ATCA-based IOC Crashes in Production

Update until 12/16/2022

Marcio Donadio / Control Systems Engineer / TID-ACS

2022

NATIONAL ACCELERATOR LABORATORY

Stanford University

U.S. DEPARTMENT OF ENERGY

Agenda

1

Introduction

Introduction

CATERS 158662 and 160141

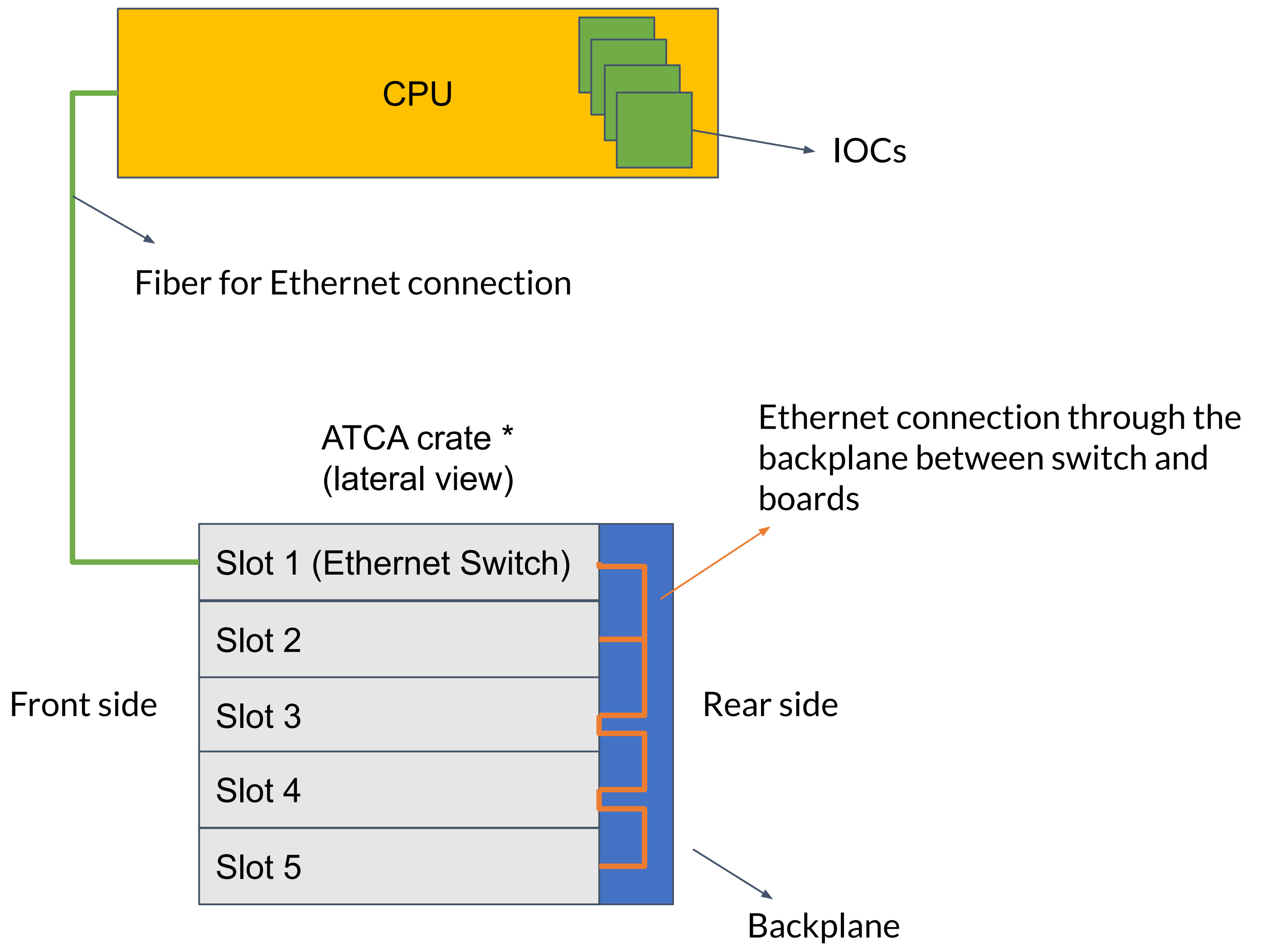
Different systems have been crashing in production:

- BPM IOCs are the most affected.
- Other systems also affected, but with less intensity:
  - MPS
  - BLEN
  - BCM
  - Wire Scanner
  - GMD

Error message on IOC console shows non captured exception from CPSW, complaining about timeout when trying to access an FPGA register.

CPSW Error: CPSW Error: CPSW **Error: No response -- timeout (Retries=0, Last timeout=500000)**: Success: /mmio/AmcCarrierCore/AmcCarrierBsa/Bsss/currPacketSize: Success at ../BsssYaml.cc, line 69 terminate called after throwing an instance of 'CPSWError' what(): CPSW Error: CPSW **Error: No response -- timeout (Retries=0, Last timeout=500000)**: Success: /mmio/AmcCarrierCore/AmcCarrierBsa/Bsss/currPacketSize: Success

Architecture Overview



ATCA crate overview

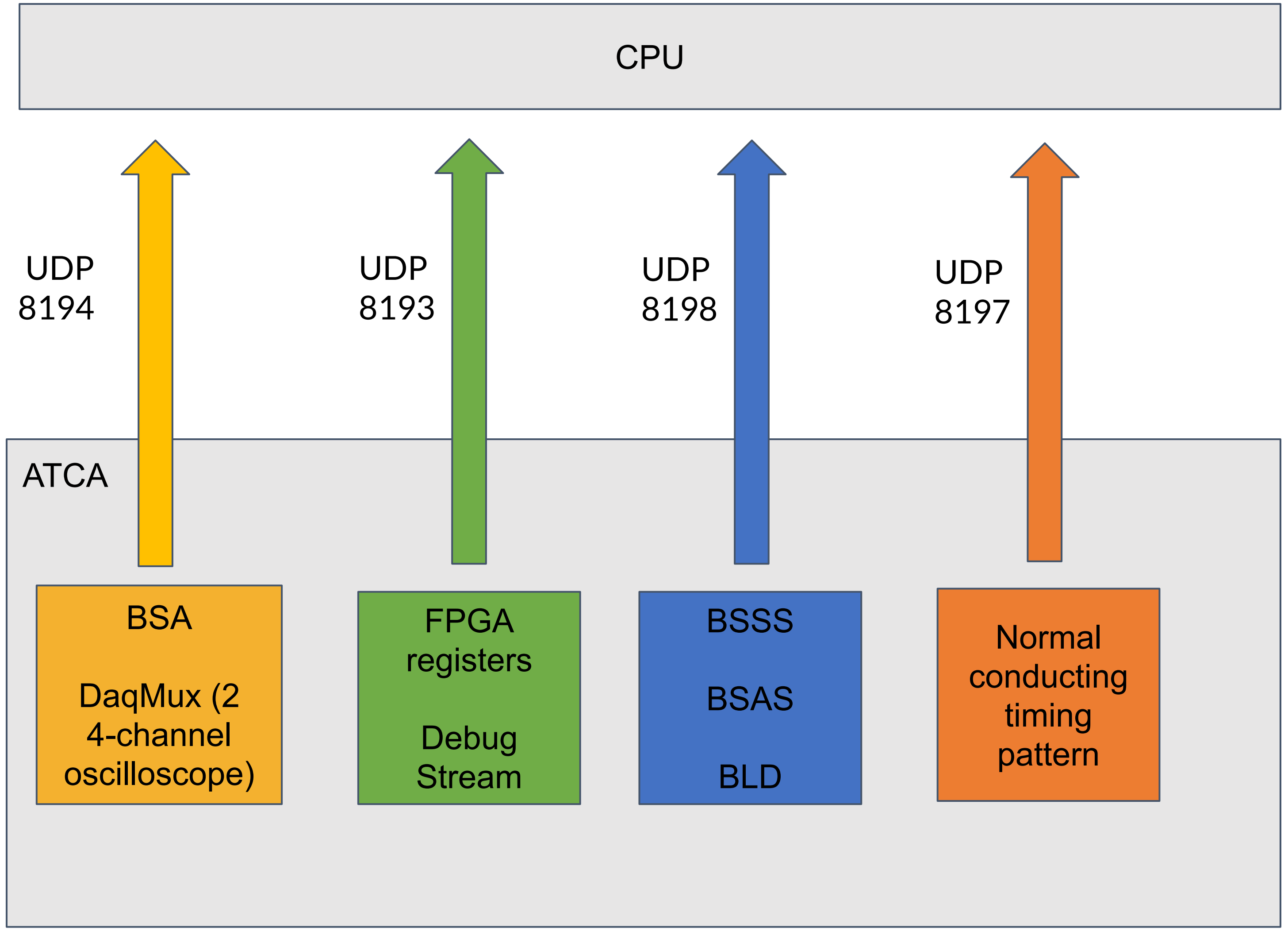
Communication between boards happen through the backplane which uses Ethernet.

Switch installed in slot 1 manages communication between boards and with the external world.

CPU where the IOCs run is connected to the ATCA crate through a 10 Gb/s fiber connection.

* RTM and all set of 7 slots not shown to make it easier to understanding

Architecture Overview



ATCA crate overview

A few protocols were created at SLAC. All use UDP.

The firmware can handle only a few UDP servers, each using one port, due to limited resources of the FPGA chip.

The firmware developers defined the use of each UDP port, which can share different functions.

Introduction

Devices used for the tests

cpu-l0b-sp02

- Crates
  - shm-l0b-sp02-1
  - shm-l0b-sp02-2
- IOCs
  - sioc-col0-bp01 until bp04 and sioc-col0-bp06
  - sioc-diag0-bp02 until bp05
  - sioc-htr-mc01
  - sioc-l0b-mp03 and sioc-l0b-mp04

cpu-sph-sp05

- Crate
  - shm-sph-sp05-1
- IOCs
  - sioc-sph-bp06 and sioc-sph-bp07
  - sioc-sph-mp05

Introduction

The army behind the tests

Marcio Donadio:

- Ernest Williams put Marcio in charge of leading and coordinating the effort.
- Analysis of the network dumps.
- Started a script to analyze the network dump and to retrieve a CPU "census" automatically.
- Conduct the tests to obtain strong correlations.
- Research of CPSW internals when trying to explain problems.
- Gathering of data from all teams involved.

Sonya Hoobler and Jeremy Mock

- Implemented the suggestions of changes in the IOCs and upgraded in production with record speed.
- Helped with the understanding of their systems and ideas to explain the root causes.
- Were always patient although the problems affect them daily (and nightly).

Matt Weaver

- Explained aspects of BSA and the timing system that helped to define how the tests could be driven.

Introduction

The army behind the tests

Larry Huckman and Ryan Herbst

- Explained the aspects of the many SLAC protocols used in transactions.
- Gave ideas for testing and tuning and important areas to check in CPSW.
- Ryan created an script based on a first version from me, which saved multiple hours of manual labor when analyzing network dumps.
- Analysis of network dumps.

Dawood Alnajjar

- Helped with the tuning of the network driver in LinuxRT.
- Provided data from crashes in the GMD.
- Started to prepare the test stand so we can continue with the tests in January 2023.

Mike Zelazny:

- Helped to "pilot" the Matlab script that controls the fault buffer cleaning and reading so we could correlate cause and effect for the problems (and did this during a Friday, late in the night).

Introduction

The army behind the tests

Ernest Williams:

- Hundreds of ideas that we could use for testing and description of previous experiences that could explain the problem.
- Provided connection among different teams.

Kukhee Kim:

- Always helping with the description of internals of BSA related and timing related EPICS modules.
- Analysis of network dumps and mapping of the analysis with the source code.

Namrata Balakrishnan, Leonid Sapozhnikov, Kyle Leleux:

- All provided data regarding crashes in Wire Scanner, BCM, and BLEN.
- Leonid helped with internals of the firmware application.
- Namrata described test cases that we could use to trigger the crashes.

Total: 13 people

If I forgot someone or a fact worth mentioning, please let me know and I'll fix the slides. I'm deeply sorry if I made this mistake.

2

Tools

Tools

IOTA 10G

Network tap, which is a device that "sniffs" the network communication between peers and record everything.

10 Gb/s capacity.

2 TB SSD storage capacity.

Grafana dashboards with filters to get a quick analysis of network usage.

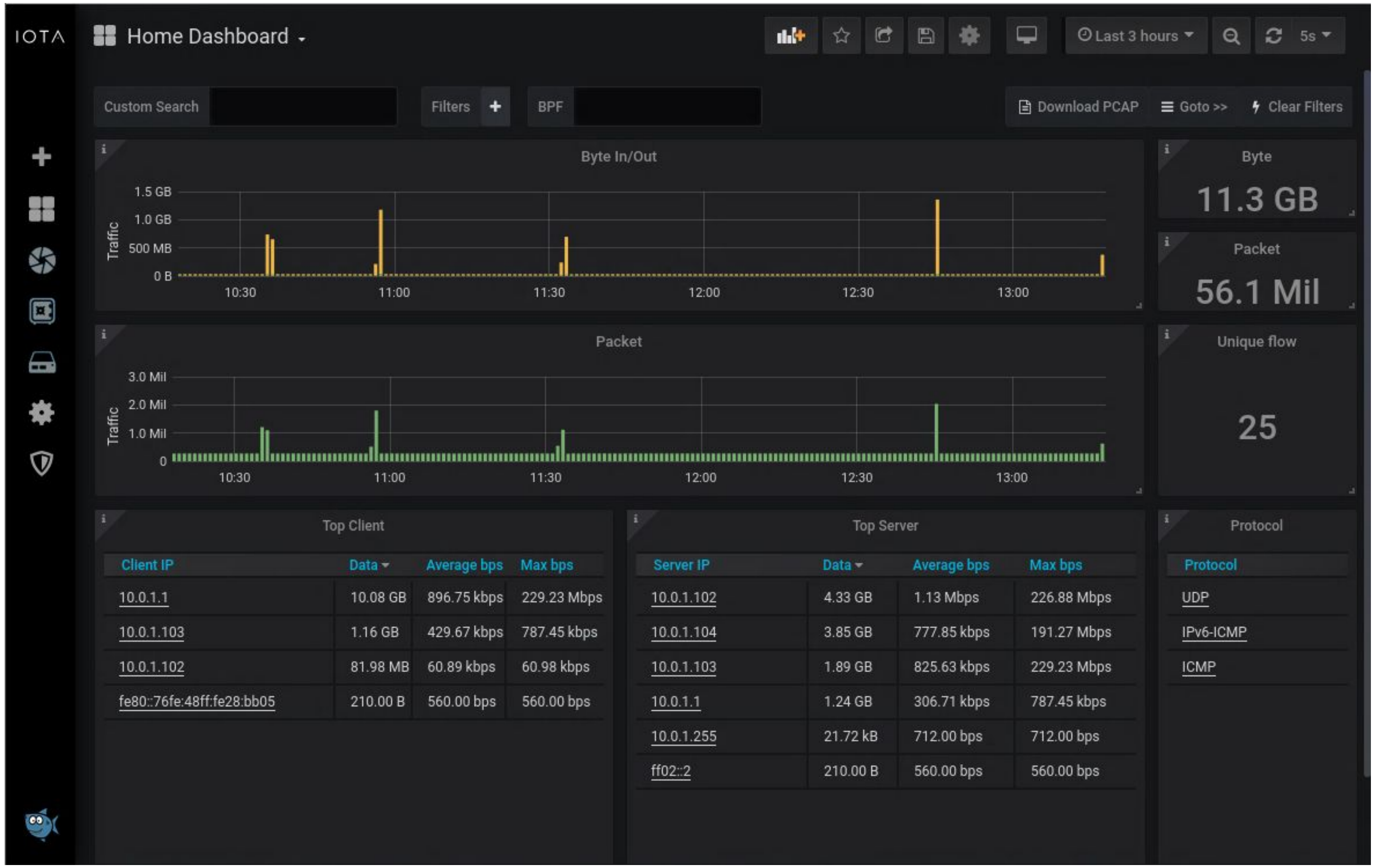
Tools

Why IOTA instead of using tcpdump in the command line?

We opted to use a network tap instead of using tcpdump because:

- It is difficult to record the exact moment when the crash occurs.
    - We would need to let tcpdump running for a couple of hours, wishing that something happens.
    - A lot of disk space in production would be used until a crash happened.
    - Someone would need to stay permanently monitoring the crashes to stop tcpdump right after. Otherwise, if let running overnight, dozens of GB of disk space would be used with one single dump file.
- IOTA keeps recording data 24/7:
    - Has its own SSD.
    - Auto cleans data after a period of time.
    - Records in chunks of 30 seconds, to control file size.
    - One can select a small portion of the plot by zooming in and download only the dump of that area.
- To extract reports with charts using tcpdump we would need to create a software to extract data, filter, and plot.
- Ready to use IOTA reports makes analysis of the traffic a lot simpler.

Tools



Byte 11.3 GB

Packet 56.1 Mil

Unique flow 25

Top Client

Client IP	Data	Average bps	Max bps
10.0.1.1	10.08 GB	896.75 kbps	229.23 Mbps
10.0.1.103	1.16 GB	429.67 kbps	787.45 kbps
10.0.1.102	81.98 MB	60.89 kbps	60.98 kbps
fe80::76fe:48ff:fe28:bb05	210.00 B	560.00 bps	560.00 bps

Top Server

Server IP	Data	Average bps	Max bps
10.0.1.102	4.33 GB	1.13 Mbps	226.88 Mbps
10.0.1.104	3.85 GB	777.85 kbps	191.27 Mbps
10.0.1.103	1.89 GB	825.63 kbps	229.23 Mbps
10.0.1.1	1.24 GB	306.71 kbps	787.45 kbps
10.0.1.255	21.72 kB	712.00 bps	712.00 bps
ff02::2	210.00 B	560.00 bps	560.00 bps

Protocol

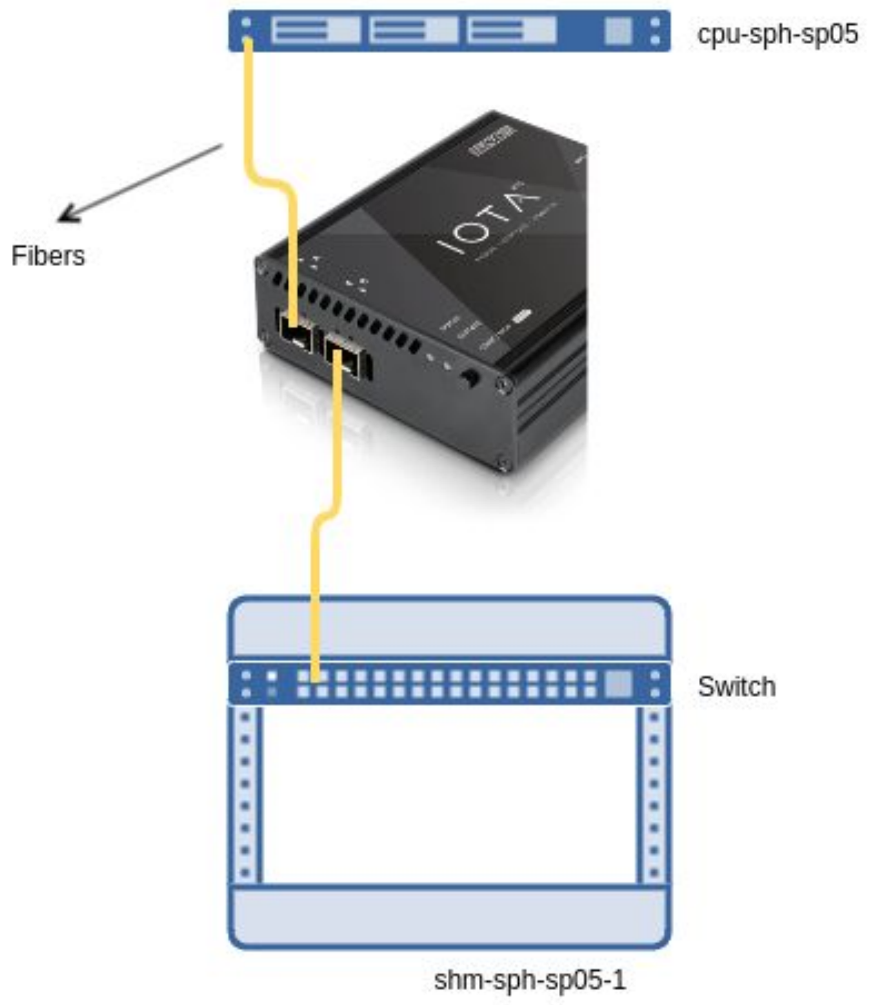
Protocol
UDP
IPv6-ICMP
ICMP

IOTA 10G - Grafana Interface

Accessible through a web browser.

Only requirement is that the IOTA unit is connected to the network and has port 3000 freed.

Tools



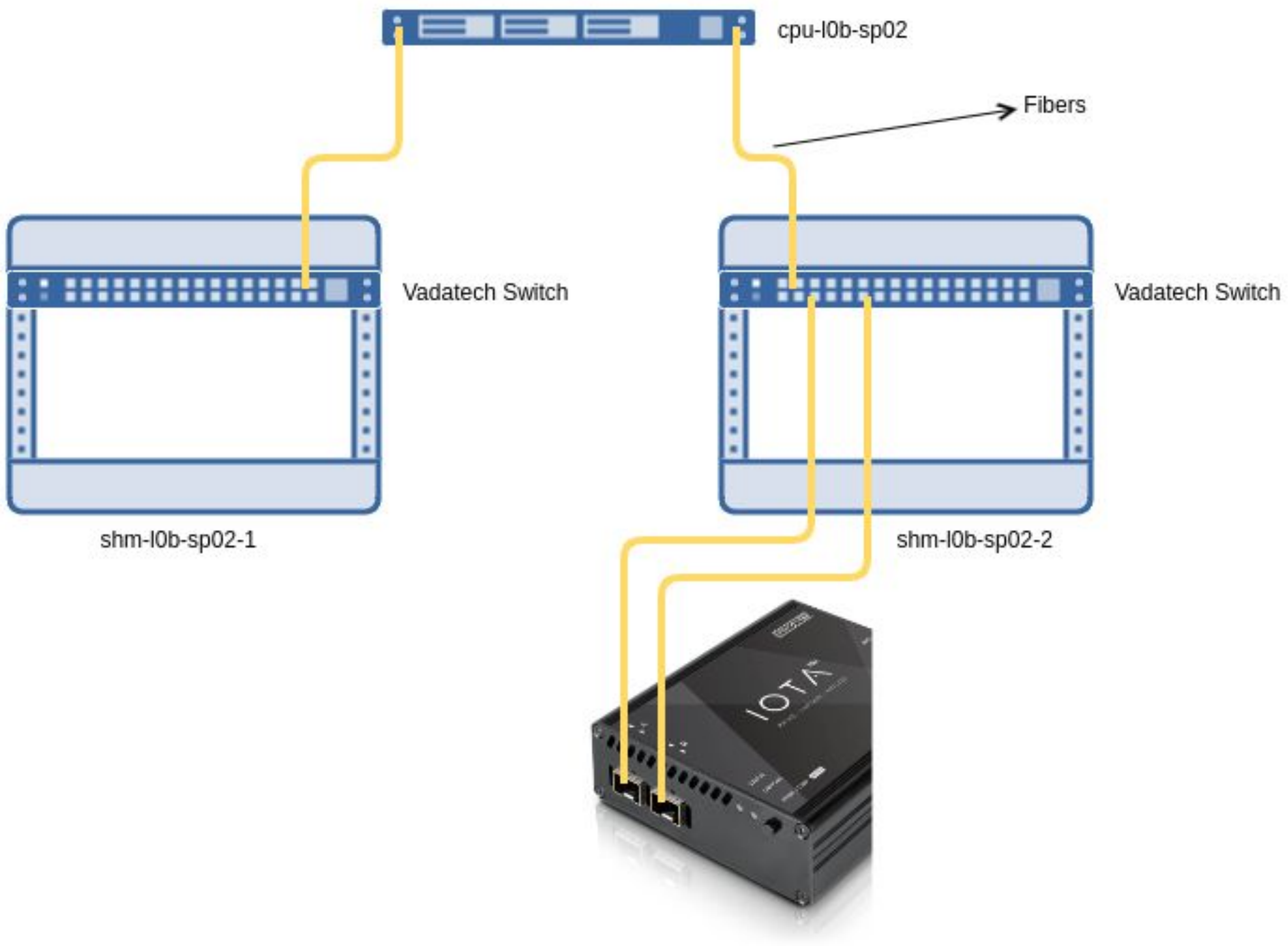
IOTA 10G - Connection style 1

Used with cpu-sph-sp05 and shm-sph-sp05-1.

IOTA keeps in the middle of the communication line between CPU and ATCA and records all transactions.

Captured data can be downloaded and analyzed later.

Tools



IOTA 10G - Connection style 2

Used with cpu-l0b-sp02 and shm-l0b-sp02-2.

All traffic going to the ATCA switch is mirrored to 2 other ports of the switch: inbound and outbound traffic are separated (Vadatech Switch limitation).

IOTA reads the data from the mirrored ports and record them in the SSD.

Non invasive: IOTA can be removed without disturbing the connection between CPU and ATCA.

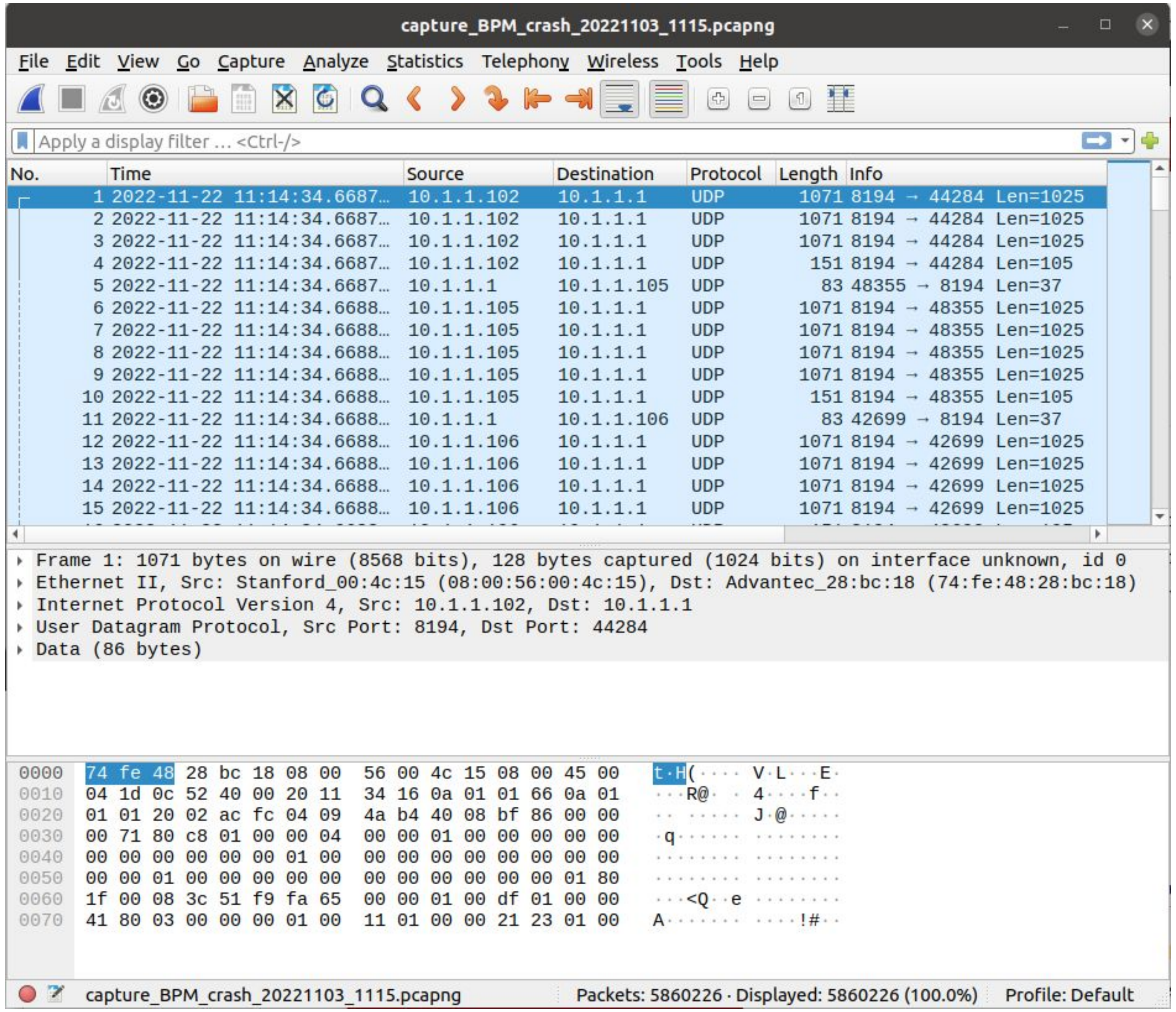
Tools

rssi_network_analyzer.py

Script started by Marcio Donadio and refactored and improved by Ryan Herbst.

- Use the network dump from IOTA.
- Decode for RSSI, SRP, and packetizer SLAC protocols.
- Prompt when something unexpected is found in the communication.
- Print dozens of packets before the problem, and a few after so one can study the scenario when the problem occurred.
    - Printed packets are already decoded to facilitate analysis.
    - The print is added to a text file for later analysis.
- Print statistics of the overall communication for the entire dump file.
- Dump files are typically 1 to 2 GB in size and contains compressed data.
    - The script takes several hours to finish an analysis.

Tools


capture_BPM_crash_20221103_1115.pcapng
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No. Time                          Source      Destination Protocol Length Info
  1 2022-11-22 11:14:34.6687…  10.1.1.102  10.1.1.1    UDP       1071 8194 → 44284 Len=1025
  2 2022-11-22 11:14:34.6687…  10.1.1.102  10.1.1.1    UDP       1071 8194 → 44284 Len=1025
  3 2022-11-22 11:14:34.6687…  10.1.1.102  10.1.1.1    UDP       1071 8194 → 44284 Len=1025
  4 2022-11-22 11:14:34.6687…  10.1.1.102  10.1.1.1    UDP        151 8194 → 44284 Len=105
  5 2022-11-22 11:14:34.6687…  10.1.1.1    10.1.1.105  UDP         83 48355 → 8194 Len=37
  6 2022-11-22 11:14:34.6688…  10.1.1.105  10.1.1.1    UDP       1071 8194 → 48355 Len=1025
  7 2022-11-22 11:14:34.6688…  10.1.1.105  10.1.1.1    UDP       1071 8194 → 48355 Len=1025
  8 2022-11-22 11:14:34.6688…  10.1.1.105  10.1.1.1    UDP       1071 8194 → 48355 Len=1025
  9 2022-11-22 11:14:34.6688…  10.1.1.105  10.1.1.1    UDP       1071 8194 → 48355 Len=1025
 10 2022-11-22 11:14:34.6688…  10.1.1.105  10.1.1.1    UDP        151 8194 → 48355 Len=105
 11 2022-11-22 11:14:34.6688…  10.1.1.1    10.1.1.106  UDP         83 42699 → 8194 Len=37
 12 2022-11-22 11:14:34.6688…  10.1.1.106  10.1.1.1    UDP       1071 8194 → 42699 Len=1025
 13 2022-11-22 11:14:34.6688…  10.1.1.106  10.1.1.1    UDP       1071 8194 → 42699 Len=1025
 14 2022-11-22 11:14:34.6688…  10.1.1.106  10.1.1.1    UDP       1071 8194 → 42699 Len=1025
 15 2022-11-22 11:14:34.6688…  10.1.1.106  10.1.1.1    UDP       1071 8194 → 42699 Len=1025

Frame 1: 1071 bytes on wire (8568 bits), 128 bytes captured (1024 bits) on interface unknown, id 0
Ethernet II, Src: Stanford_00:4c:15 (08:00:56:00:4c:15), Dst: Advantec_28:bc:18 (74:fe:48:28:bc:18)
Internet Protocol Version 4, Src: 10.1.1.102, Dst: 10.1.1.1
User Datagram Protocol, Src Port: 8194, Dst Port: 44284
Data (86 bytes)

0000  74 fe 48 28 bc 18 08 00  56 00 4c 15 08 00 45 00   t·H(···· V·L···E·
0010  04 1d 0c 52 40 00 20 11  34 16 0a 01 01 66 0a 01   ···R@· · 4····f··
0020  01 01 20 02 ac fc 04 09  4a b4 40 08 bf 86 00 00   ·· ····· J·@·····
0030  00 71 80 c8 01 00 00 04  00 00 01 00 00 00 00 00   ·q······ ········
0040  00 00 00 00 00 00 01 00  00 00 00 00 00 00 00 00   ········ ········
0050  00 00 01 00 00 00 00 00  00 00 00 00 00 00 01 80   ········ ········
0060  1f 00 08 3c 51 f9 fa 65  00 00 01 00 df 01 00 00   ···<Q··e ········
0070  41 80 03 00 00 00 01 00  11 01 00 00 21 23 01 00   A······· ····!#··

capture_BPM_crash_20221103_1115.pcapng   Packets: 5860226 · Displayed: 5860226 (100.0%)   Profile: Default
```

Wireshark

The RSSI analyzer script shows data for each individual UDP port.

Wireshark is used to have a more general idea of how the traffic was among all boards and UDP ports.

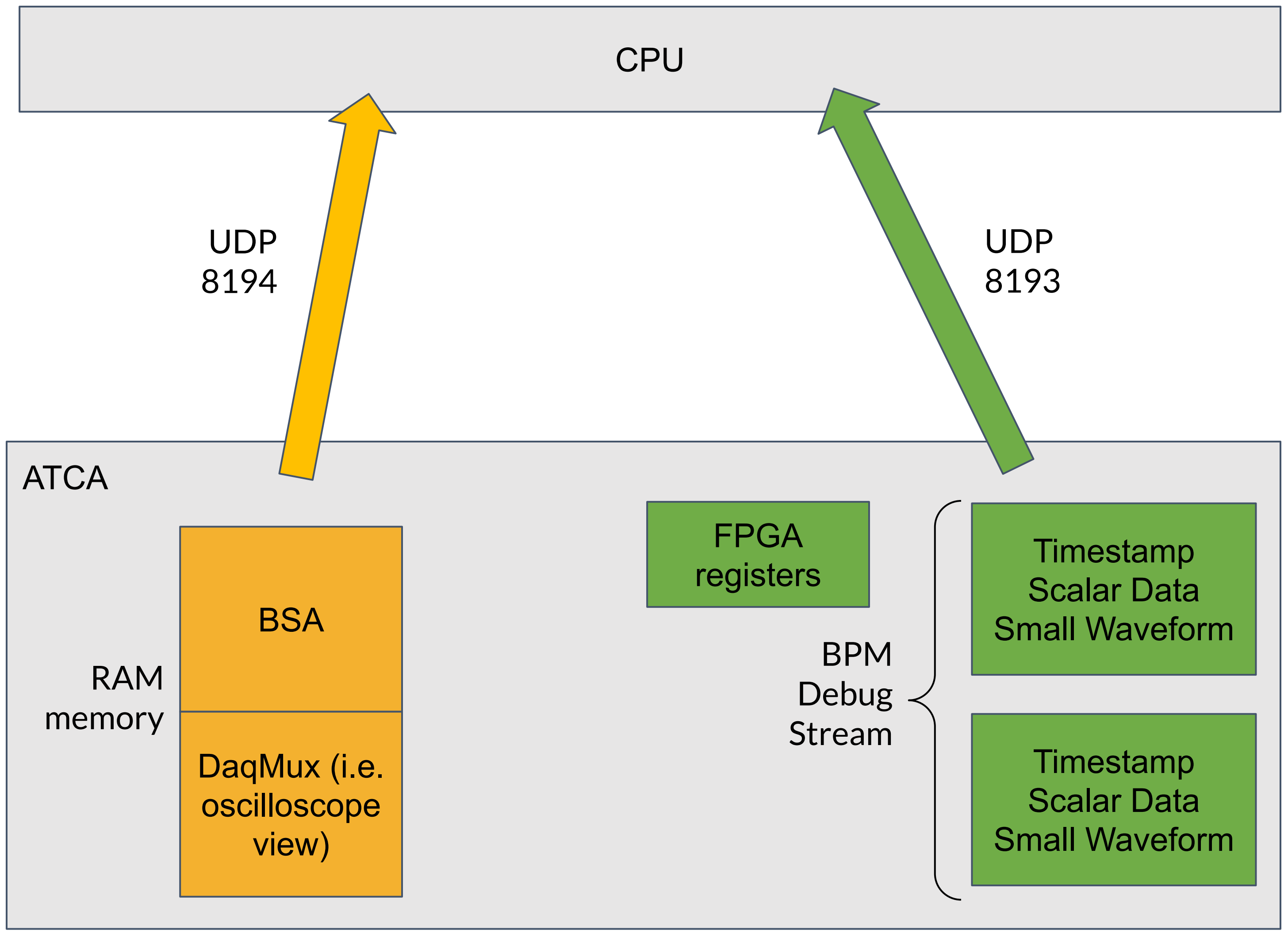
The timestamp of the problem pointed out by the RSSI analyzer is checked in Wireshark to get a broader picture of the situation.

Can easily filter by UDP port and even RSSI or SRP bytes if needed.

3

Details of the Problem

Details of the Problem



Our Main Concern - UDP ports 8193 and 8194

Port 8194 is shared between:

- BSA
- DaqMux

Port 8193 is shared between:

- FPGA register access.
- Debug stream (BPM has one per AMC) containing:
  - Timestamp
  - A set of scalar data
  - Small waveforms (only BPMs and GMDs use debug stream with waveforms)

Debug stream is the way to send time stamped data in an atomic operation.

Superconducting systems usually don’t use the debug stream, but BPMs need it.

Details of the Problem

Trigger of the problem

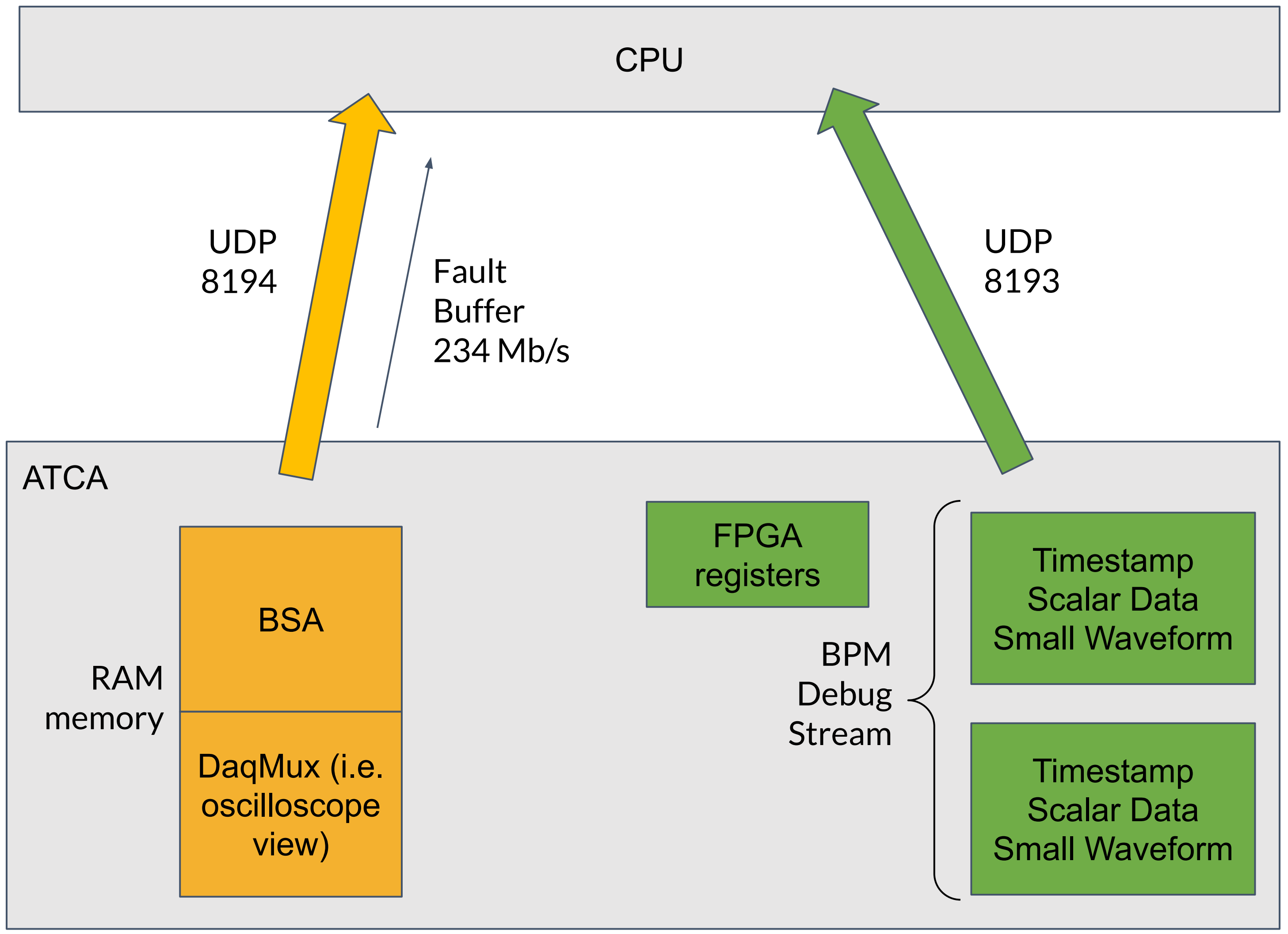
Transfer of fault buffer data is what triggers the problem 100% of the time.

- Mapping of the trigger was the result of a work during a Friday night with Mike Zelazny, Marcio Donadio, Ernest Williams and Kukhee Kim.
- A Matlab process cleans one fault buffer of the systems every 15 minutes.
    - The number of fault conditions happen so frequently that after a few seconds the buffer is filled again.
- The data from the RAM memory is transferred to the software once the buffer is filled.
    - 440 MB per carrier board in 15 to 20 seconds (176 Mb/s to 234 Mb/s per board).
- Eventually one IOC will crash once the fault buffer transfer starts.
    - 100% of the crashes happened during this moment.

During the week 12/12/2022 to 12/16/2022 there was no beam and I could force faults manually.

- During the hours I was not testing, IOCs were always stable.
- When I forced a fault, eventually one IOC would crash.
- I saw no crashes out of a moment of fault buffer data transfer.

Details of the Problem



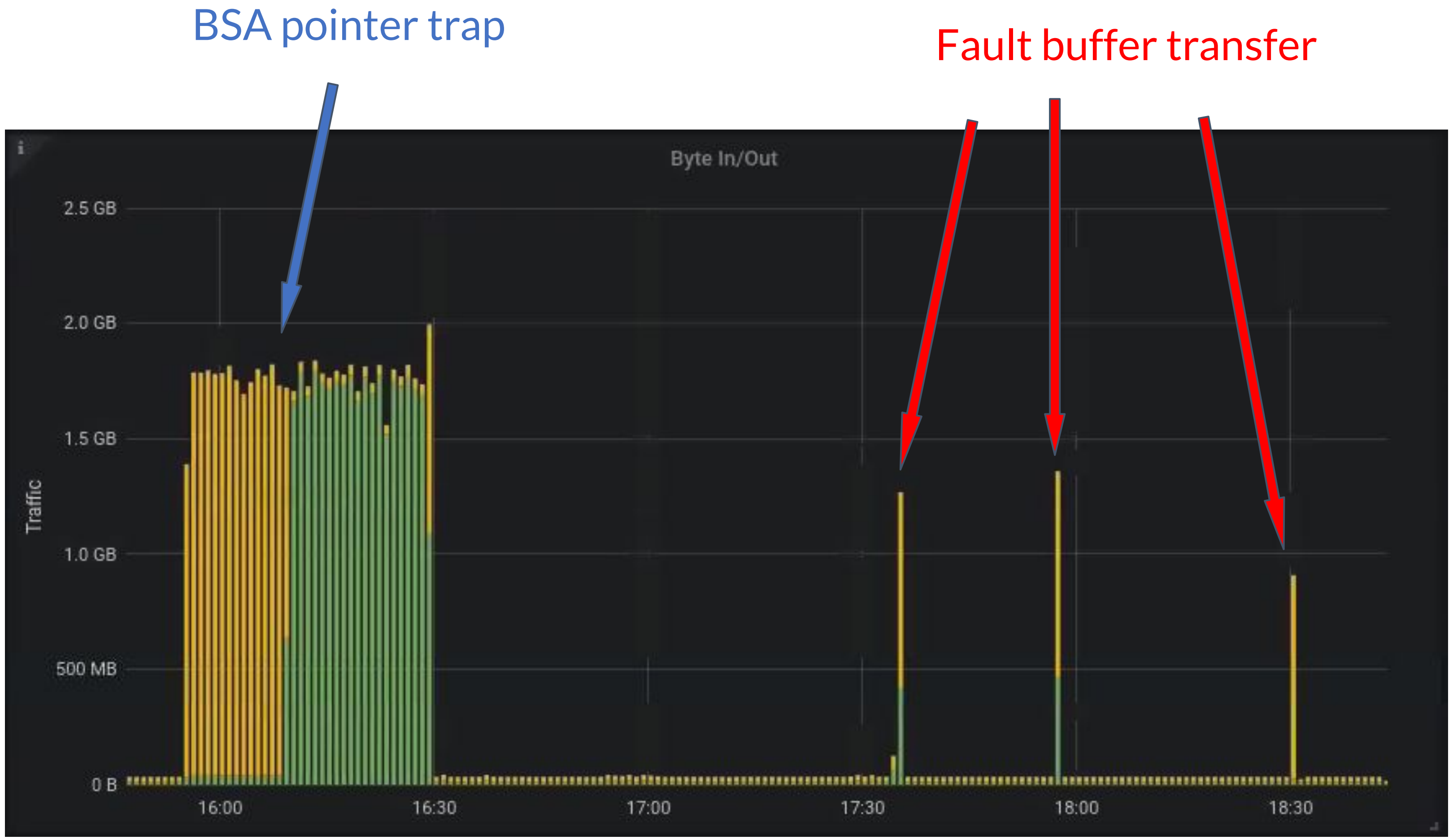
CPU
UDP 8194
Fault Buffer 234 Mb/s
UDP 8193
ATCA
RAM memory
BSA
DaqMux (i.e. oscilloscope view)
FPGA registers
BPM Debug Stream
Timestamp Scalar Data Small Waveform
Timestamp Scalar Data Small Waveform

UDP port 8194

We can configure a timeout and retries on software so it can overcome the raise in data to process.

We raised the timeout and retries for port 8194 and the problem improved, but was not solved.

Details of the Problem



Traffic data (bytes) x time (hour and minute of occurrence)

BSA pointer trap effect

Together with the fault buffer transfer event, randomly a BSA pointer trap event will happen (details on this is not in the scope of this presentation).

This is more frequent when real fault conditions occur, but I could trigger it once when creating manual faults.

The software keeps asking BSA data from the firmware, apparently at a 6.5 kHz frequency.

The software only stops requesting data when a new fault condition happens. On the image in the left, I left it running for 34 minutes before forcing a new fault condition. I believe that the condition will never stop by itself.

I couldn't correlate an increase in crashes with the occurrence of a BSA pointer trap.

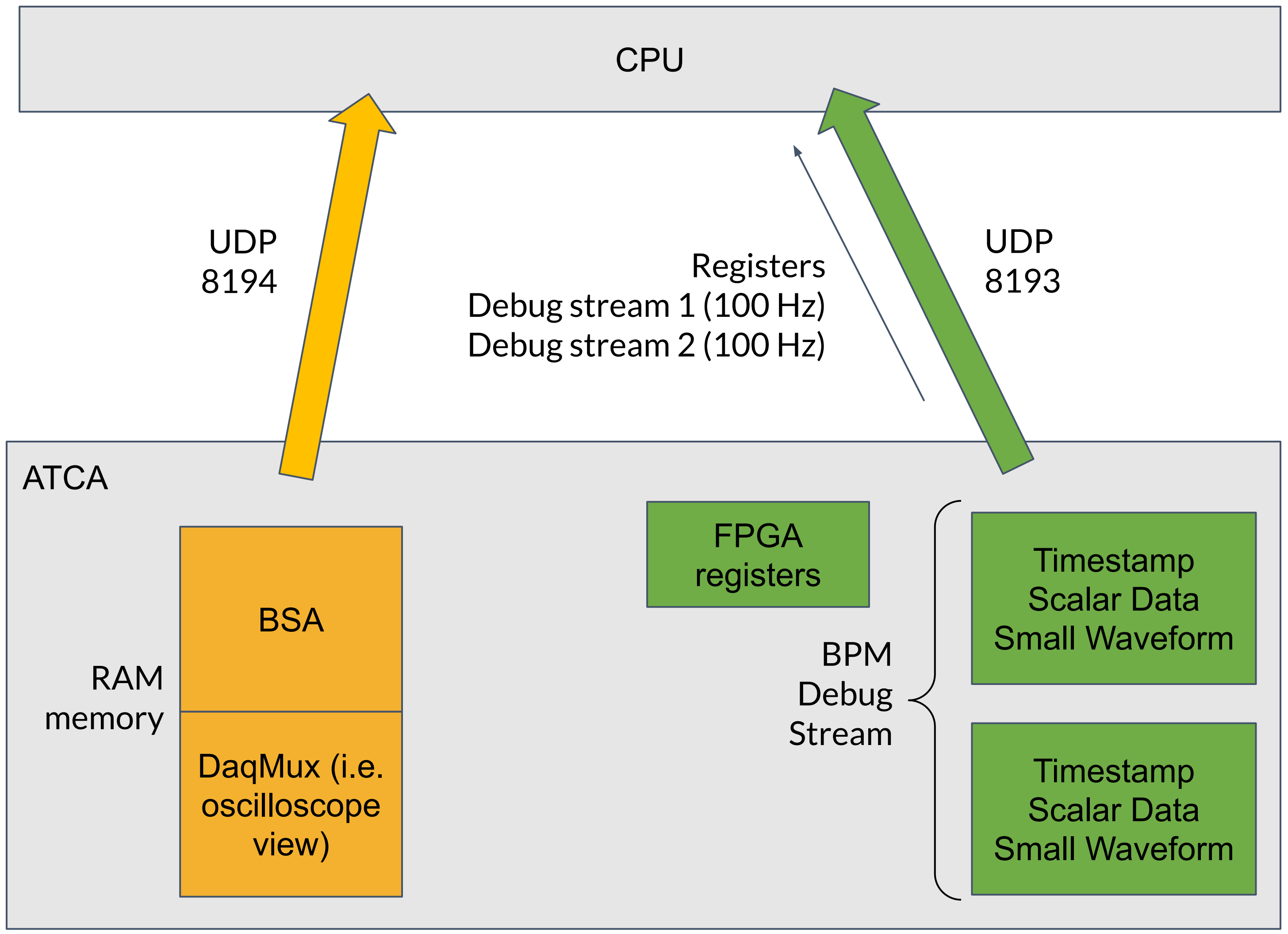
Details of the Problem

Why looking at port 8193?

Error message on IOC console when fault buffers are being transferred in port 8194.

- CPSW Error: CPSW Error: CPSW Error: No response -- timeout (Retries=0, Last timeout=500000): Success: **/mmio/AmcCarrierCore/AmcCarrierBsa/Bsss/currPacketSize**: Success at ../BsssYaml.cc, line 69 terminate called after throwing an instance of 'CPSWError' what(): CPSW Error: CPSW Error: No response -- timeout (Retries=0, Last timeout=500000): Success: **/mmio/AmcCarrierCore/AmcCarrierBsa/Bsss/currPacketSize**: Success
- The error message shows a timeout when accessing FPGA registers. FPGA register access is in port 8193, not 8194.
- Analysis of network dumps showed that the software badly struggles when accessing data from port 8193.
    - Software sometimes signals the firmware that it is busy.
    - Software acknowledges the firmware the receipt of old packets, but not new packets.
    - One time we could see that the software sent the firmware a close-connection packet. This happens only when the peer receives no message for 3 seconds.
- Analysis of what is happening in port 8194 during the fault buffer transferring was surprising:
    - Software handles the communication in port 8194 comfortably.

Details of the Problem



UDP port 8193

In the case of BPM, FPGA register data and debug streams are sent to the CPU.

The firmware sends debug streams based on 2 triggers configurations:

1. User defined frequency (currently 10 Hz or less).
2. Calibration signal from the TPR (currently 100 Hz).

The thread taking care of communication in port 8193 may not be able to handle 100 Hz communication while the fault buffers are being transferred.

Details of the Problem

Correlation of a 100 Hz debug stream transfer and rate of crashes

With two BPM systems available for testing, I could create two testing scenarios.

- sioc-sph-bp06:
    - 2 AMC cards in the carrier board.
    - 2 debug streams.
    - Manually configured with 10 Hz trigger for debug stream and disabled calibration trigger.
- sioc-sph-bp07:
    - 1 AMC card in the carrier board.
    - 1 debug stream.
    - Manually configured with 10 Hz trigger for debug stream and 100 Hz for calibration trigger. This mimics the current operation in production.
- FPGA register transaction timeout of 0.5 seconds (UDP port 8193).

Results:

- sioc-sph-bp06 never crashes.
- sioc-sph-bp07 crashes once for every 5 fault conditions approximately.

4

Conclusions, recommendations, and next steps

Conclusions, recommendations, and next steps

Conclusions and recommendations

Correlation of fault buffer transfer with crashes was proved.

- The software could reduce the bandwidth transfer of a fault buffer by limiting the frequency when it requests a new BSA ram memory region.
- Fixing the BSA pointer trap may improve the crashes, although I couldn't correlate it with an increase in crash rates.

Correlation of debug stream frequency and rate of crashes was proved.

- The firmware could limit the frequency of debug stream delivery to 10 Hz.
  - Currently, according to the documentation, the limitation is in the order of hundreds of Hertz.

Software modules need to implement exception handling.

- In most cases, reading a register and failing to do this should not be reason enough to have an IOC crashing.
- Although this could improve the situation by far, we observed one moment when the software closed the connection. In this case, the error handling would not have helped.
- Need a care thinking about actions to take for a small set of registers that are key for BSA transaction in case a register read fails.

Conclusions, recommendations, and next steps

Conclusions and recommendations

Increasing the timeout for register reading reduced the rate of crashes.

- I could verify this with the manual fault during one week of tests.
- Sonya Hoobler attested that changing the timeout to 1 second (from 100 ms) improved the crashes. Ryan Herbst mentioned that for applications using Rogue, they concluded that 1 second was an optimal number to use.
- One approach could be to increase this number every time a crash happens until we have an optimal number.
- Ryan Herbst mentioned that it is better not to configure retries on software when reading or writing to registers.

The contribution of DaqMux to the crashes was not assessed:

- BPM doesn't use DaqMux.
- BLEN, BCM, wire scanner, and GMD use DaqMux.

I checked CPU and memory usage during fault buffer transfer and BSA pointer trap conditions and nothing abnormal was found:

- CPU usage raised from 5% to 15%.
- Memory usage increased in 2%.

Conclusions, recommendations, and next steps

Next steps

Understand how a raise in traffic in port 8194 causes a timeout when software is communicating with port 8193.

- This needs a deep dive in the CPSW source code.
- We may have:
  - A sub-optimal architecture for the use cases we are dealing with.
  - A hidden bug.
  - A sub-optimal configuration of thread priorities in the real-time scheduler.
  - A hidden problem in the Linux kernel device driver responsible for the network (although for this case I would expect problems when accessing other ports, too)

Implement exception handling for our modules:

- ATCACommon
- tprTrigger
- bsaDriver

Conclusions, recommendations, and next steps

Next steps

If possible and if the team agrees:

- Implement bandwidth limitation of the BSA fault buffer in software.
- Implement rate limitation of the debug stream in the BPM firmware application.

Fix the BSA pointer trap error.

Continue to increase the SRP timeout value until crashes stop.

Test how much DaqMux contributes to IOC crashes.