Packet Trace Modelling and Visualization

Petru Lauric

OCT.2014

freescale™

External Use

Introduction

- Trace-based performance analysis and debug tools designed specifically for network processors
- Networking-oriented system-level analysis
- Packet-focused performance analysis
- The speaker is a software developer and a co-author of the Freescale Packet Analysis Tool

Agenda

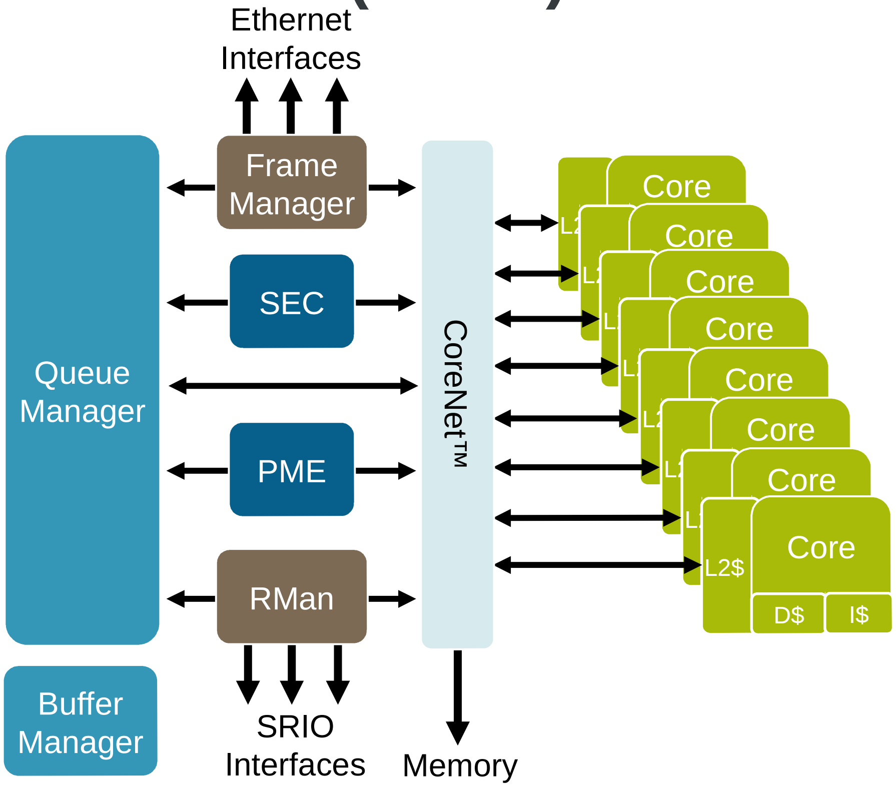
Main topics:

- Network processor analysis
- Hardware packet trace
- Linux environment software trace considerations
- Modelling the networking functionality
- Analysis data visualization
- Short demo

Network Processors

- Devices designed specifically for networking applications
  - Typically one or more general purpose processing cores (GPPs)
  - Specialized hardware mechanisms for offloading network related tasks

Freescale DataPath Acceleration Architecture (DPAA)



Why Packet-Oriented Analysis?

- Traditional analysis tools:
  - Software-centric
  - Hardware-centric
  - Networking feature agnostic
- Networking-specific analysis tools
  - Traffic-centric: Wireshark, iperf etc.
  - External observations only
- New paradigm
  - Packet-centric analysis of the internals of the system under test
  - Support integrating with other analysis tools and technologies

Packet Analysis Use Cases

Packet Tracing	Shows which parts of the system process the packets. For example, use this to verify that the packet flow is what you expect.
Packet Loss Analysis	Understand why the packets become “lost” in the system: intentionally discarded due to QoS constraints, discarded due to congestion, misrouted etc.
Networking Metric Measurement	Measure the packet throughput, jitter, error rates etc. at various points in the system.
Latency Analysis	Measure the time spent processing packets at various points in the system.
Load Balancing Analysis	Measure how the packet processing is distributed across the system (e.g. among cores).

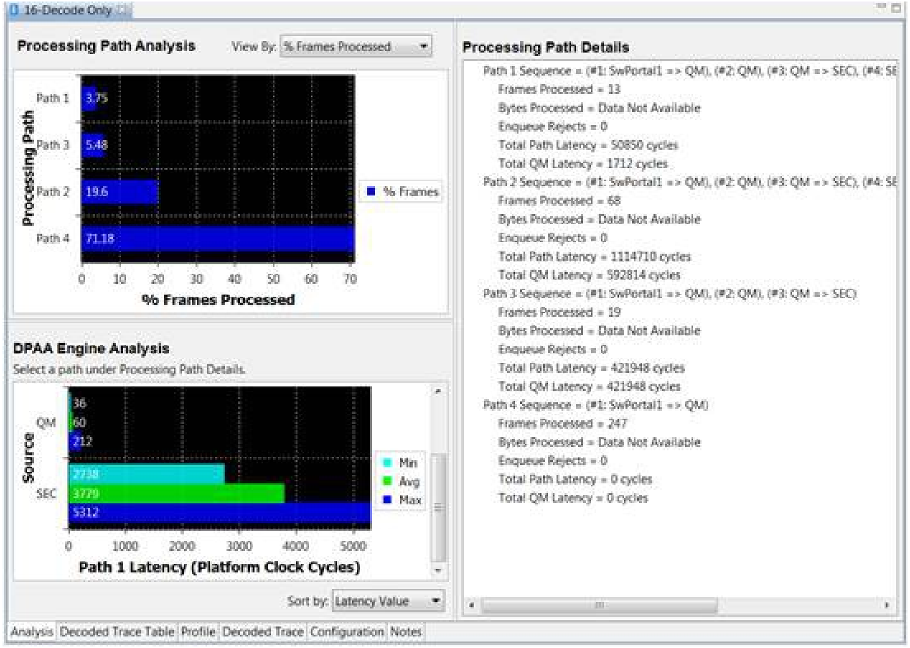
Network Processor Analysis Tools

Trace analysis tool considerations

- Software trace is insufficient when hardware offloading is used
- The system's parallelism requires:
  - Collecting trace data from multiple sources
  - Making meaningful trace data correlations
  - Using effective visualization techniques
- Trace setting configuration complexity
- Quantitative and qualitative trace data collection control

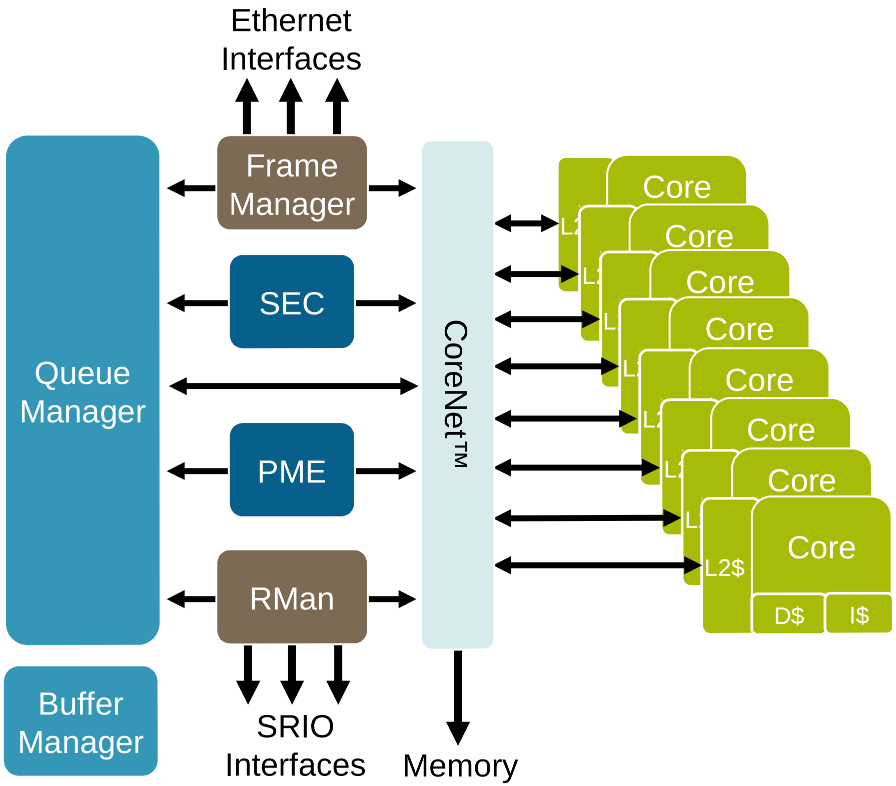
Freescale Packet Analysis Tool – Overview

- User friendly system-level analysis, performance measurements and debug
- Provides visibility into networking tasks offloaded to hardware
- Non-intrusive or low-intrusive data collection
- Packet-centric analysis data visualization
- Exemplifies the key concepts for implementing packet-focused analysis tools



Packet Analysis Tool – Freescale DPAA Overview

- *The tool only supports the Freescale network processors which implement the DataPath Acceleration Architecture (DPAA)*
- *For details, search freescale.com for "QORIQDPAAWP"*



- **Acceleration of frame/packet processing**
- **Classification and distribution of data flows among cores and software partitions**
- **Abstract and manage efficiently inter-core communications and the access to shared resources**

Packet Analysis Tool – DPAA Hardware Trace

- Visibility into Frame Manager and Queue Manager activities via IEEE-ISTO 5001 Nexus trace
- The tool selectively enables the output of trace and the verbosity level for each trace point
- For **each** traced packet, the trace data contains info such as:
  - The frame’s (packet buffer’s) address
  - Error code(s) if applicable
  - Trace point id
  - Timestamps from a hardware clock
- The packet headers and payload are **not** traced

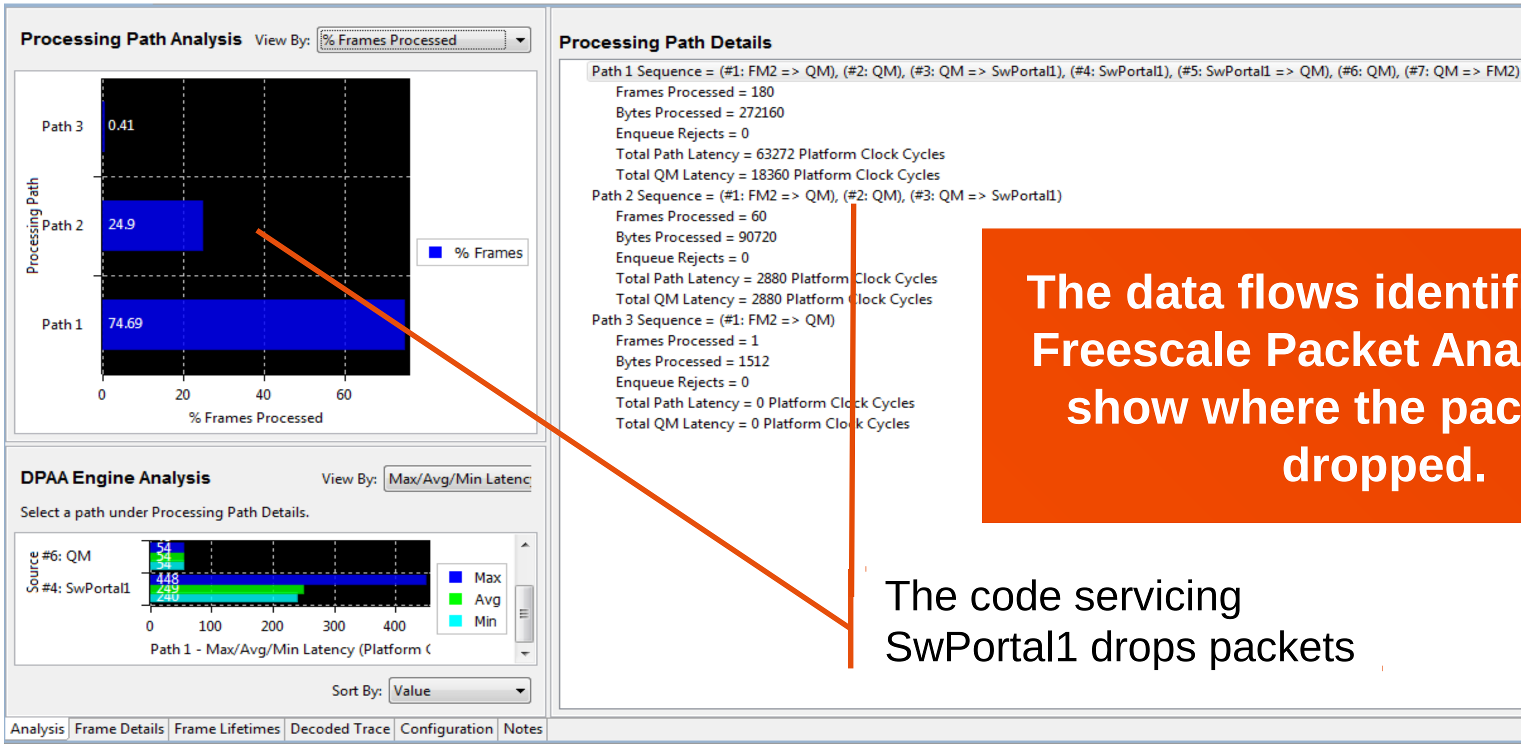
Packet Loss Analysis Example

iperf output

[…]

[ID] Interval	Transfer	Bandwidth	Jitter	Lost/Total Datagrams
[4] 75.0-76.0 sec	8.55 MBytes	71.8 Mbits/sec	0.008 ms	2034/ 8136 (25%)

iperf reported dropped packets



The data flows identified by the Freescale Packet Analysis Tool show where the packets are dropped.

The code servicing SwPortal1 drops packets

Trace Analysis Tool Considerations

Packet Analysis Tool – DPAA Hardware Trace

Trace analysis tool considerations:

Software trace is insufficient when hardware offloading is used

[…]

- The DPAA hardware trace is collected from key points in the networking hardware
  - The central data exchange mechanism (QM - Queue Manager)
  - The “network interface controller” (FM - Frame Manager)
- The packets can be monitored while being processed by the hardware

Packet Analysis Tool – Packet Trace Sources

The system's parallelism requires:

- Collecting trace data from multiple sources
- Making meaningful trace data correlations
- Using effective visualization techniques

- The hardware trace data shows how and when the various subsystems (cores, hw accelerators, network interfaces) interact
- The software trace (future tool extensions) can be collected from Linux (e.g. using LTTng) to show how the packets were processed on the cores

Packet Analysis Tool – Packet Trace Correlation

- Trace data correlation types
  - Among subsystems – same packet observed at multiple points
  - Among “related packets” (data flow)
  - Among trace data sets from multiple sources
- Packet Analysis Tool correlation
  - Each DPAA hardware trace message - one packet/frame
  - The trace messages contain the frame address (packet buffer address), which is used as a unique identifier
  - The hardware trace data and (in the future) software trace can be correlated for “packet lifetime tracing”
  - Automatic analysis of the trace data discovers the “processing paths”

Packet Analysis Tool – Flow Level Analysis

Processing Path

- Entity representing a data flow
- Used to group stats for “related” frames (similar lifetime)
- Automatically identified by the Packet Analysis Tool by analyzing the hardware trace data, no software instrumentation requirements
- Frames are tracked based on the address from trace

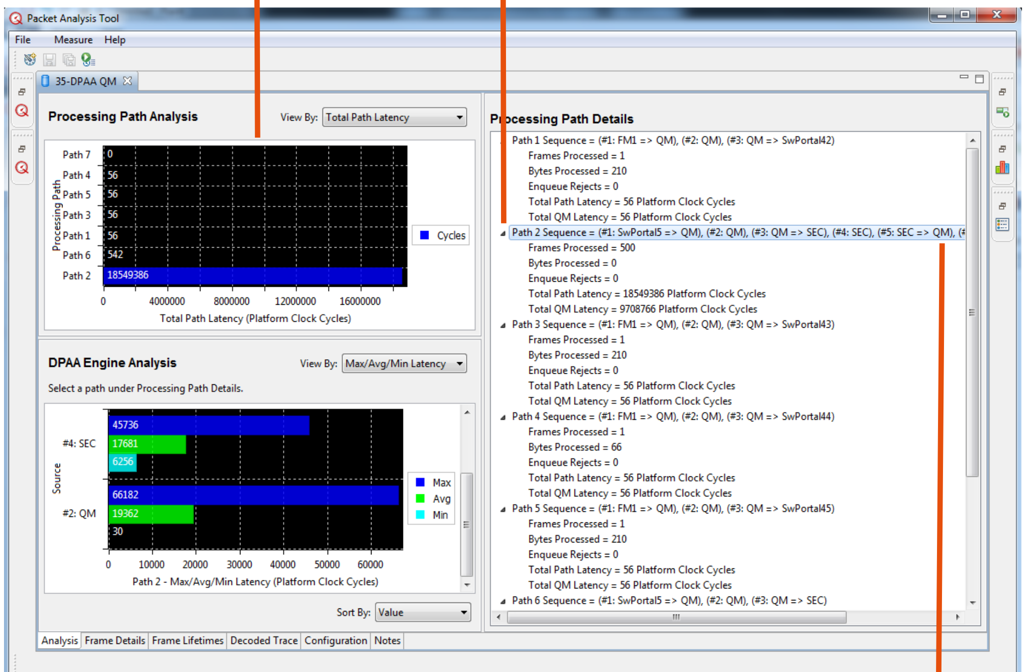
Packet Analysis Tool Visualization

Packet Analysis Tool Visualization – Processing Paths

Data Flows From Trace Data

Flow Level Latency Statistics

- Sum of total time spent on this path
- Total number of frames processed on this path

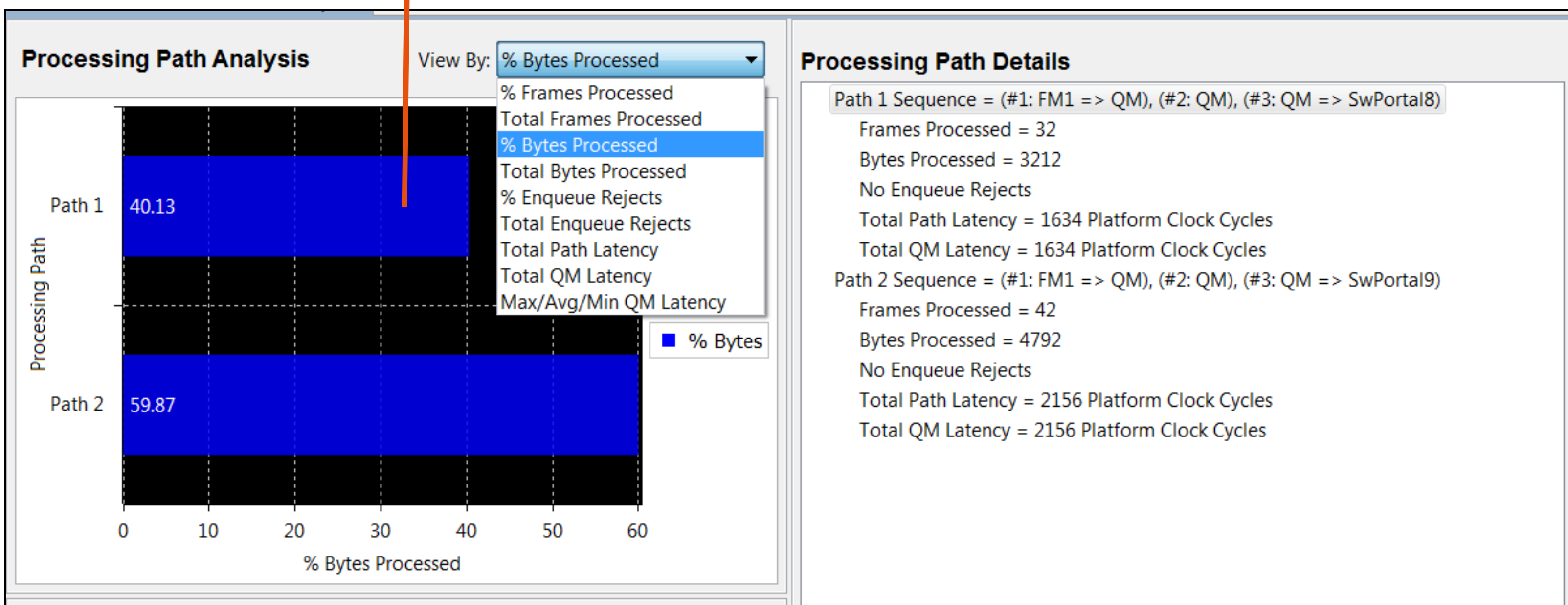


Frame Processing Stages

Path 2 Sequence = (#1: SwPortal5 => QM), (#2: QM), (#3: QM => SEC), (#4: SEC), (#5: SEC => QM), (#6: QM), (#7: QM => SwPortal5)

Packet Analysis Tool Visualization – Processing Path Compare

Compare path stats to determine "weight" of each data flow and to evaluate the system's **load balancing**

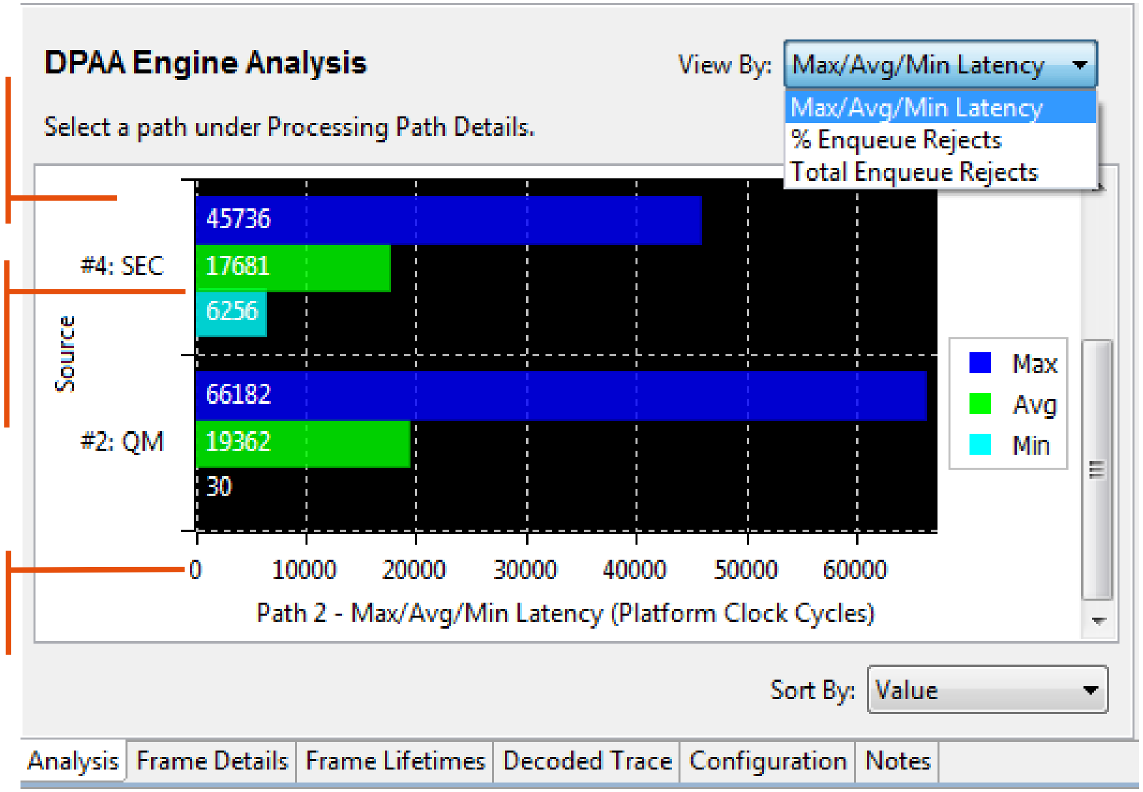


Packet Analysis Tool Visualization – DPAA Subsystem Analysis

Compare DPAA engine (subsystem) level stats to search for bottlenecks

Observe frame processing latency on SEC engine and the QM stage before SEC processing

Updated for the currently selected processing path



Packet Analysis Tool Visualization – Frame Details View

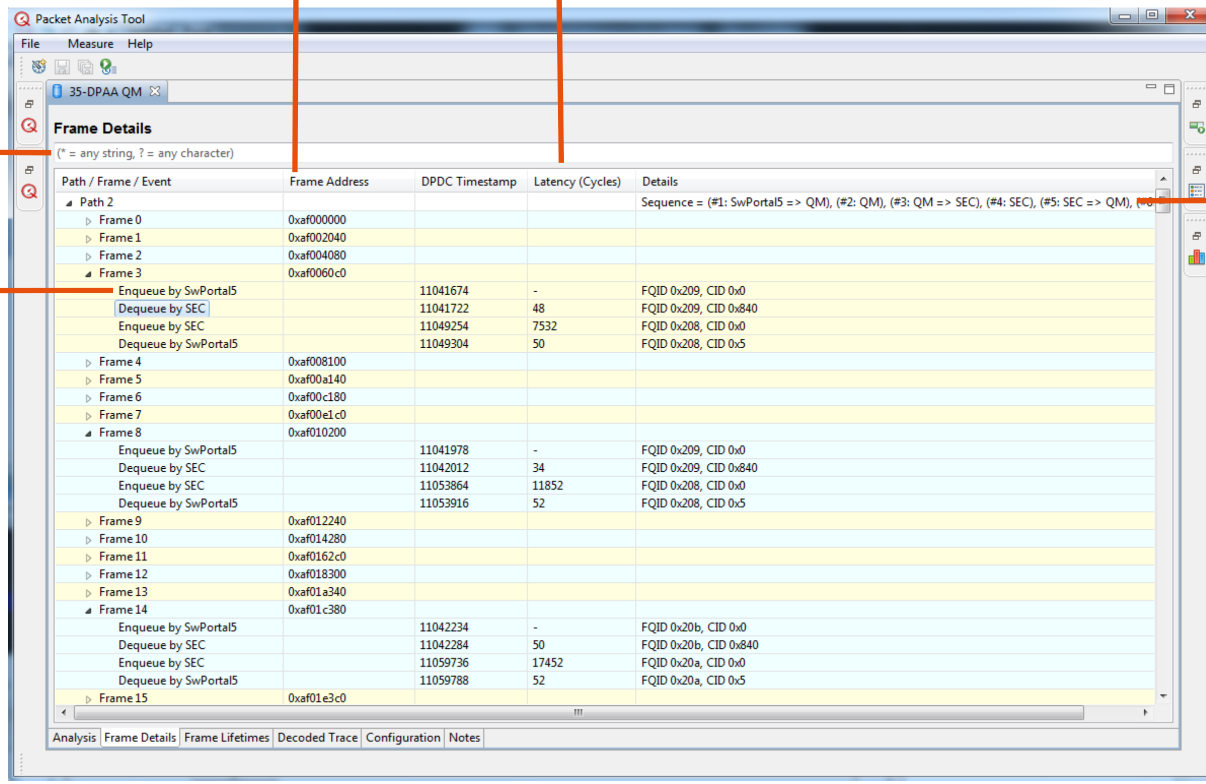
Frame Address

Frame Processing Latencies
(e.g. observe the increasing SEC latency)

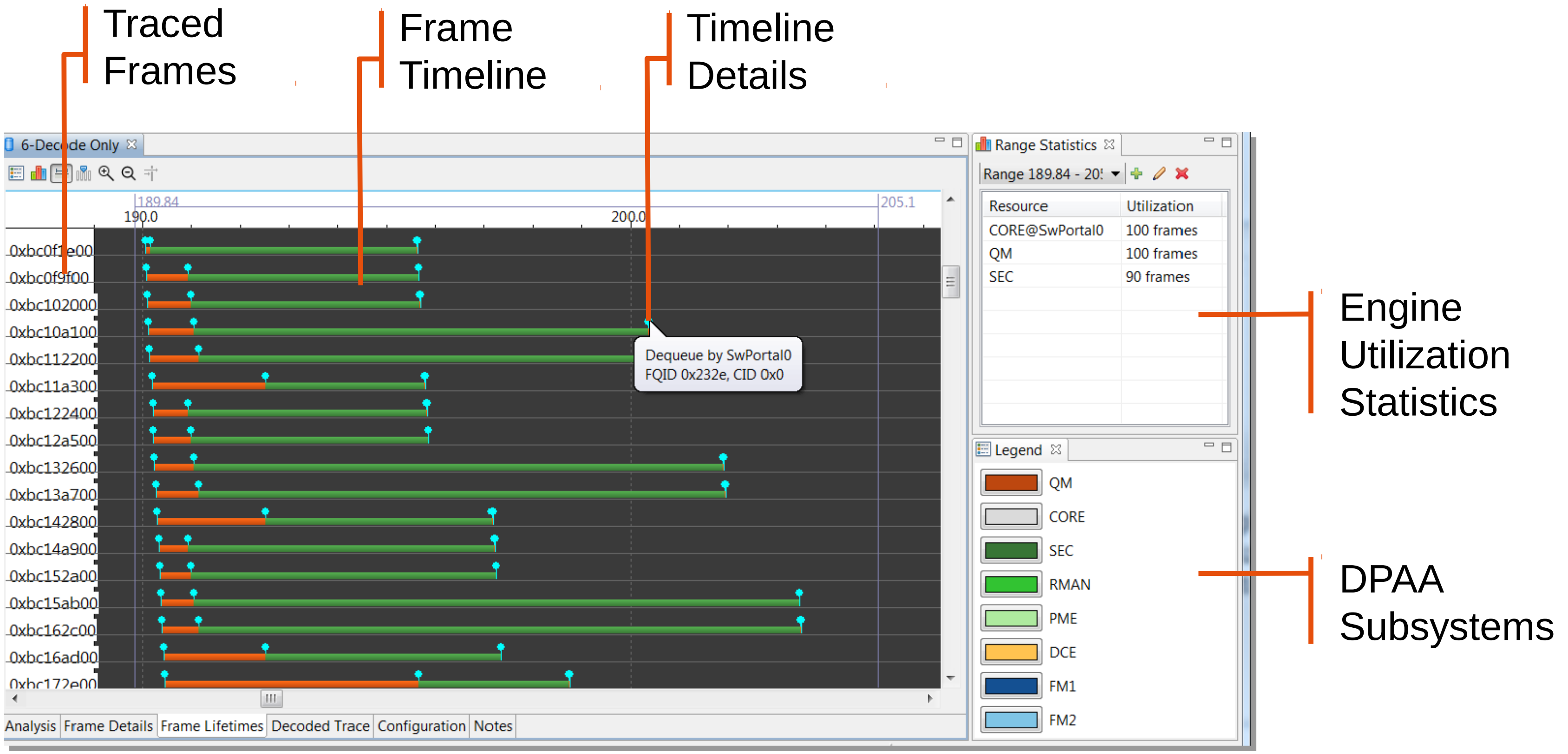
View Filter

Frame Processing Details

Frame Processing Path



Packet Analysis Tool Visualization – Frame Lifetimes View



Packet Analysis Based on Software Trace

Packet Analysis Using Software Trace

- The networking software trace can provide data similar to the hardware trace:
  - The frame’s (packet buffer’s) address
  - Error code(s) if applicable
  - Trace point id
  - Timestamp
- Software only trace
  - LTTng, Ftrace, plain log files
  - Can be intrusive
- Hardware assisted software trace
  - Use the hardware trace features of the cores, if available: Freescale Nexus or ARM CoreSight
  - Typically non-intrusive or very low intrusiveness
  - Highly accurate timestamps

Packet Analysis Software Trace Correlation

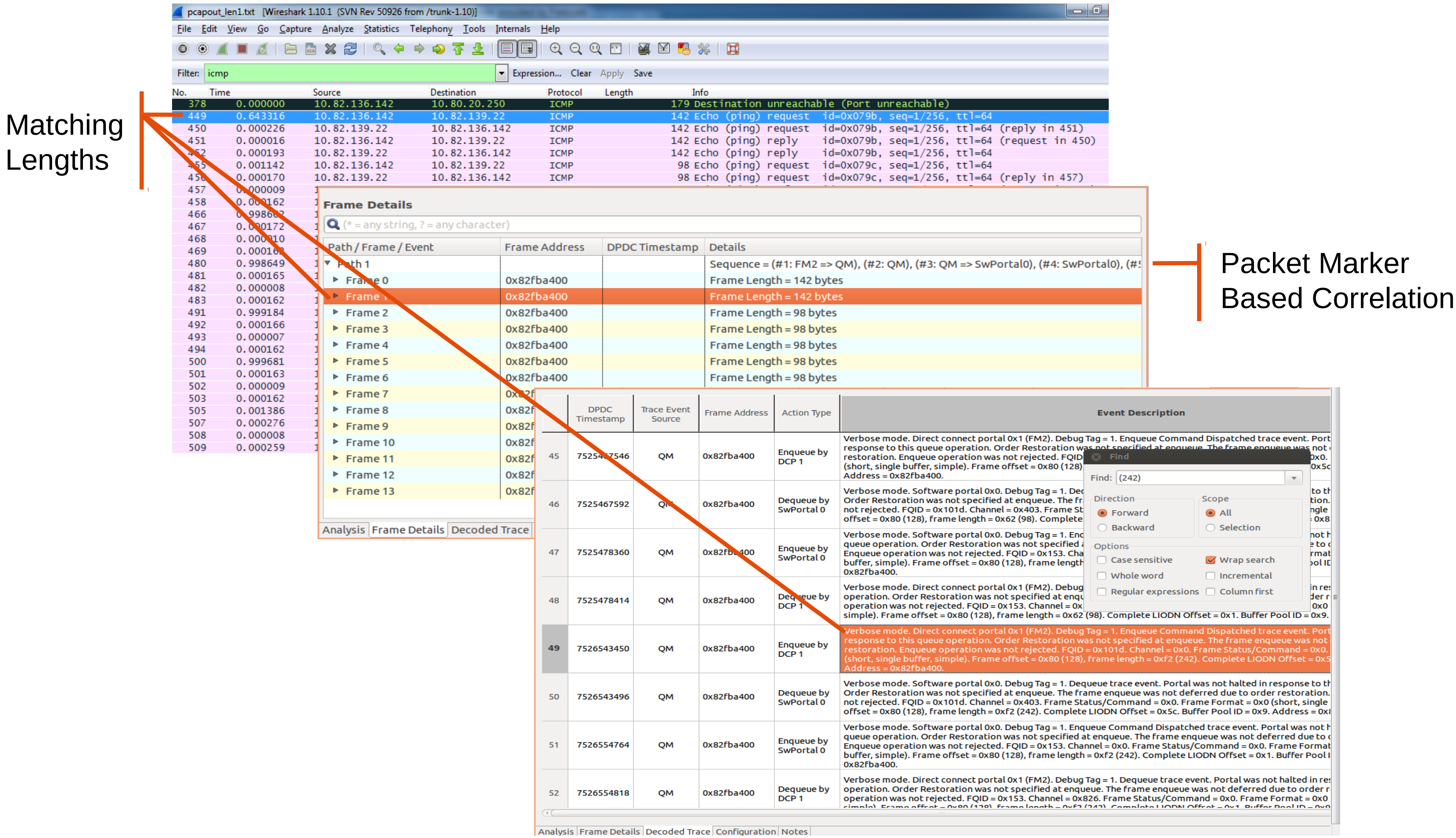
Trace data correlation types
- Among subsystems – same packet observed at multiple points
- Among "related packets" (data flow)
- Among trace data sets from multiple sources

- Between subsystems:
  - Use the packet address
- Data flow identification
  - Similar to the hw trace
- Correlating trace data sets from different sources
  - Typically based on timestamps
  - Timestamp correlation and normalization may be required
  - Other correlation methods – use trace data annotations (markers) – see the next slides

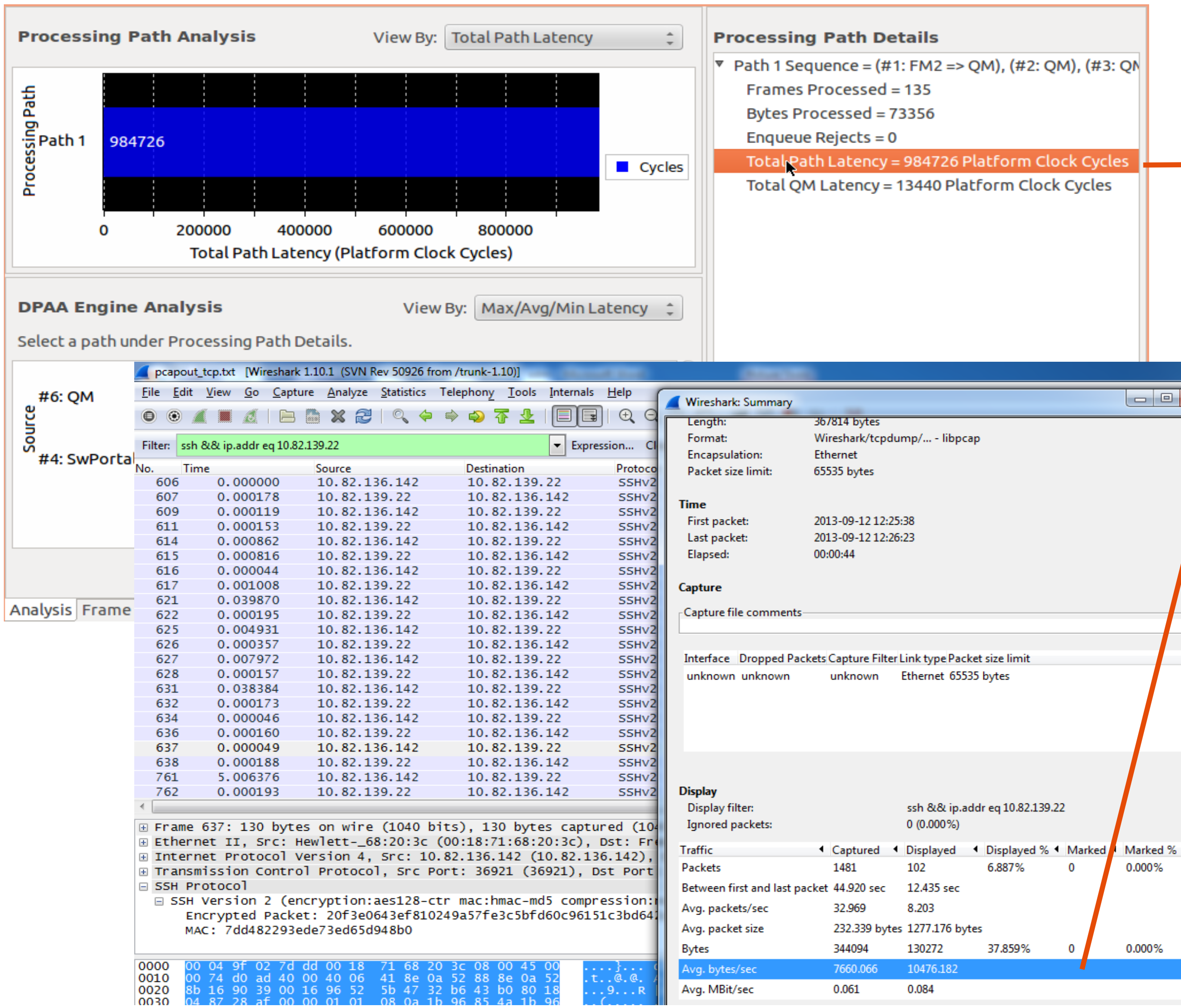
Packet Analysis Tool – WireShark Correlation

- Two data sets need to be correlated: the log of network packets (analyzed by Wireshark) and the hardware trace (analyzed by the Packet Analysis Tool)
- One easy correlation method: use ICMP packet **markers**
- At the beginning of the test, inject ICMP packet with size X
- At the end of the test, inject ICMP packet with size Y

Packet Analysis Tool – WireShark Correlated Analysis



Packet Analysis Tool – WireShark Correlated Analysis (continued)



After Correlation, compare Hardware stats with Network Traffic stats.

What’s Next

Extending Packet Analysis to Software

- Software visibility
  - LTTng
  - Dynamic tracing
- Hardware assisted software tracing
  - See the Linaro ARM Coresight framework
- Focus on the networking software/hardware interfacing

Trace Data Correlation and Visualization

- The packet trace data and logs of different types and/or from multiple sources - represented and stored using the Common Trace Format (CTF)
- The Trace Compass Eclipse project - generic framework for multi-set trace data collection, analysis and visualization
  - CTF trace data sets
  - libpcap packet capture logs
  - LTTng kernel and userspace logs
- Trace Compass supports data driven visualization and analysis
- Considered for future Freescale packet analysis tools

Conclusion

- Packet centric trace analysis – focused on the specifics of the networking devices and applications
- The packet trace provides unique insight
  - The tasks offloaded to hardware can be easily monitored
  - Key networking metrics (latency, packet loss rates, throughput) can be easily measured and reported at software module and hardware subsystem level
- The packet analysis data is well suited for high level system modelling and visualization

For more info, search freescale.com for “packet analysis”.

Backup

Packet Analysis Tool – Trace Data Collection Control

- Trace data collection constraints
  - The network traffic volume is very high (e.g. Freescale T4240 has multiple 1Gbps and 10Gbps network interfaces)
  - The trace bandwidth is limited
  - The trace storage is limited

- Traced Frames
  - Only the frames marked for debug are traced
  - The frames can be automatically marked for debug when packets are received from the network
  - The packets can also be marked by the instrumented software running on the cores

- The tool configures
  - Which packets get marked for debug
  - Which key points in the system output trace and for which debug mark
  - The verbosity level for each trace point

www.Freescale.com