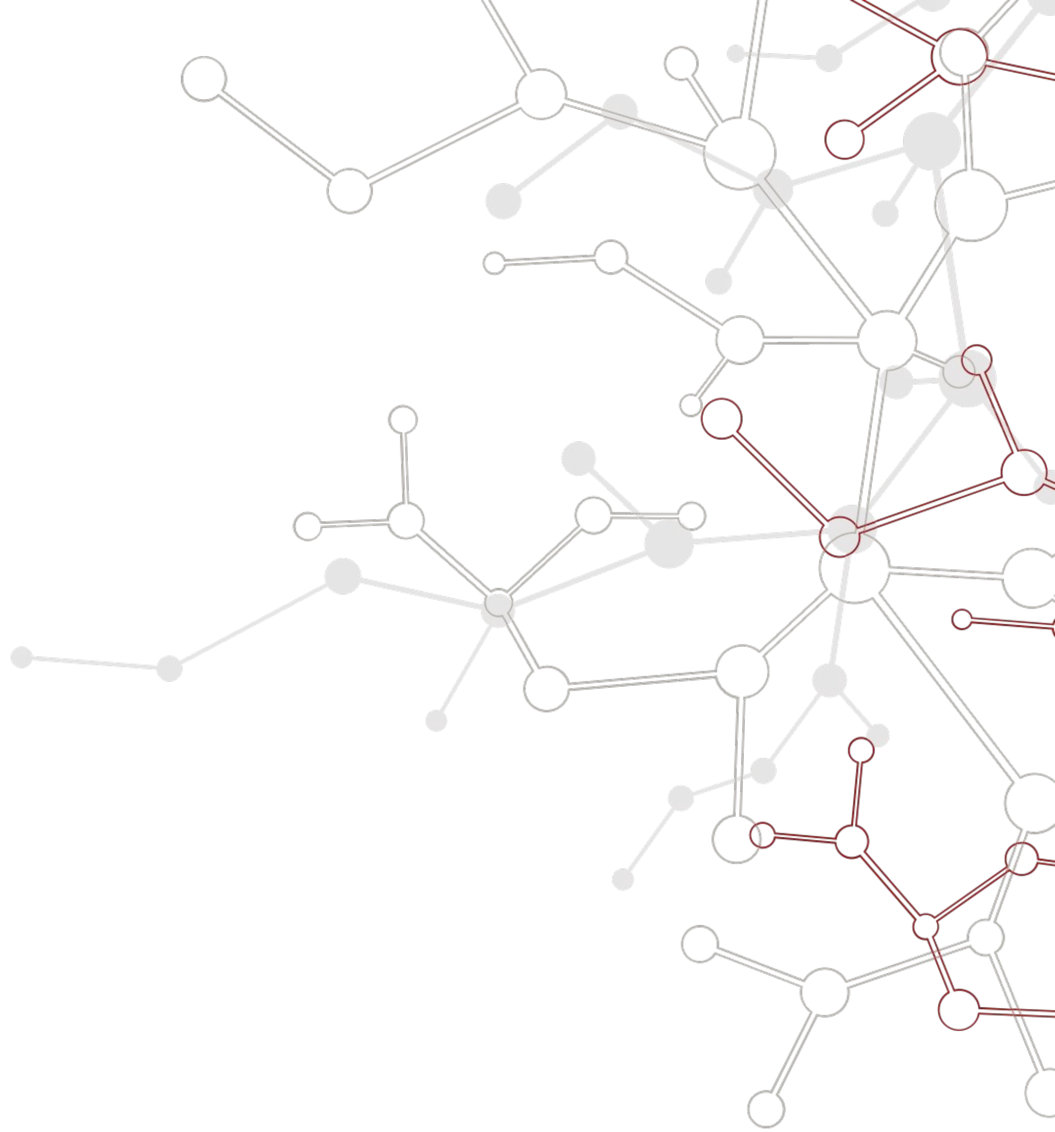
RTEMS & NFSv2

Tuning for reliable operation

Jeremy Lorelli

Technology Innovation Directorate - Instrumentation Division
Controls Software Engineering Department
Embedded Systems Group

January 25th, 2024



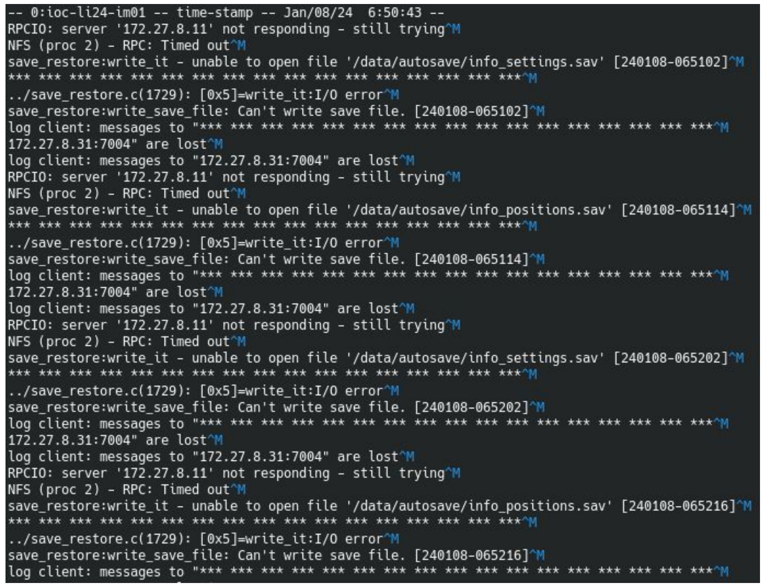
Outline

Background

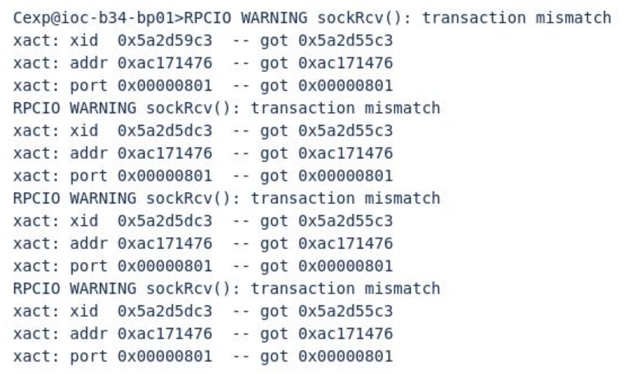
- RTEMS = Real-Time Executive for Multiprocessor Systems
  - Used to be “Real-Time Executive for Military Systems”
    - And before that: “Real-Time Executive for Missile Systems”
  - Real-time operating system (RTOS)
  - No concept of userspace
  - Networking stack
    - RTEMS 6+ ***libbsd*** is FreeBSD 12 based
    - RTEMS 5 and below use older FreeBSD-based networking stack (aka “***legacy networking*** stack”)
- SLAC runs RTEMS 4.9.4 and 4.10.2 today
  - Used to use 4.7.X and earlier versions of 4.9.X/4.10.X
- Legacy networking:
  - NFSv2 only, driver written by Till Straumann for SLAC
- NFSv2 (circa ~1990)
  - RPC based protocol
  - UDP only
    - UDP = no transport layer reliability
    - NFSv4 adds support for TCP

Overview

- Certain RTEMS IOCs fail to write autosave files
  - Some partially succeed, others are completely unable
  - They don't seem to recover without a reboot
  - Network conditions are fine
  - Seems to be exclusive to RTEMS IOCs
- Reported off an on for ~10 years
- Spring 2023 power outage seems to have triggered the issue at large
  - Prior to this, it was not observed as frequently
- Some initial theories we had
  - Bad network cards, damaged by power outage
  - Bad network switch causing excessive packet loss/damage
  - Previously fixed RPC/NFS driver bug has resurfaced
  - NFS server bug
- CATERs: #146947, #162668, #164090, #166516, #97639



```
-- 0:ioc-li24-im01 -- time-stamp -- Jan/08/24  6:50:43 --
RPCIO: server '172.27.8.11' not responding - still trying^M
NFS (proc 2) - RPC: Timed out^M
save_restore:write_it - unable to open file '/data/autosave/info_settings.sav' [240108-065102]^M
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***^M
../save_restore.c(1729): [0x5]=write_it:I/O error^M
save_restore:write_save_file: Can't write save file. [240108-065102]^M
log client: messages to "*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***^M
172.27.8.31:7004" are lost^M
log client: messages to "172.27.8.31:7004" are lost^M
RPCIO: server '172.27.8.11' not responding - still trying^M
NFS (proc 2) - RPC: Timed out^M
save_restore:write_it - unable to open file '/data/autosave/info_positions.sav' [240108-065114]^M
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***^M
../save_restore.c(1729): [0x5]=write_it:I/O error^M
save_restore:write_save_file: Can't write save file. [240108-065114]^M
log client: messages to "*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***^M
172.27.8.31:7004" are lost^M
log client: messages to "172.27.8.31:7004" are lost^M
RPCIO: server '172.27.8.11' not responding - still trying^M
NFS (proc 2) - RPC: Timed out^M
save_restore:write_it - unable to open file '/data/autosave/info_settings.sav' [240108-065202]^M
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***^M
../save_restore.c(1729): [0x5]=write_it:I/O error^M
save_restore:write_save_file: Can't write save file. [240108-065202]^M
log client: messages to "*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***^M
172.27.8.31:7004" are lost^M
log client: messages to "172.27.8.31:7004" are lost^M
RPCIO: server '172.27.8.11' not responding - still trying^M
NFS (proc 2) - RPC: Timed out^M
save_restore:write_it - unable to open file '/data/autosave/info_positions.sav' [240108-065216]^M
*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***^M
../save_restore.c(1729): [0x5]=write_it:I/O error^M
save_restore:write_save_file: Can't write save file. [240108-065216]^M
log client: messages to "*** *** *** *** *** *** *** *** *** *** *** *** *** *** *** *** ***^M
```



```
Cexp@ioc-b34-bp01>RPCIO WARNING sockRcv(): transaction mismatch
xact: xid  0x5a2d59c3  -- got 0x5a2d55c3
xact: addr 0xac171476  -- got 0xac171476
xact: port 0x00000801  -- got 0x00000801
RPCIO WARNING sockRcv(): transaction mismatch
xact: xid  0x5a2d5dc3  -- got 0x5a2d55c3
xact: addr 0xac171476  -- got 0xac171476
xact: port 0x00000801  -- got 0x00000801
RPCIO WARNING sockRcv(): transaction mismatch
xact: xid  0x5a2d5dc3  -- got 0x5a2d55c3
xact: addr 0xac171476  -- got 0xac171476
xact: port 0x00000801  -- got 0x00000801
RPCIO WARNING sockRcv(): transaction mismatch
xact: xid  0x5a2d5dc3  -- got 0x5a2d55c3
xact: addr 0xac171476  -- got 0xac171476
xact: port 0x00000801  -- got 0x00000801
```

NFS/RPC Driver Details

- UDP is simple & stateless, so reliability must be implemented by the driver
  - RFC 5531 (RPC v2) defines an “XID” to make room for reliability over UDP
- Reliability implemented using a “retry period”
  - If RPCIOD doesn’t receive a reply to the request within the period of time defined by the retry period, it retransmits the request
    - Same XID, same data
    - Mitigates the effect of packet loss or NFS server errors
  - The retry period is variable
    - Adjusted based on round trip time
    - Increased by 2x after each retry

RPC Control Flow Pseudocode

```
while True:
    while xact = sockRcv():
        nodeXtract(xact.node)
        # Ensure xid does not re-appear in table
        xact.xid += XACT_HASHS
        rtt = computeRoundTrip(xact)
        retry_period = computeRetryPeriod(rtt)
        wakeRequestor()

    for xact in newToSend:
        xact.age = now
        xact.trip = FIRST_ATTEMPT
        addToList(pendingTransactions, xact)

    # Handle the timeout queue
    for xact in pendingTransactions:
        if xact.tolive <= 0:
                xact.xid += XACT_HASHS
                xact.status = TIMEDOUT
                timeoutStats()
        else:
                res = sendTo(socket, xact)
                if not res:
                        handleError()
                if not isFirstTry(xact):
                        retry *= 2
        xact.trip = now
        xact.tolive -= timeSinceLastIter


    # Sleep until we need to retransmit one
    wakeThreadAfter(pendingTransactions[0].tolive)
```

Investigation: DEV

- Testing done on ioc-b34-bp01
  - Thanks Sonya!
  - mvme-6100, BPM IOC
- Initial test code
  - Read/write to IOC data directory in a loop every ~5 seconds
    - Random patterns, different file sizes up to 1M
  - Developed a small suite of networking utils for RTEMS, as an analogue to busybox/coreutils for Linux
    - ping, traceroute, packet loss checking tool
- Packet sniffing using IOTA 10G+ from Profitap

Investigation: Results (DEV)

- No packet loss, low latency, overall good network conditions
- File I/O fails at a low rate
  - Over a 72 hour test period, 4 file I/O calls failed due to timeout
- Lots of retransmissions being attempted by the RPCIO driver
  - Variable retry period seems to hover around ~8ms (as reported by rpcUdpStats())
  - This seems excessive…
- Pattern in error spew from ioc-b34-bp01
  - Between 7:30-7:34AM every morning, RPC times out
    - Turns out there were cronjobs dumping SQL databases at that time every morning
      - *surrey04b has a 1GB NIC that is easily saturated*
    - Murali staggered those cronjobs and modified the script to resolve the issue

Investigation: PROD

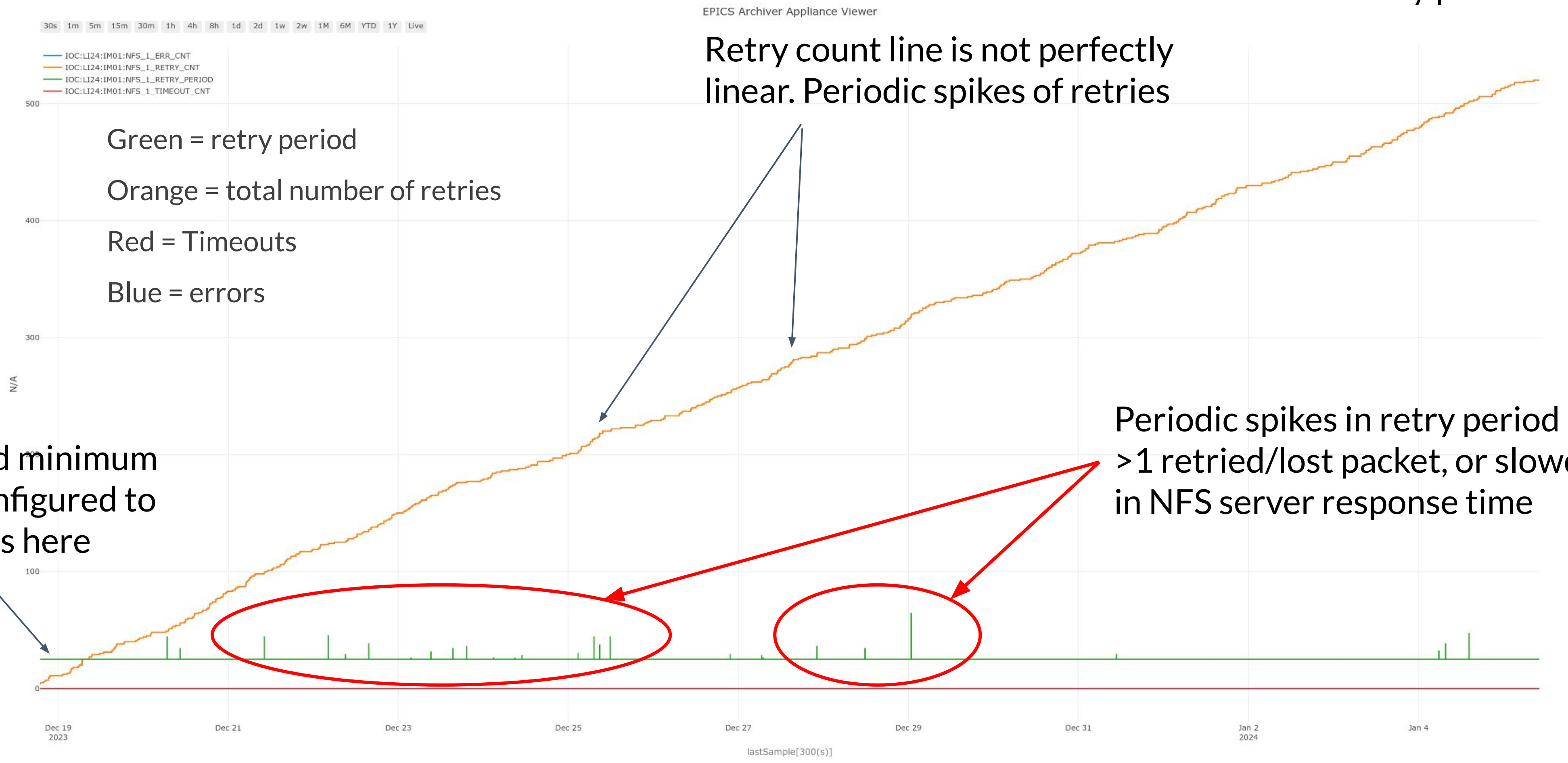
- The issues in DEV are unrelated
- Exposed NFS and network stats as PVs, integrated into iocAdmin
  - Planning to merge this upstream into iocStats
  - ~40 PVs total
- Deployed monitoring on ioc-li24-im01 in mid-December
  - Also included:
    - pvAccess resource leak fix
    - minimum bound for the retry period
  - Thanks to Kristi for the support!
- Stats collected over winter break, with some interesting results

```
IOC:LI24:IM01:NET_UDP_RECV
IOC:LI24:IM01:NET_UDP_SEND
IOC:LI24:IM01:NET_UDP_ERR
IOC:LI24:IM01:NET_TCP_RECV
IOC:LI24:IM01:NET_TCP_SEND
IOC:LI24:IM01:NET_TCP_ERR
IOC:LI24:IM01:NFS_0_MOUNT
IOC:LI24:IM01:NFS_0_REQ_CNT
IOC:LI24:IM01:NFS_0_RETRY_CNT
IOC:LI24:IM01:NFS_0_ERR_CNT
IOC:LI24:IM01:NFS_0_TIMEOUT_CNT
IOC:LI24:IM01:NFS_0_RETRY_PERIOD
IOC:LI24:IM01:NFS_0_NODE_CNT
… (up until NFS_4)
```

Investigation: Results (PROD)

- Period of interest is December 20th to January 4th

Data shown here is with a 25ms lower bound on the retry period



Green = retry period

Orange = total number of retries

Red = Timeouts

Blue = errors

Retry count line is not perfectly linear. Periodic spikes of retries

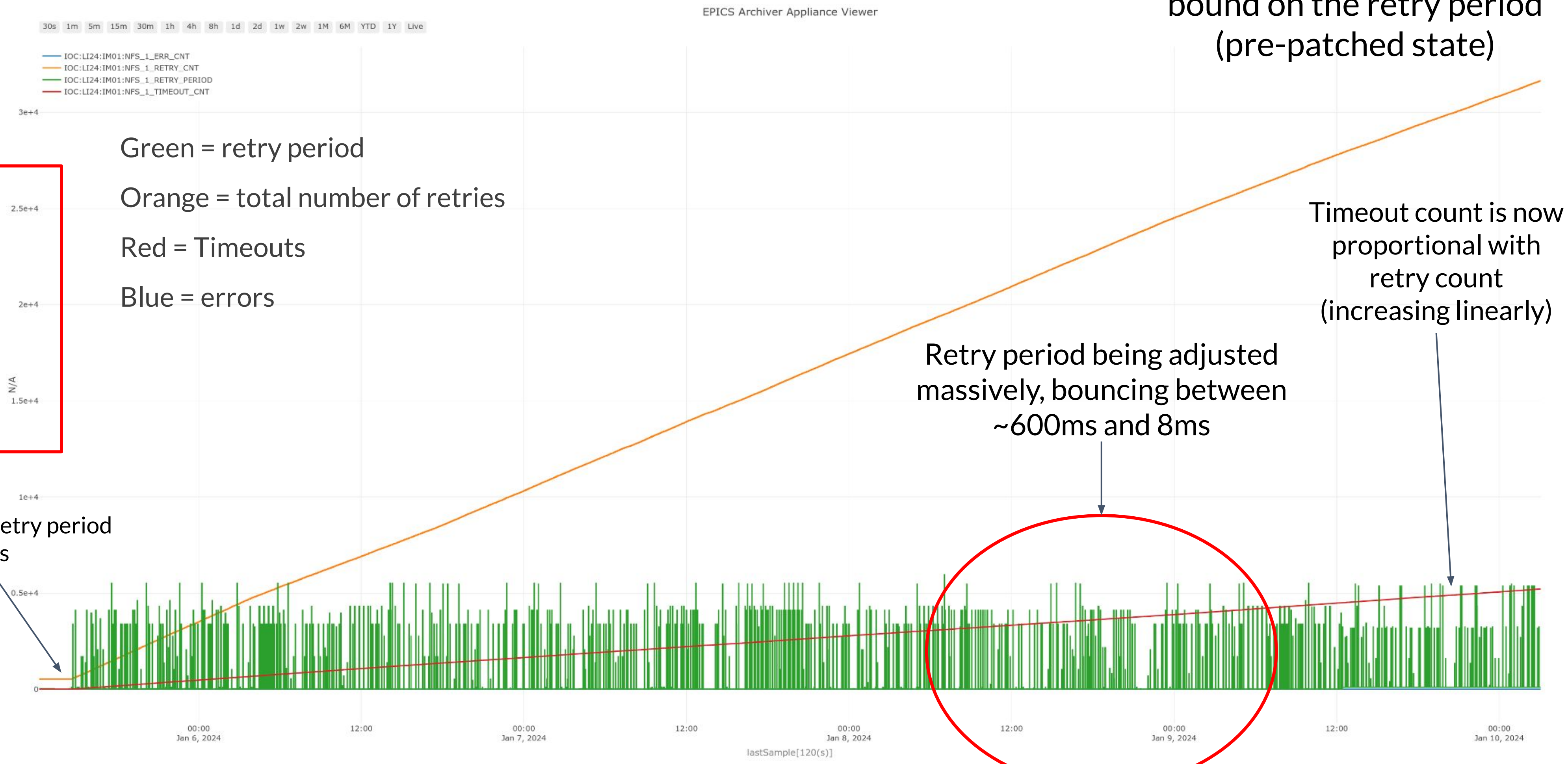
Periodic spikes in retry period indicate >1 retried/lost packet, or slowdowns in NFS server response time

Retry period minimum bound is configured to be 25ms here

Investigation: PROD (Results)

- Period of interest is January 5th to 10th

Data shown here is with a 0ms lower bound on the retry period (pre-patched state)



Thousands of retries

Green = retry period

Orange = total number of retries

Red = Timeouts

Blue = errors

Timeout count is now proportional with retry count (increasing linearly)

Retry period being adjusted massively, bouncing between ~600ms and 8ms

Point where retry period min set to 0ms

Patching RTEMS

- To mitigate the issue, we need to:
  - Adjust retry period equation, including bounds to prevent it from dropping too low
  - Add function that can be called from cexpsh to adjust limits and eq
- Summary of changes:
  - Adjusted retry period equation, imposed min/max bound on retry period
  - Added *rpcUdpSetRetryParams* to change retry period equation parameters
    - Although they're tweakable, the defaults are tuned well enough
  - ***Only RTEMS and ssrlApps will need to be recompiled***
- Tested on: ioc-b34-bp01, ioc-li24-im01. Both mvme-6100, 4.10.2
- Default settings equivalent to:

```
// min (ms), max (ms), multiplier, influence fraction
rpcUdpSetRetryParams(25, 3000, 8, 0.25)
```

  - Defaults are already tuned, this function is available for future proofing

Deploying RTEMS Releases

- Pull request **pending**: https://github.com/slaclab/rtems/pull/1 **(Branch: 4.10.2_PR_rpcio_retry_period)**
  - Once merged:
    - RTEMS 4.10.2 -> tag **4.10.2_slac_p3-1.0**
    - RTEMS 4.9.4 -> tag **4.9.4_slac_p3-1.0**
  - When booting, you should see:

```
Welcome to RTEMS 4.10.2-slac_p3-1.0 GeSys
```

- What's the best strategy for deployment?
  - Option 1: New patch level (i.e. rtems_p4)
    - EPICS base will need modification to point at the right place, IOCs will need to be recompiled, dhcp changes
    - We will need to do this once Till fixes the other RTEMS bugs regardless
  - Option 2: Recompile rtems_p3 in place
    - In this case, IOCs simply need to be rebooted to get the fix
    - **This is the method we recommend**
  - Option 3: Wait until other RTEMS bugs are fixed, then release new patch level

Deploying RTEMS Releases

- Rolling back to previous release:
  - **4.10.2_slac_p3-1.0** -> **4.10.2_slac_p3**
  - **4.9.4_slac_p3-1.0** -> **4.9.4_slac_p3**
  - Rebuild both RTEMS and ssrlApps

Closing Remarks

- Thanks to Kristi, Sonya and Till Straumann for their support
- RTEMS and RTEMS related drivers have been moved to GitHub
    - Good way to facilitate collaboration with Till and other RTEMS developers at SLAC
    - TID-ID-CSE has been moving packages and EPICS modules to GitHub ahead of AFS decommissioning
    - Links are in the next slide
- Providing configuration options for tunable parameters, like the retry period, should be the standard
    - Limits and other equation parameters tunable using *rpcUdpSetRetryParams*
    - If this becomes a problem again in the future, can be fixed by only changing scripts
- I have free RTEMS stickers!

Sources & Links

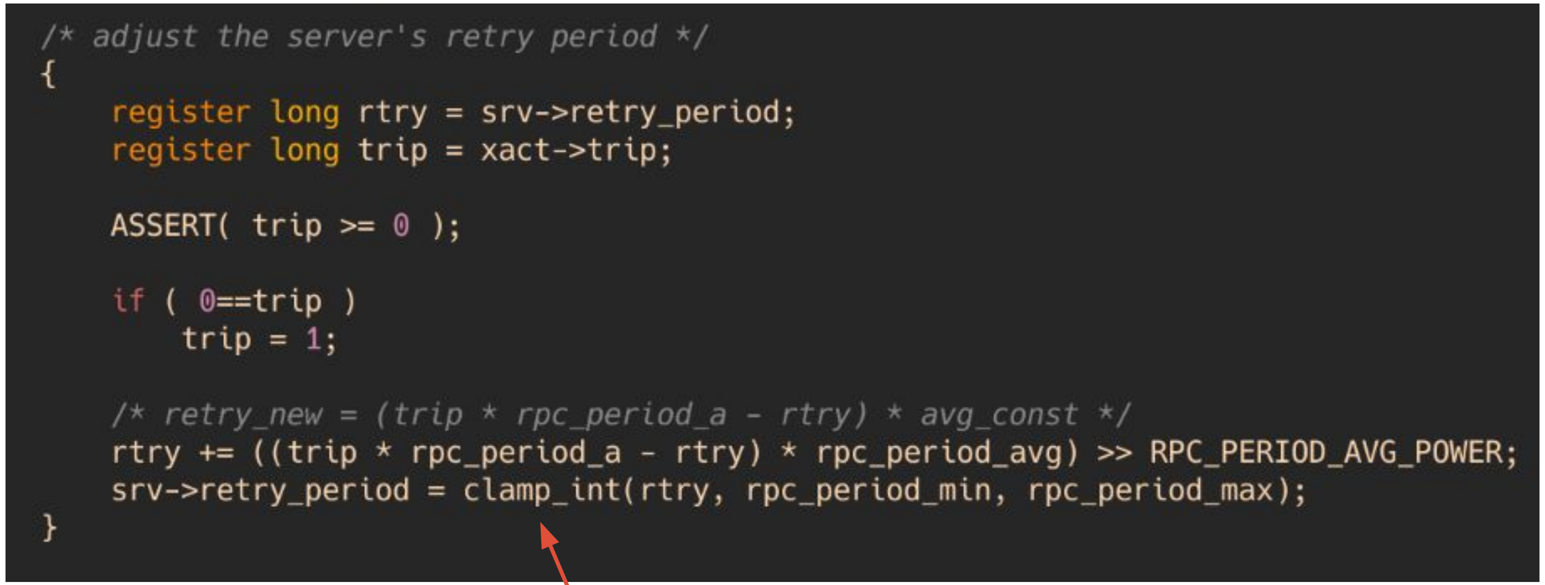
- https://datatracker.ietf.org/doc/html/rfc5531 (RPC, version 2)
- https://github.com/slaclab/rtems
- https://github.com/slaclab/rtems-svgm-bsp
- https://github.com/slaclab/if_gfe-rtems
- https://github.com/slaclab/rtems-beatnik-bsp
- https://github.com/slaclab/ssrl-ppc-bsp-vectors
- https://github.com/slaclab/porting-bsd-rtems
- https://github.com/slaclab/if_ex-rtems
- https://github.com/slaclab/if_em-rtems

Patch (reference)

- cpukit/libfs/src/nfsclient/src/rpc.c, line 1308-1324 (this is the most important part of the patch)

$$y = y + \frac{(T * A - y)N}{M}$$

- T = round trip time
- A = integral multiplier of round trip time
- N = Numerator of influence frac
- M = Denom of influence frac, constrained to power of two to allow use of right shift



```
/* adjust the server's retry period */
{
    register long rtry = srv->retry_period;
    register long trip = xact->trip;

    ASSERT( trip >= 0 );

    if ( 0==trip )
        trip = 1;

    /* retry_new = (trip * rpc_period_a - rtry) * avg_const */
    rtry += ((trip * rpc_period_a - rtry) * rpc_period_avg) >> RPC_PERIOD_AVG_POWER;
    srv->retry_period = clamp_int(rtry, rpc_period_min, rpc_period_max);
}
```

Clamp is the most important part!