napatech

Wireshark in a Multi-Core Environment Using Hardware Acceleration

Presenter: Pete Sanders, Napatech Inc.

Sharkfest 2009 – Stanford University

napatech

Presentation Overview

- About Napatech
- Why Network Acceleration Adapters are Needed
- Line Speed Capturing
- Filtering Unwanted Traffic
- Payload Removal (snaplen, Slicing)
- Multi-CPU buffer splitting (load balancing)
- Discarding Duplicate Frames (Deduplication)
- Time Stamping
- Transmit
- Napatech LibPCAP Library
- Demonstration

About Napatech

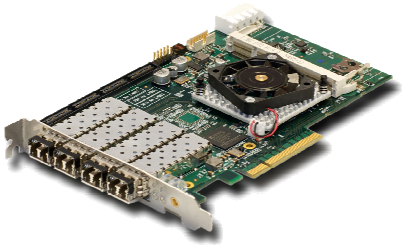
- Napatech is a leading OEM supplier of the highest performing 1 & 10 Gb/s Hardware Acceleration Network Adaptors
- Application offloading through hardware acceleration:
  - A flexible Feature-Upgradable FPGA technology
  - A scalable migration path from 1 Gb/s to 10 Gb/s networks, and beyond
  - A Uniform platform API that is easy to integrate and maintain
  - Industry standard LibPCAP support

Denmark
Copenhagen
40 Employees
HQ, R&D and **Admin**

USA East Coast
Boston, MA
6 Employees
Sales & Support

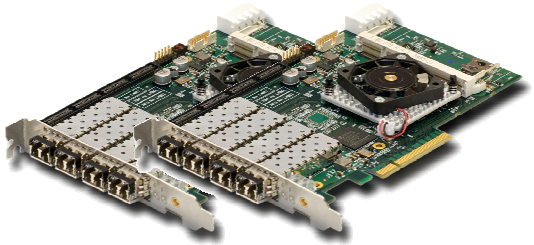
USA West Coast
Mountain View, CA
6 Employees
Sales & Support

Napatech Adapter Portfolio – PCIe



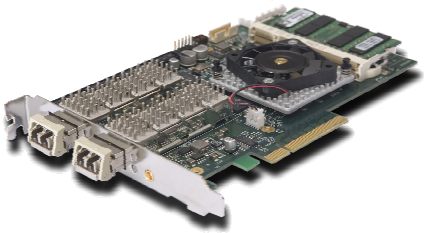
NT4E-STD Adapter

- 4x1Gb Ethernet interface SFP or RJ45
- ½ Length PCI-E
- 4 Gbps lines speed capture
- Time stamping
- Host OS time sync
- Host-based retransmit
- CPU utilization: <1%
- Linux, FreeBSD and Windows drivers
- 3 different product variants available



NT4E + NTPORT4

- 4x1Gb Ethernet interface SFP or RJ45
- ½ Length PCI-E
- External time sync connector
- 8Gbps packet processing, filtering, tagging, timestamp, slicing, local retransmit
- CPU utilization: <1%
- Linux, FreeBSD and Windows drivers
- 3 different product variants available



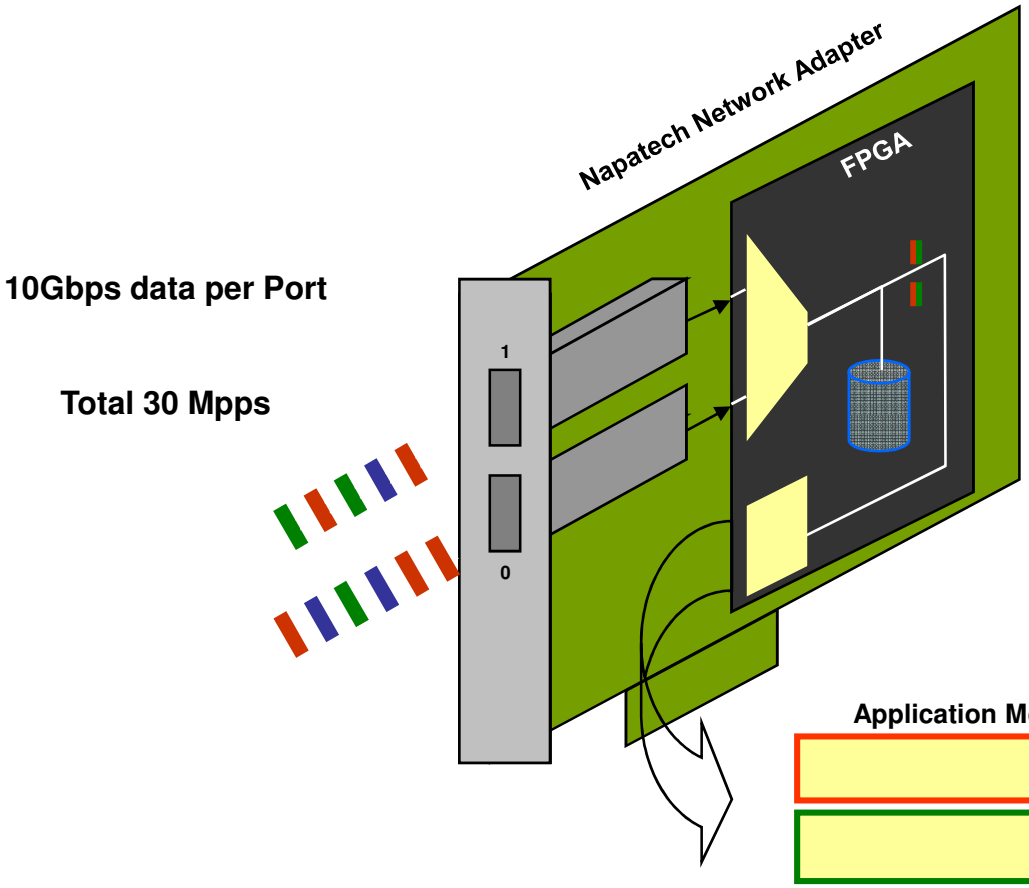
NT20E Adapter

- 2x10Gb Ethernet interface XFP
- ½ Length PCI-E
- External time sync connector
- 20G packet processing, filtering, tagging, timestamp, slicing, local retransmit
- CPU utilization: <1%
- Linux, FreeBSD, and Windows drivers

What problem does hardware acceleration solve?

- Network traffic is growing ***much faster*** than the computing power of standard servers
- Standard NICs are built for efficient data ***communications***, not data ***capture and analysis***
- In order for capture and analysis applications to handle the increased network utilization hardware acceleration is required.

Network processing example



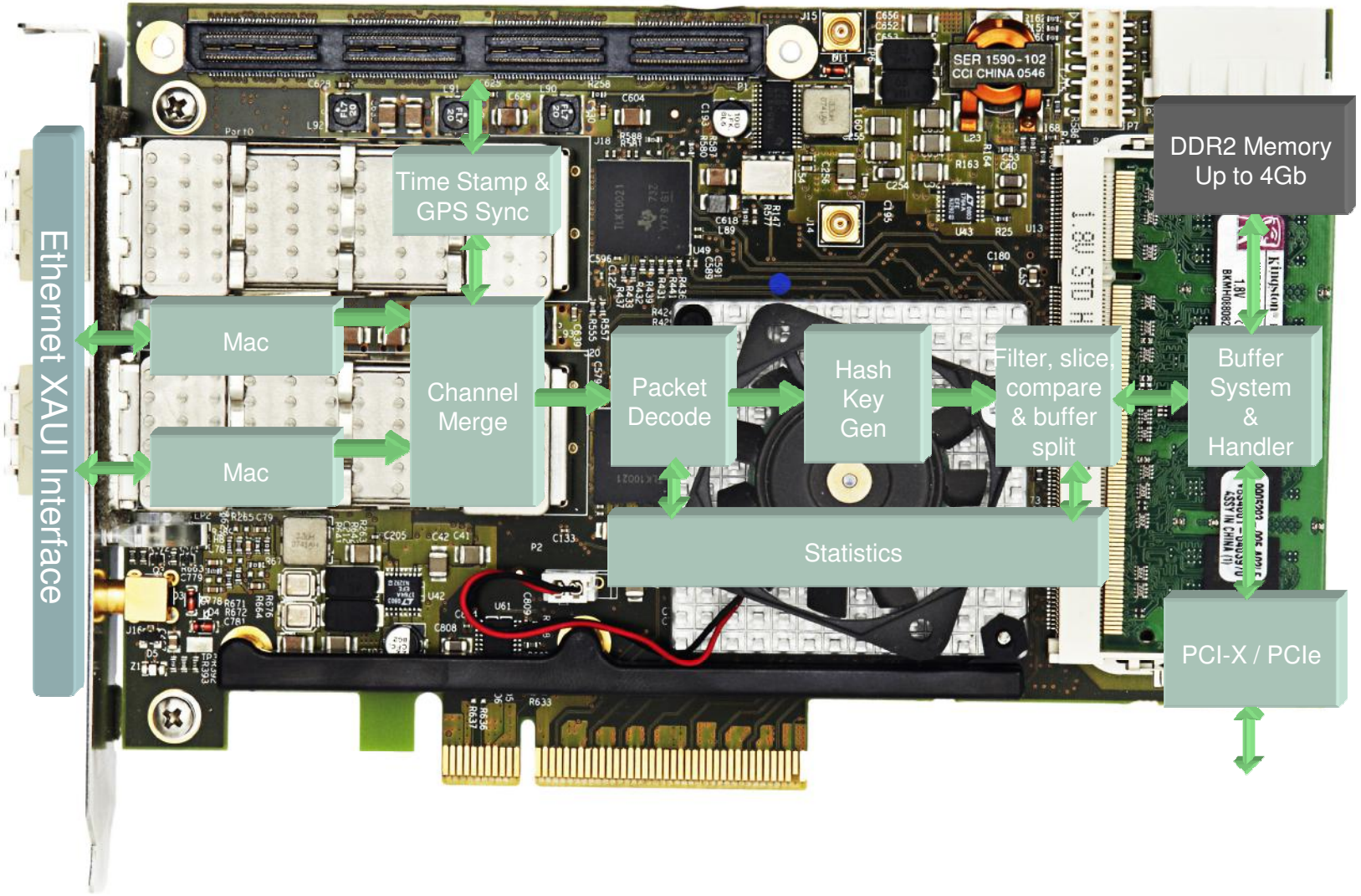
Example of:

- **Channel merge**
- **Filtering**
- **Data type separation in multiple host buffers**

Total CPU load on the server host for all features: < 1%!

napatech

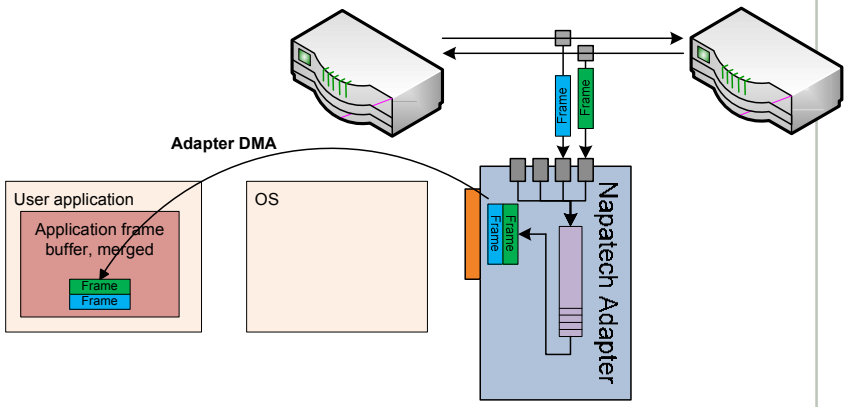
Adapter Architecture

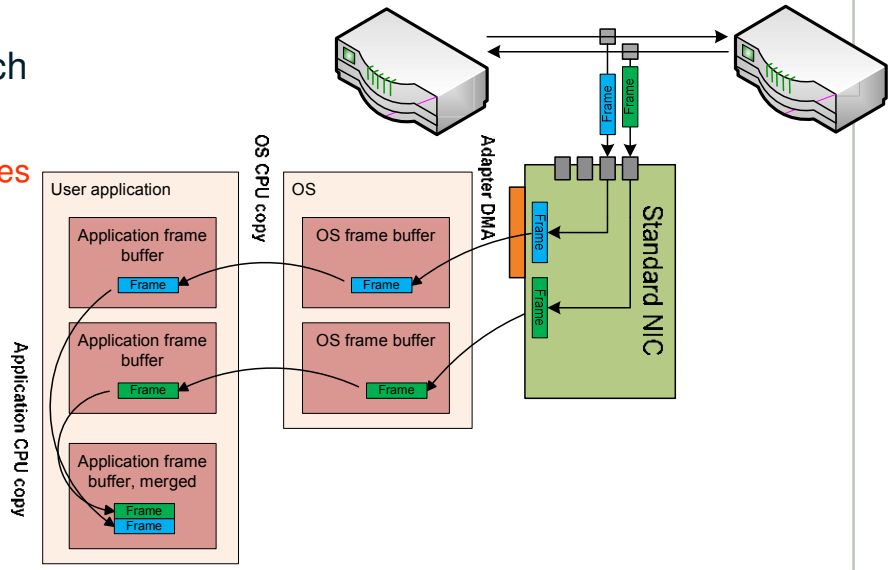


Merging of Streams

- All Napatech adapters support merging of streams.
  - When customer applications need to process both RX and TX data from a link, it is often important to process the request-response traffic in the correct order.
  - The Napatech adapters support merging of data from 2 or more ports into a stream.
  - Merging of data is done based on the frame reception time. This means that request-response traffic will always be delivered to the host in the correct order.
  - Processing of packets in time order can be important:
    - When data is to be analyzed on the fly.
    - When data is to be stored for later analysis.
  - This functionality enables higher host processing performance.

- Standard NICs do not have this functionality, which means that received data must be sorted by the host CPU.
  - Sorting frames in time order by the host CPU reduces the host processing performance.
  - If data is to be stored on disk in time order, an extra CPU memory copy is needed.





Merging of Streams, Continued

- The Napatech adapters support tapping of network data.
  - Only the RX fiber needs to be connected.
  - Line speed tapping using standard network taps is supported.
  - Data from related ports can be merged in time order.
- Napatech recommends the use of network taps instead of switch SPAN ports for tapping of network data (see the table below).

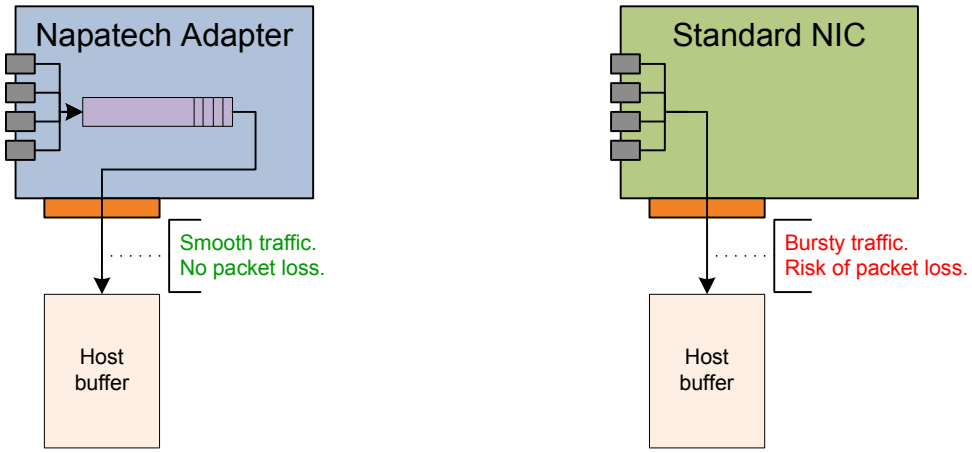
Feature	Tap		SPAN Port	
	Napatech Adapter	Standard Adapter	Napatech Adapter	Standard Adapter
Packets are merged.	Yes	No	Yes	Yes
The time order of packets is correct.	Yes	No	No	No
Data can be captured at all traffic conditions.	Yes	No	No[1]	No[1]
All packets with errors can be captured.	Yes	No	No	No
There are no requirements to the network setup.	Yes	Yes	No[2]	No[2]
No switch configuration is needed.	Yes	Yes	No	No

Notes:

1. Often the switch SPAN has the same speed (e.g. 1 Gbps) as the ports it is monitoring, so if two 1 Gbps switch ports are mirrored to a 1 Gbps switch SPAN port, data gets lost if the network load on the two mirrored ports is higher then 50%.
2. The used switch must have a free SPAN port.

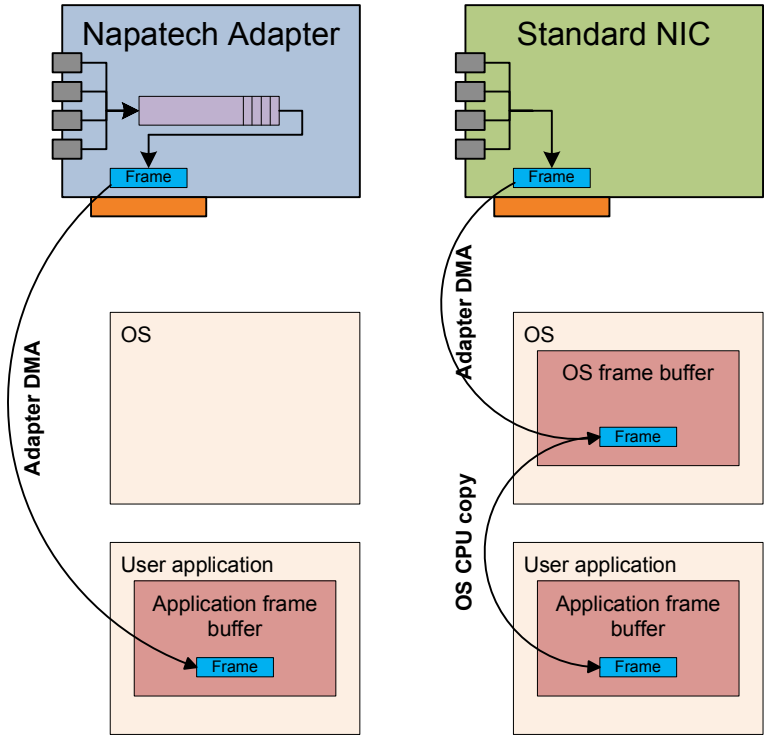
Frame Burst Buffering on Adapter

- All the Napatech adapters have on-board memory that can buffer network traffic:
  - The NT20E adaptercan buffer up to 1200 ms of traffic in the on-board memory if the PCI bus is busy.
  - This feature is important when the PCI interface or the host application does not have the needed performance to capture bursts at line speed.



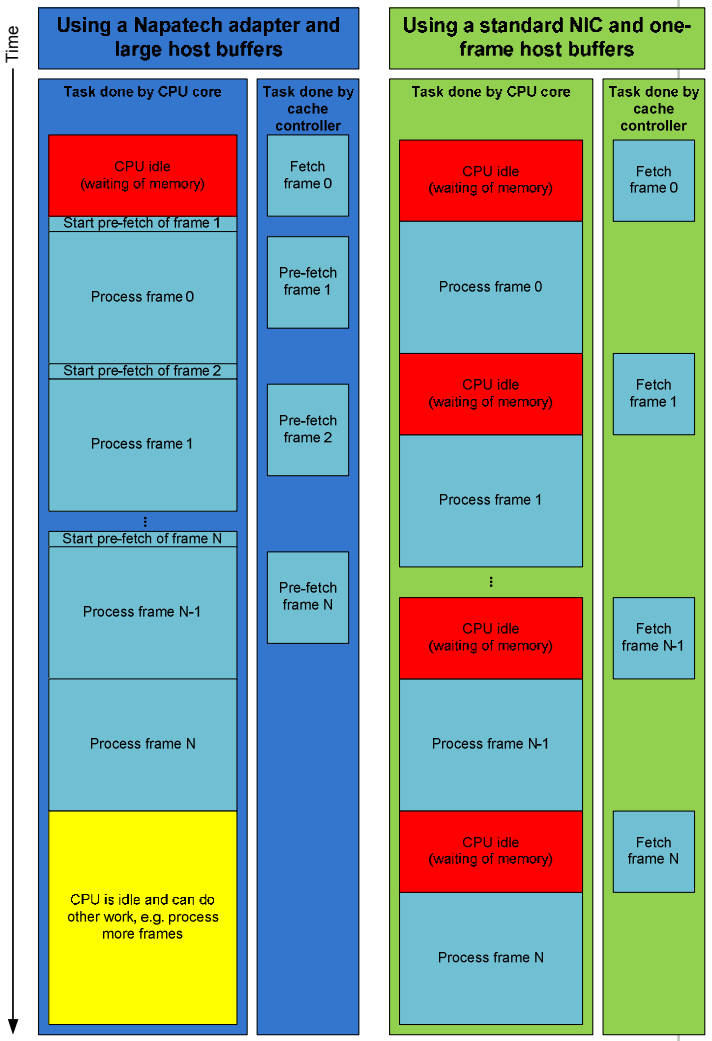
OS Bypass, Zero Copy

- All Napatech adapters support zero copy of captured frames directly from the adapter memory to the user application memory (bypassing the operating system).
- The saving of avoiding having the OS to copy all frames is considerable.
  - An NT20E Napatech adapter, using the zero copy interface, will use less than 1% of one CPU core to deliver 12 Gbps data to the user application memory.



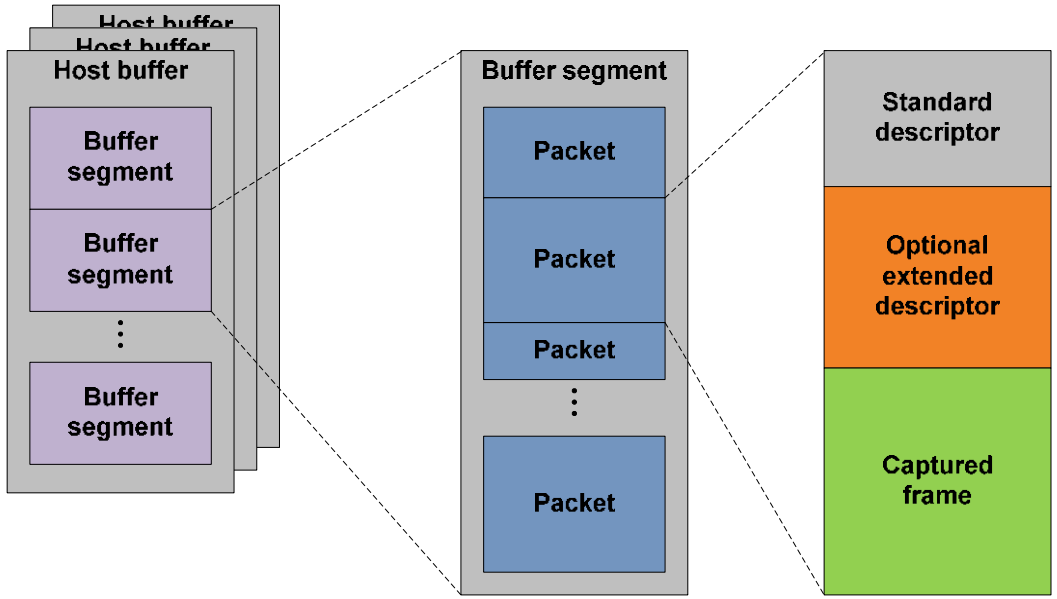
Large Host Buffers

- All Napatech adapters support large host buffers (limited by hardware address space).
- There are two benefits of using large host buffers:
  - The overhead introduced by the driver and the operating system (OS) is kept to a minimum, as many packets can be passed to the application at a time.
  - The application can improve host CPU caching and thereby the host performance. This is done by pre-fetching the frames to be processed, so that frames are available in the CPU cache when they are needed by the CPU.
    - **Napatech has measured that pre-fetching frames before they are needed by the CPU can give more than a 100% increase in processing speed.**
- Standard NICs deliver frames to the host one at a time in separate host buffers:
  - The driver and the OS must process frames one at a time resulting in a large processing overhead. At the same time pre-fetch of frames is not possible.
    This results in a lower host processing speed by a factor of 2 or more.



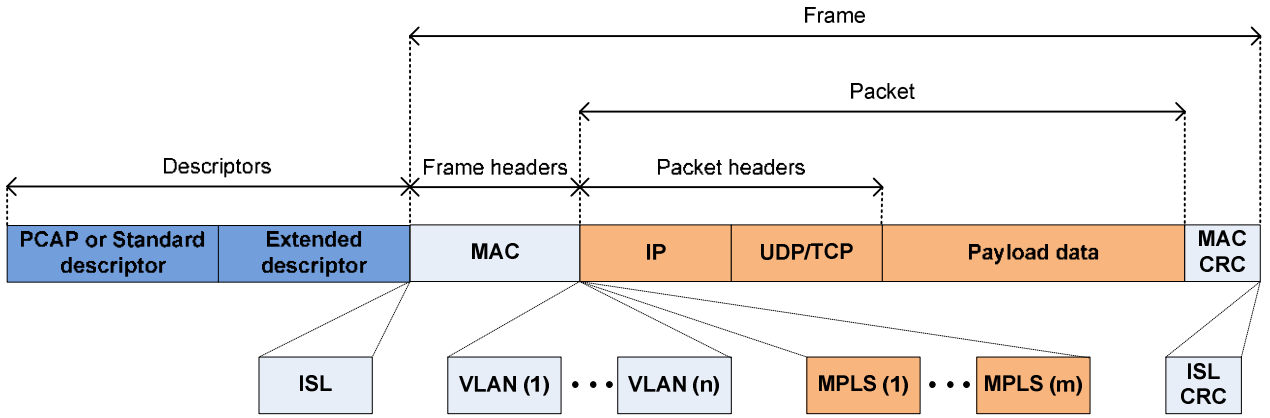
Large Host Buffers, Continued

- Napatech Host Buffers
  - 1, 2, 4, 8, 16 or 32 host buffers can be created.
  - Each host buffer split into segments.
  - Number of segments and the segment size are user configurable.
  - Buffers can be assigned to physical ports or can be port-independent.
  - Adapter can direct traffic to individual buffers based to IP flow and or protocol filter



Packet Descriptors

- Headers vs. Descriptors
  - PCAP, “Standard” and “Extended” **headers** are referred to as PCAP, “Standard” and “Extended” **descriptors**.
    - To distinguish them from network headers (e.g. MAC, IP, UDP or TCP).

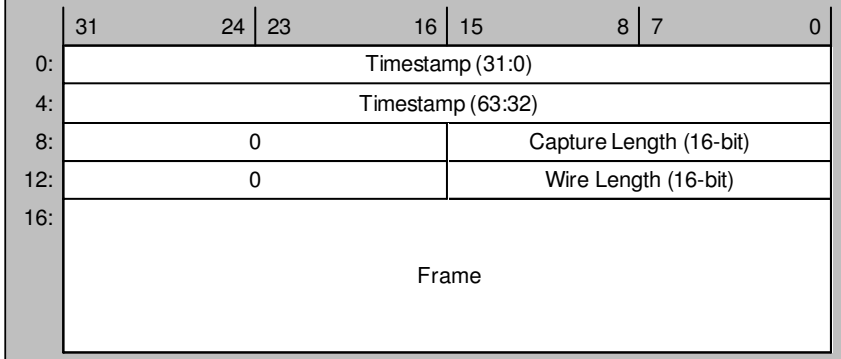


- Descriptors are delivered to the application along with the entire Ethernet frame
  - Descriptors are created by the hardware and inserted into the packet buffer along with the captured frame.

Feature Details – Descriptors

- PCAP descriptor:
  - Complient to standard PCAP format
  - Fixed 16 bytes size
  - 6 different time stamp formats supported

PCAP Descriptor



- Napatech native Standard Descriptors provide additional information about frame:
  - Timestamp, Wire Length and Capture Length fields located at same locations as for PCAP descriptor
  - Other information like: Ethernet CRC and CV errors, IP, TCP, UDP check sums, etc.
- Napatech native Extended Descriptors provide additional packet classification data:
  - TCP/UDP source and destination ports
  - Additional layer 2-3 decodes (VLAN, MPLS, ISL, ARP, IPv4, IPv6, etc.)
  - 2 and 5 tuple hash value
  - Offsets to IP, TCP and UDP portion of frame.

Line Speed Capturing Review

Napatech Adapter Feature	Benefit	Standard Network Adapters
Frame burst buffering on adapter	No data is lost, even when captured data bursts exceed the PCI interface speed, or the PCI interface is temporarily blocked. For NT20X 2 x 10 Gbps can be handled down to 150 bytes frames. For NT20X 1 x 10 Gbps can be handled at any frame size.	Data is lost, when captured bursts exceed the PCI interface speed, or the PCI interface is temporarily blocked.
Long PCI bursts	Very high PCI performance can be achieved for all frame sizes.	The PCI performance will depend on the frame size. E.g. the overhead for a 64-byte frame can be as much as 45%, while for a 1-KB frame it will be only 5%.
Large host buffers	Data can be processed at much higher speed, and the frame processing overhead is much lower (releasing processing power to the user application).	Frames are handled one at a time giving a large processing overhead resulting in lower user application speed.
OS bypass, zero copy of captured packets directly to user application memory	There is no packet copying or OS handling overhead.	A standard OS packet handling interface performs one or more copy of all frames resulting in lower application speed.
Merging of streams	Adapters can merge packets received on 2 or more ports in reception time order, whereby the host CPU is off-loaded.	The sorting of frames in time order must by done by the host CPU, reducing the possible host processing performance.

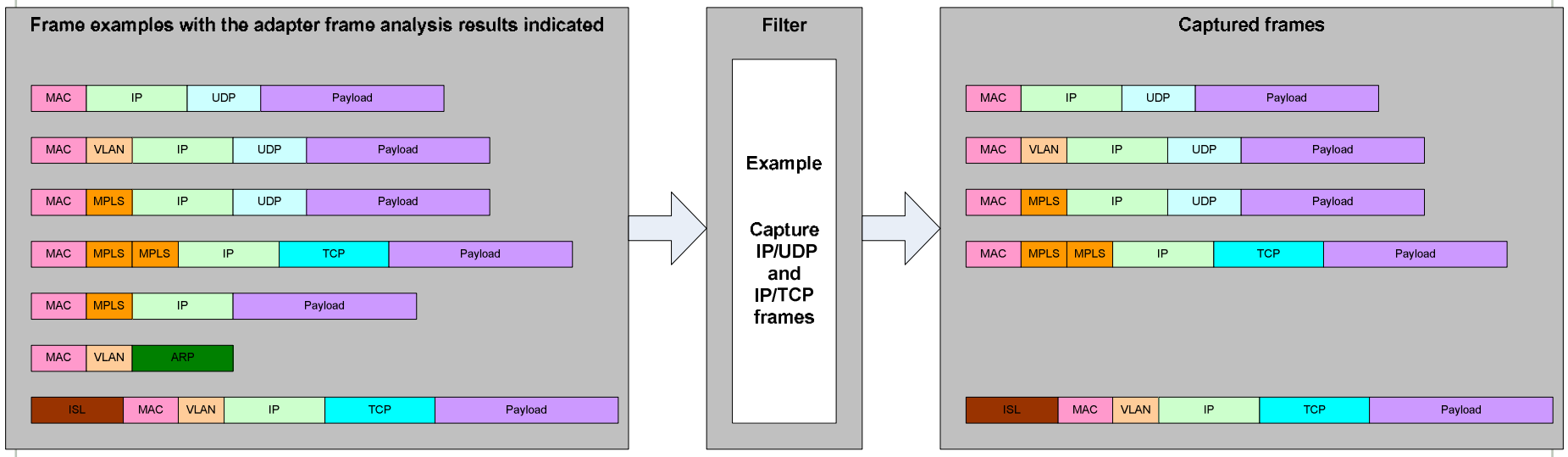
Programmable Adapter Filters

- Filter functionality:
  - 64 fully programmable filters.
  - Received frames can be filtered at full line speed for all frame sizes and all combinations of filter settings.
  - Dynamic offset of filters, based on automatic detection of packet type:
    - 26 predefined fields (Ethernet, IPv4, IPv6, UDP, TCP, ICMP, …)
      (see next slide for example)
  - Fixed offset relative to dynamic offset position.
  - Predefined filters: IPv6, IPv4, VLAN, IP, MPLS, IPX.
  - 64-byte patterns can be matched.
  - The length of the received frame can be used for filtering frames.
  - The port on which the frame was received can be used for filtering.
- Benefits:
  - Enables filtering of network frames so that the user application only needs to handle relevant frames, off-loading the user application.
  - Filtering can be done at network line speed.

Programmable Adapter Filters, Continued

- Filter Example
  - The figure below illustrates how filters can be used to capture IP/UDP and IP/TCP frames
  - NTPL syntax:

```
Capture[Priority=0; Feed=0] = ((Layer3Protocol == IP) AND
        ((Layer4Protocol == UDP) OR (Layer4Protocol == TCP)))
```



Fixed Slicing

- All Napatech adapters support fixed slicing.
- Using fixed slicing it is possible to slice captured frames to a fixed maximum length before they are transferred to the user application memory.
- The fixed slicing is configured using the NTPL:
  - Example: Slice captured frames to a maximum length of 128 bytes:

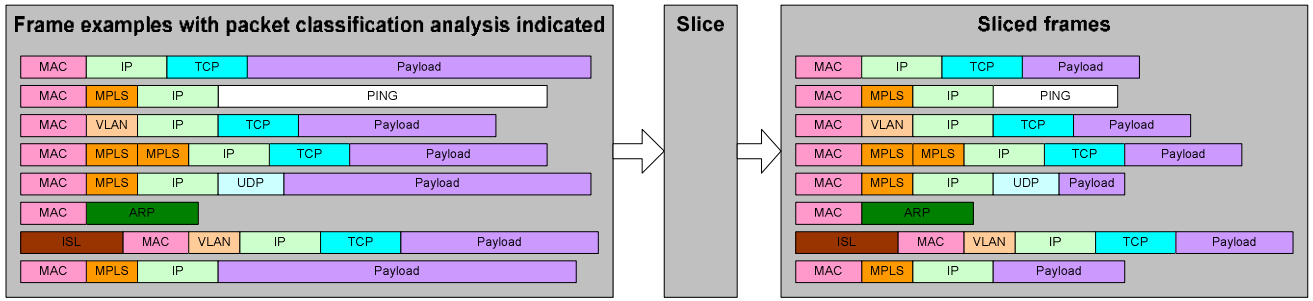
```
Slice[Priority=0; Offset=128]  = all
```

  - **Note: snaplen in libpcap translated automatically to slicing in hardware**
- **For many use cases it will be much more useful to use the dynamic slicing functionality of the Napatech adapters**
  (see next slide).

Dynamic Slicing

- All Napatech adapters support dynamic slicing :
  - Dynamic slicing functionality is supported:
    - Slicing relative to a recognized protocol header (e.g. 32 bytes after the UDP header)
    - Slicing based on filtering results (e.g. different slicing for UDP and TCP frames)
  - Slicing priorities are supported:
    - Enabling different slicing of different frame types (see example below)
- The dynamic slicing functionality is based on the packet classification of dynamic offset information.
- The dynamic slicing is configured using the NTPL:
  - Example: Dynamic slicing using priorities and filtering: Slice TCP frames to a length of the TCP header + 32 bytes, other IP frames to a length of the IP header + 32 bytes and all other frames to a length of 128 bytes:

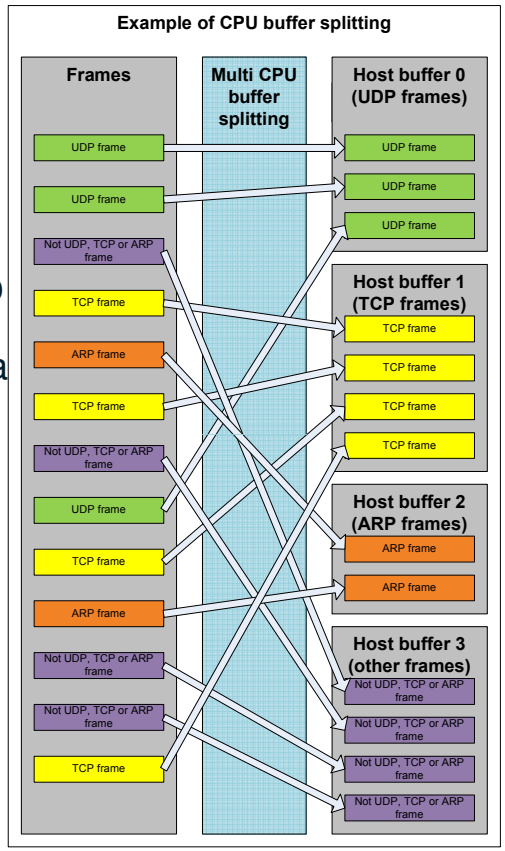
```
Slice[Priority=2; Offset=128]  = all
Slice[Priority=1; Offset=32; Addheader=Layer2And3HeaderSize]     = (Layer3Protocol == IP)
Slice[Priority=0; Offset=32; Addheader=Layer2And3And4HeaderSize] = (Layer4Protocol == TCP)
```



- The dynamic slicing enables reduction of the amount of data that must be transferred to the host memory and that the user application needs to handle.

Multi CPU Buffer Splitting

- Multi CPU buffer splitting enables the adapter to distribute the processing of captured frames among the host CPUs.
  - CPU load distribution is hardware-accelerated.
  - Captured data is placed in separate buffers for the different CPU cores in the host system.
- The multi CPU buffer splitting functionality can be configured to place data in 1, 2, 4, 8, 16 or 32 different host buffers.
  - The algorithm used by the adapter for placing a captured frame in a host buffer is based on packet flow information and or protocol filter.
- Flows can be defined by:
  - The results from the filter logic including port numbers
  - The generated hash key value
  - A combination of the above 2 possibilities



Multi CPU Buffer Splitting, Continued

- NTPL is used to define multi CPU host buffer splitting.
  - Example 1 (using the filter logic, see also the figure at previous slide):

```
HashMode = None
Capture[Priority=0; Feed=0] = (Layer4Protocol == UDP)
Capture[Priority=0; Feed=1] = (Layer4Protocol == TCP)
Capture[Priority=0; Feed=2] = (Layer3Protocol == ARP)
Capture[Priority=0; Feed=3] = (((Layer4Protocol !== UDP) AND (Layer4Protocol != TCP)) AND
                               (Layer3Protocol != ARP))
```

  - Example 2 (using 5-tuple hash, data distributed to 16 host queues):

```
HashMode = Hash5Tuple
Capture[Priority=0; Feed=(0..15)] = All
```

  - Example 3 (using a combination of filter logic and hash key to define flows):

```
HashMode = Hash5TupleSorted
Capture[Priority=0; Feed=(0..3)] = (mUdpSrcPort == (16000..16500))
Capture[Priority=0; Feed=4,5]    = (mTcpSrcPort == mTcpPort_HTTP)
Capture[Priority=0; Feed=6]      = (((Layer3Protocol == IP) AND
                                     (mUdpSrcPort != (16000..16500))) AND
                                    (mTcpSrcPort != mTcpPort_HTTP))
Capture[Priority=0; Feed=7]      = (mMacTypeLength == mMacTypeLength_ARP)
```

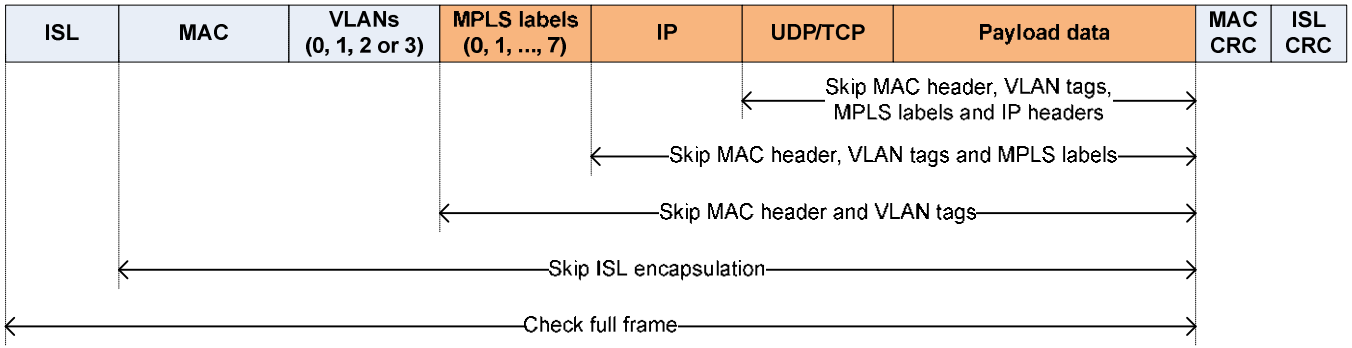
Multi CPU Buffer Splitting, Continued

- What is the advantage of multi CPU buffer splitting?
  - It enables/simplifies distribution of the processing of the captured data on multiple CPU cores.
    - ⇨ Providing a higher user application performance.
  - The intelligent distribution of captured frames (e.g. using a sorted 5-tuple hash key) on multiple buffers will increase the cache hit rate of the host CPU cores.
    - ⇨ Improved cache hit rate can significantly increase the user application performance.
  - The Napatech multi CPU buffer splitting implementation supports the Linux Numa memory localization functionality. In AMD multi CPU configurations it means that memory accesses can be distributed on multiple memory controllers.
    - ⇨ The increased memory bandwidth can increase the user application performance.

`

Deduplication

- Deduplication is a functionality that discards duplicate frames.
- Duplicate frames are typically received:
  - When the network is monitored at a router span port, or
  - When monitoring two MPLS links used to implement a replicated MPLS link.
- The NT adapter recognizes a frame as a duplicate frame when:
  - The two frames are received on the same physical port or on the same port set.
  - There is a time and frame limit for discarding duplicate frames.
  - The dynamic compare offsets match.
- The deduplication functionality generates a per-port statistic over the number of frames being discarded.
- Duplicate frames are normally not 100% identical.
- The following dynamic compare offsets can be configured (see figure below):



Feature Details – Time Stamping

- Six formats:
  - 64-bit free running with 10 ns resolution
  - Windows NDIS format 100 ns 64-bit time stamp, base 1/1-1601
  - PCAP format 64-bit time stamp (µs:sec), base 1/1-1970
  - PCAP format 64-bit time stamp (ns:sec), base 1/1-1970
  - Native NDIS format: 10 ns increment 64-bit time stamp, base 1/1-1601
  - Native UNIX format: 10 ns increment 64-bit time stamp, base 1/1-1970
- Time stamp formats created in hardware
  - Napatech kernel driver never touches packet data.
  - No need to convert native header format to PCAP
- External time synchronization sources
  - GPS
  - CDMA (PPS)
  - IEEE 1588 (third party interface)
- Mutiple adapters can be slaved together

napatech

Host-based Transmit

- High-speed host-based transmit can be used for:
  - Retransmit of captured data
  - Replay of data that have been captured and stored to a file
  - Generation of traffic
- The inter frame gap between transmitted frames can be controlled with high precision (typically better than 20 ns).
- A captured frame can be retransmitted without modifying the packet descriptor:
  - A captured frame inside a segment buffer can be skipped (on transmitted) by flipping a single bit in the standard descriptor.
- The adapter can be set up to:
  - Either generate a new Ethernet CRC for transmitted frames, or
  - Transmit the frames with the Ethernet CRC in the host memory:
    - Being the Ethernet CRC of the received frame or
    - An Ethernet CRC generated by a host application.
- Ethernet CRC generation can be configured on a per-frame basis.
- Frames can be transmitted at the speed supported by the PCI interface:
  - For 8-lane PCIe typically 10-12 Gbps.
  - High-speed transmit is supported for all frame sizes.

Napatech LibPCAP Library

- The Napatech LipPCAP is based on the LipPCAP 0.9.8 release
- Delivered as open source ready to configure and compile.
- Linux and FreeBSD supported
- Support for all feed configurations supported by the NT adapters:
  - Packet feeds can be configured at driver load time
  - Feeds are configured using simple NTPL syntax.
  - Feeds are started and stopped through libpcap
- Full support for protocol filters and deduplication configuration via NTPL scripts.
- Snaplen (-s) option translated to slicing in hardware

Napatech LibPCAP Library

- Example. Build and install of new LibPCAP:
  - Extract standard LibPCAP distribution:
    # tar xfz napatech_libpcap_0.9.8-x.y.z.tar.gz
  - Configure LibPCAP:
    # autoconf
    # ./configure --prefix=/opt/napatech --with napatech=/opt/napatech
  - Build the shared library version of LibPCAP:
    # make shared
  - As root, install the shared library:
    # make install-shared
- Simple Wireshark installation:
  # ./configure –with-libpcap=/opt/napatech
  # make
  # make install

Napatech LibPCAP Library

- Configuration example showing how to setup adapter to capture HTTP frames and distribute them to 8 host buffers using a 5-tuple hash key.

```
DeleteFilter = All
SetupPacketFeedEngine[ TimeStampFormat=PCAP; DescriptorType=PCAP;
    MaxLatency=1000; SegmentSize=4096; Numfeeds=8 ]
PacketFeedCreate[ NumSegments=128; Feed=(0..6) ]
PacketFeedCreate[ NumSegments=16; Feed=7 ]
HashMode = Hash5TupleSorted
Capture[ Feed = 0..6 ] = mTcpSrcPort == mTcpPort_HTTP
Capture[ Feed = 7 ] = Layer3Protocol == ARP
```

  - Eight LipPCAP applications can be started to handle frames from the “ntxc0:0”,“ntxc0:1”, “ntxc0:2”, … “ntxc0:7” virtual adapter devices.

Sharkfest 2009 Demonstration